

Springer Undergraduate Texts
in Mathematics and Technology

SUMAT

Hans-Joachim Bungartz
Stefan Zimmer
Martin Buchholz
Dirk Pflüger

Modeling and Simulation

An application-oriented introduction

Springer Undergraduate Texts in Mathematics and Technology

Series Editors:

J. M. Borwein, Callaghan, NSW, Australia
H. Holden, Trondheim, Norway

Editorial Board:

L. Goldberg, Berkeley, CA, USA
A. Iske, Hamburg, Germany
P.E.T. Jorgensen, Iowa City, IA, USA
S. M. Robinson, Madison, WI, USA

For further volumes:
<http://www.springer.com/series/7438>

Hans-Joachim Bungartz • Stefan Zimmer
Martin Buchholz • Dirk Pflüger

Modeling and Simulation

An Application-Oriented Introduction

Translated from the German by Sabine
and Richard Le Borne



Springer

Hans-Joachim Büngartz
Department of Informatics
Technische Universität München
Munich, Germany

Stefan Zimmer
Dirk Pflüger
IPVS
University of Stuttgart
Stuttgart, Germany

Martin Buchholz
Realtime Technology AG
Munich, Germany

Translators
Sabine and Richard Le Borne
Department of Mathematics
Tennessee Technological University
Cookeville, TN, USA

ISSN 1867-5506

ISBN 978-3-642-39523-9

DOI 10.1007/978-3-642-39524-6

Springer Heidelberg New York Dordrecht London

ISSN 1867-5514 (electronic)

ISBN 978-3-642-39524-6 (eBook)

Mathematics Subject Classification (2010): 00A72, 00A71, 68U20, 65C20

English translation of “Modellbildung und Simulation. Eine anwendungsorientierte Einführung” by Hans-Joachim Büngartz, Stefan Zimmer, Martin Buchholz and Dirk Pflüger, eXamen.press, 2nd edition, Springer-Verlag Berlin Heidelberg 2014

© Springer-Verlag Berlin Heidelberg 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher’s location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

In many areas in the sciences, in particular the natural and engineering sciences, modeling and simulation have established themselves as a third pillar supporting the acquisition of knowledge: Previously, the understanding, prediction, or optimization of the behavior of processes and systems had to be based on either experiments or theoretical analyses. Today, however, we are offered the elegant possibility of conducting “experiments on the computer.” Completely new fields have been developed—either in terms of specializations within classical fields or as independent fields on the interface of existing ones. They all live on the cross-disciplinary interaction between efficient methods (mostly from mathematics or computer science) with exciting application fields and models that are not at all restricted to the natural and engineering sciences, for example, financial mathematics. To this end, the naming convention that we use nowadays suggests the slight differences in emphases: “Scientific Computing” stresses mathematical (in particular numerical) aspects, whereas “High-Performance Computing (HPC)” and “Advanced Computing” both stress computer science aspects. In the former case, HPC, the focus is given to supercomputers, while Advanced Computing is based on an integrative approach to include algorithms, computers, data, and software. Nomenclature such as “Computational Sciences,” “Computational Science and Engineering,” or “Computational Engineering,” on the other hand, indicates that the main interest lies within the subject of simulation.

Despite both the wide spectrum of application areas and the high bandwidth of employed methods—analytical and approximate, numerical and discrete, deterministic and stochastic, originating from mathematics (differential equations, etc.) or from computer science (fuzzy logic, Petri nets, etc.)—modeling and simulation are based on a consistent, systematic approach that is recognized ever more frequently in the relevant courses. In fact, this current book resulted from lecture notes from courses in “Foundations of modeling and simulation” and “Modeling and simulation,” resp., taught by the first two authors several times at the Universität Stuttgart and the Technische Universität München. While primarily tied in with the computer science curriculum, both courses also address students of mathematics as well as students in the technical or natural sciences. It is quite typical, even

inevitable, for each student to be somewhat familiar with the material while simultaneously being embraced with something new.

Because our interest is in modeling and simulation as a systematic approach, material that is supposedly familiar will be presented in a new light while previously hidden connections will be revealed and the eye will be trained to recognize not only the particular solution approach but also the underlying systematic approach—an essential intention of this book. In this context and of major importance: Our goal is not to provide an introduction to the basic use of existing tools, however widespread and powerful they may be, but rather to provide a point of entry into the exciting world of building better tools.

Considering the complexity of the topic it would be absurd to claim, or even to expect, a complete treatment in either breadth or depth. Therefore, while this book touches on some very interesting and relevant applications that differ quite a bit with respect to the methodical approach taken, the selection of topics naturally reflects the personal preferences and experiences of the authors. But then the fundamental similarities existing between the different approaches will be highlighted since these often remain invisible if the topic of modeling and simulation is approached through only a single application domain. The simulation pipeline, beginning with the derivation of the model up to its validation, is a prominent example. Consequent to this and part of the maxim of this book is the repeated concern regarding “a little bit of everything, but nothing in detail” that can be levied against such courses or book concepts. This book is about a first encounter with models and simulations, about gaining an impression of the diversity of tools employed from mathematics and computer science, and about the diversity of problem settings. We will discuss fluid dynamics, but we cannot nor would we want to provide the same level of detail given in a book on computational fluid dynamics; we will mention numerical techniques without listing all of their variants and discussing all of their properties as it is expected from a book on numerical analysis; and all scenarios will be greatly simplified, which typically results in models that are a bit less connected to reality.

We do not want to train the readers of this book to become specialists in individual topics, but rather provide an overview to students in computer science, mathematics, or the natural or engineering sciences—and of course increase their appetite for more.

A further challenge is the balance between modeling and simulation—here to be taken in the more literal meaning of the word, i.e., the part implying the calculation or solution of the models. For example, we will discuss models without losing track of their origins and the goals of the modeling—the simulation. Conversely, we will discuss numerical methods without the models appearing out of the blue.

It is our opinion that a concentration on breadth as well as the dovetailing of connections is very important, even if it is reached at the expense of depth in the individual topics—however, this book is neither exclusively on mathematical modeling nor exclusively on numerical analysis.

There are at least two different possibilities for the structure of this book. One may organize the material based on the employed methodology which would, for example, lead to chapters on models with graphs, models with ordinary differential

equations, or models with partial differential equations. The advantage is methodical rigor. However, certain topics such as traffic simulation would appear multiple times which hinders a comparative consideration and evaluation of alternative approaches.

Such a disadvantage is avoided if the material is organized by subject areas in which one models and simulates—by comparison, this has the effect of turning the above advantages into disadvantages and vice versa. We decided for the second alternative after considering that the resulting structure would be more plausible and attractive especially to the beginner. Furthermore, the important message concerning the existence of more than one possible model and more than one applicable apparatus from mathematics or computer science might be delivered better through the second approach.

Chapter 1 provides an introduction into the topics of modeling and simulation. It begins with a presentation of the simulation pipeline and the simulation cycle, resp. Next, we discuss general questions concerning mathematical models—e.g., concerning their derivation, analysis of their properties, existence and uniqueness of solutions, model hierarchies and model reduction—and the transition into a simulation process. Chapter 2 concisely provides the methodical tools that are subsequently required from the various areas of mathematics and computer science. Once more we have to choose one out of the several possible strategies—ranging from the uncompromising (and room saving) “this and that is all assumed to be known” approach up to the caring (and exceeding the scope of any textbook) “all that is needed will be explained” approach. We opted for a compromise: a brief summary of elementary mathematics, discrete mathematics, linear algebra, analysis, stochastics, statistics, and numerics. Hence, all the essentials are included, however, not in epic breadth. Most of the topics should, in more or less detail, be material of the introductory or bachelor level courses in the respective fields of study. However, we will provide numerous references so that possible gaps can be filled efficiently. To facilitate the classification and to support a selective reading of this book, we will always begin the application scenarios with explicit links to the required tools.

The subsequent chapters are grouped by topics into four parts and treat different areas exemplarily in which models and model-based simulation are employed to a large extent.

Part I is devoted to the topic “Gaming—deciding—planning.” Here, we discuss problem settings in the areas of game theory (Chap. 3), decision theory (Chap. 4), scheduling (Chap. 5), as well as mathematical finance (Chap. 6).

Part II covers modeling and simulation in the area of the transportation system—an area which is well suited to illustrate the diversities of problem settings, approaches, and employed tools. Here, we first introduce a macroscopic simulation of road traffic using simple models based on differential equations (Chap. 7). In contrast, the classical microscopic approach is based on cellular automata (Chap. 8). A completely different approach is offered in stochastic traffic simulation in which queuing systems provide the central descriptive tool (Chap. 9).

Part III treats scenarios from the wide field of dynamical systems. The first respective insight is given by the classic topic of population dynamics (Chap. 10). Control engineering will serve as an example to cover the conventional approaches

for the control of technical systems, e.g., multibody systems or fuzzy control systems (Chap. 11). This part is concluded by a brief excursion into the world of chaos (Chap. 12).

Part IV, the final part, discusses topics with a strong connection to physics—topics which typically lead to computationally intense simulations and thus are closely linked to high-performance computers. We will discuss molecular dynamics as the representative from particle methods (Chap. 13) and then continue with two representatives from models based on partial differential equations, namely heat conduction (Chap. 14) and fluid dynamics (Chap. 15). The concluding scenario, the modeling and calculation of realistic global illumination, illustrates the use of physically motivated models and simulation in computer science, or more precisely, in computer graphics (Chap. 16).

As mentioned before, this book is also suited for a selective treatment in courses or for selective self-study of topics of interest. Combined with the introductory Chap. 1 as well as the respective prerequisites of Chap. 2, each part presents a self-contained unit and can be studied as such or covered in a class, resp.

As always, quite a few have contributed to this book. We owe special thanks to our colleagues in the department—for critical comments, helpful hints, and some of the figures—as well as to the students of the previously mentioned courses who naturally contributed through their questions and comments to this final version of the book.

Last but not least, we have the pleasure to see the end product of an endeavor that gives us the English edition of our book. The translation was worked in parallel with, and is the counterpart of, the second edition that is in German. As with any production of scale, there are many helping hands that have our gratitude: This book was the idea of Dr. Martin Peters from Springer and it was his enthusiasm that convinced us to begin the project. Once started, it was the Springer team that provided the foundation for a successful partnership. But, perhaps most important for a translated work are the translators—we are deeply grateful to Professor Sabine Le Borne (Hamburg University of Technology) and Professor Richard Le Borne (Tennessee Technological University). They managed to preserve our style which is characterized through our attempt to never become too dry and sterile even when immersed in a topic that might appear as such.

Munich, Germany
Stuttgart, Germany
Munich, Germany
August 2013

H.-J. Büngartz
S. Zimmer, D. Pflüger
M. Buchholz

Contents

1	Introduction	1
1.1	The Simulation Pipeline	1
1.2	Introduction to Modeling	4
1.2.1	General Prerequisites	5
1.2.2	Derivation of Models	7
1.2.3	Analysis of Models	9
1.2.4	Classification of Models	11
1.2.5	Scales	11
1.3	Introduction to Simulation	13
1.3.1	General Remarks	13
1.3.2	Assessment	14
2	Required Tools in Short	17
2.1	Elementary and Discrete Topics	18
2.2	Continuous Aspects	19
2.2.1	Linear Algebra	19
2.2.2	Analysis	21
2.2.3	Significance for Modeling and Simulation	28
2.3	Stochastic and Statistical Issues	29
2.3.1	Why Randomness?	29
2.3.2	Discrete Probability Spaces	30
2.3.3	Continuous Probability Spaces	36
2.3.4	Asymptotics	41
2.3.5	Statistical Inference	43
2.4	Numerical Aspects	47
2.4.1	Basics	47
2.4.2	Interpolation and Quadrature	51
2.4.3	Direct Solution of Linear Systems of Equations	59
2.4.4	Iteration Methods	61

2.4.5	Ordinary Differential Equations	68
2.4.6	Partial Differential Equations	78
2.5	Interrelationships Between Tools and Applications	83

Part I Gaming—Deciding—Planning: A Warm-up for Modeling

3	Game Theory	87
3.1	Games in Strategic Normal Form.....	88
3.2	Games Without Assumptions on the Opponent	90
3.3	Response Mappings	91
3.4	Dominant Strategies.....	93
3.5	Nash Equilibria	94
3.6	Mixed Strategies.....	95
3.7	Outlook	97
4	Group Decision Making	99
4.1	Individual Preferences and Group Decision Making	100
4.2	Examples for Decision Making Methods	103
4.3	Conditions for Choice Functions, Arrow's Theorem	106
5	Scheduling	111
5.1	Process Scheduling (Deterministic)	112
5.2	Process-Scheduling (Stochastic).....	118
5.3	Job-Shop Problems	124
5.4	Further Scheduling Problems	128
6	Wiener Processes	131
6.1	From the Bernoulli Experiment to the Normal Distribution	132
6.2	Normally Distributed Input Parameters	134
6.3	Wiener Processes	135
6.4	Application: Development of Money Investments.....	138

Part II Traffic on Highways and Data Highways: Once Through the Simulation Pipeline

7	Macroscopic Simulation of Road Traffic	149
7.1	Model Approach.....	150
7.2	Homogeneous Traffic Flow	152
7.2.1	An Initial Result	152
7.2.2	Velocity, Flow and Density	153
7.2.3	Fundamental Diagram	154
7.2.4	Model Refinements	155
7.3	Inhomogeneous Traffic Flow	158
7.4	Simulation of a Simple Circular Road	160
7.4.1	A First Attempt	161
7.4.2	An Improved Simulation	163

7.5	Signal and Traffic Velocity	165
7.6	Summary and Outlook	169
8	Microscopic Simulation of Road Traffic	171
8.1	Model Approach.....	172
8.1.1	Cellular Automata	172
8.1.2	Road Traffic	174
8.2	A First Simulation	176
8.3	Stochastic Extension: Randomization	178
8.3.1	Free Traffic Flow	179
8.3.2	Higher Densities, Traffic Jams out of Nowhere	180
8.3.3	Validation and Calibration: Fundamental Diagrams	182
8.4	Modeling of Traffic Networks	186
8.4.1	Traffic Graph.....	186
8.4.2	Intersections.....	188
8.4.3	Plans and Intentions	193
8.5	Model Refinements.....	199
8.6	Summary and Outlook	201
9	Stochastic Traffic Simulation	203
9.1	Model Approach.....	204
9.2	Queuing Systems	206
9.2.1	Stochastic Processes	207
9.2.2	Classification of Elementary Queuing Systems	213
9.2.3	Examples for the Kendall Notation	214
9.2.4	Performance Indices and First Results	215
9.3	Queuing Networks	218
9.3.1	Parameters in Queuing Networks	219
9.3.2	Asymptotic Analysis	220
9.4	Analysis and Simulation	222
9.4.1	Markov Processes and Markov Chains	223
9.4.2	Queuing Systems	229
9.4.3	Queuing Networks	234
9.4.4	Simulation	235
9.5	Summary and Outlook	237

Part III Dynamical Systems: Cause, Effect, and Interplay

10	Population Dynamics.....	241
10.1	Malthusian Growth Model	242
10.2	Refined Single Species Models	242
10.2.1	Linear Model with Saturation	243
10.2.2	Logistic Growth	243
10.3	Two Species Models	245
10.4	A Discrete Single Species Model.....	250

11 Control Engineering	255
11.1 The Basics of Control Theory	256
11.1.1 Control Loop	257
11.1.2 Description of Linear Dynamical Systems	258
11.1.3 Requirements for the Controller	258
11.1.4 PID Controller	259
11.2 Exemplary Modeling of a Multibody System	261
11.2.1 Linearized Model with Conservation of Linear and Angular Momentum	263
11.2.2 Complete Model with Lagrange Equations	266
11.2.3 Simulation of the Pendulum	270
11.3 Fuzzy Set Theory	271
11.3.1 Membership in Fuzzy Sets	271
11.3.2 Operations with Fuzzy Sets	274
11.3.3 Linguistic Variables	276
11.3.4 Fuzzy Logic	277
11.4 Rule-Based Fuzzy System	280
11.4.1 Fuzzification	281
11.4.2 Inference	282
11.4.3 Defuzzification	283
11.4.4 Example	284
11.5 Fuzzy Control of the Inverted Pendulum	284
11.5.1 Parameters and Constraints	285
11.5.2 Swinging Up the Pendulum	286
11.5.3 Stabilizing the Pendulum	288
11.6 Outlook	288
12 Chaos Theory	291
12.1 Introduction	292
12.2 From Order to Chaos	293
12.2.1 Logistic Mapping and Its Fixed Points	293
12.2.2 Numerical Analysis and Bifurcations	295
12.2.3 Transition into Chaos	298
12.3 Strange Attractors	301
12.3.1 Self-Similarity and Fractal Dimension	302
12.3.2 Hénon Mapping	304
12.3.3 General Two-Dimensional Quadratic Mapping	305
12.4 Chaotic Behavior of a Driven Pendulum	307
12.4.1 Model of the Pendulum	308
12.4.2 Discretization	309
12.4.3 Cycles and Attractors	310

Part IV Physics in the Computer: Take-Off Toward Number Crunching

13 Molecular Dynamics	317
13.1 Modeling of Molecules and Interactions	318
13.1.1 Fundamental Physical Forces	318
13.1.2 Potentials for Uncharged Atoms	319
13.1.3 Computation of the Force Acting on an Atom	323
13.2 Equations of Motion and Their Solutions	324
13.2.1 Equations of Motion	324
13.2.2 Euler's Method	325
13.2.3 Velocity-Störmer-Verlet	326
13.2.4 Remarks	327
13.3 Simulation Domain	328
13.3.1 NVT Ensemble	328
13.3.2 Boundary Conditions	329
13.4 Implementation	330
13.4.1 Linked-cells Data Structure	331
13.5 Parallelization	333
13.6 Outlook	335
14 Heat Transfer	337
14.1 Derivation of the Heat Equation	338
14.1.1 Number of Dimensions	340
14.2 Discretization	341
14.2.1 3-Point-Stencil	342
14.2.2 5-Point-Stencil	344
14.2.3 Boundary Treatment	346
14.3 Numerical Solution of the PDE	346
14.3.1 Simple Relaxation Methods	347
14.3.2 Multigrid Methods	348
15 Fluid Dynamics	355
15.1 Fluids and Flows	356
15.2 Mathematical Model	357
15.2.1 Navier–Stokes Equations	357
15.2.2 Remarks Concerning the Derivation	360
15.3 Discretization of the Navier–Stokes Equations	361
15.3.1 Finite Differences	361
15.3.2 Treatment of Spatial Derivatives	362
15.3.3 Treatment of Temporal Derivatives	363
15.3.4 Treatment of Boundary Conditions	364
15.4 Numerical Solution of the Discretized Equations	365
15.4.1 Time Step	365
15.4.2 Spatially Discrete Momentum Equations	367

15.4.3	Spatially Discrete Poisson Equation for the Pressure	367
15.4.4	Regarding Stability	368
15.5	Application Example: Flow Around an Obstacle	368
15.6	Outlook	370
15.6.1	Problem Settings and Models	370
15.6.2	Discretizations	371
15.6.3	Structured Grids	373
15.6.4	Unstructured Grids	376
15.6.5	Approaches for the Treatment of Changing Geometries	379
16	Global Illumination in Computer Graphics	381
16.1	Quantities from Radiometry	382
16.2	The Rendering Equation	384
16.3	Techniques for the Solution of the Rendering Equation	387
16.3.1	Ray Tracing	388
16.3.2	Path-Tracing	390
16.3.3	Further Ray-Tracing Derivates	391
16.4	The Radiosity Method	392
16.4.1	Basic Principle	392
16.4.2	Computation of the Form Factors	394
16.4.3	Solution of the Radiosity Equation	396
16.4.4	Remarks and Improvements	397
	Concluding Remarks	401
	References	403
	Index	407

Chapter 1

Introduction

What are models and how do we obtain and assess them? How do abstract models turn into tangible simulation results? What are the ever increasing number of “simulators” doing exactly, what constraints apply to their activities and how can their results be validated? These and other questions are discussed in the first chapter of our book. It is designed to be a general introduction as well as a separate introduction to each of the four subsequent parts. The first section of this chapter provides the general terms and definitions that apply to simulation and introduces the so-called simulation pipeline. In sections two and three we provide the basic foundations of modeling and simulation, respectively.

1.1 The Simulation Pipeline

The notion of simulation is quite ambiguous and requires clarification. In the context of this book, two of its interpretations are of particular relevance. In a broader sense, simulation is the complete process of the forecasting or replication of a certain scenario. Since such simulations are nowadays performed almost exclusively computer-based, we will not—as it is oftentimes seen elsewhere—refer to it as computer simulation. In a tighter sense (and in the title of this book), simulation only refers to the central part of this process, i.e., the actual computation—a classical case of a “pars pro toto”. In the following, we will make use of both interpretations and only provide explicit clarification if the respective interpretation is not given implicitly.

In a broader scope, therefore, simulations are nothing other than “virtual experiments” on the computer. This remains unchanged by the fact that in most application areas served by simulation (for example, physics, chemistry or mechanics), the respective representatives of the “computational guild” are typically allocated to the theoreticians.

The attractiveness of these virtual experiments is obvious. For a multitude of cases, “real” experiments are simply impossible due to the underlying time and spatial scales, for example. To illustrate, one needs only to consider astrophysics: no matter how hard-working, it is impossible for any physicist to devote the necessary billions of years at the telescope to study the life cycle of a galaxy; or geophysics—it may be possible to create experimental earthquakes, i.e., artificial earthquakes in a James Bond production, but those are not practical in real life. Moreover, not all that is possible in principle is actually desirable—one only needs to consider the testing of nuclear weapons, animal experiments, or genetic engineering. The former took their leave just at the time when the respective nations reached the ability to execute them in a completely virtual manner on the computer. The ethical component—nuclear bombs do not become friendlier if they are “brought to perfection” through simulations—must not be left out here, but as well will not be discussed further. And even in the remaining set of the feasible and justifiable, the effort is often the limiting factor: The static of buildings, the vulnerability of the HIV virus, the evacuation of a fully filled soccer stadium, economical or military strategies, etc. etc.—all these are not tested quickly, not even in the lab; not to mention the effort that fundamental experiments require in modern physics in the context of the Large Hadron Collider. Thus, there is no way to go without simulation and it is therefore worthwhile to take a closer look at its methodology. However, it is indisputable: Simulations *complement* theoretical analyses and experiments, they do not *replace* them.

The goals pursued by a simulation can be very diverse. Oftentimes, one wants to reconstruct a scenario which is well-known in principle in order to better understand it. This applies for example to catastrophes of a technical or natural kind. Why has an earthquake developed, why at this particular place, why at this particular instant in time? Why did one of the large traffic bridges across the Mississippi River in the US state of Minnesota collapse in August 2007? How could the tsunami in south-east Asia in late December 2004 develop such a devastating effect? The goal to predict unknown scenarios is also knowledge driven, but in general even more challenging. This applies not only to the catastrophes mentioned above (and for possible repetitions, resp.) as well as to urgent questions concerning climate change or the propagation of the world population, but also to many technical questions (properties of new alloys or composite materials). Besides discovery, another goal pertains to improvement, i.e., the optimization of a known scenario. Prominent examples include the (route) scheduling of airlines, the efficiency factor of chemical reactors and the efficiency of heat exchangers or the data throughput in a computer network.

Here, a simulation in the broader sense is not an intergral act, but rather a highly complex process consisting of a sequence of several steps which are traversed several times in various feedback loops.

To this end, the picture of a “simulation pipeline” has been established (see Fig. 1.1). We summarize the essential steps:

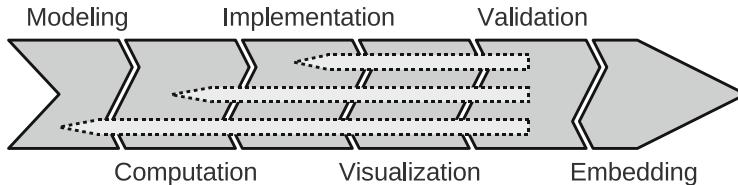


Fig. 1.1 The “simulation pipeline”

- The *Modeling*: At the very beginning we need a model, i.e., a simplified formal description of a suitable extract from the item of interest, which will then serve as the basis for the subsequent computations.
- The *computation or simulation in the tighter sense*, resp.: The model will be preprocessed (e.g., discretized) so that it is compatible with a computer platform. The solution of this preprocessed model requires the identification of efficient algorithms.
- The *implementation* (or more generally the *software-development*): The computational algorithms previously determined must be implemented efficiently (with respect to computational time and storage complexities, parallelization issues, etc.) on the target architecture or architectures. Currently, this step significantly exceeds the implementation in the classical sense: It is no longer sufficient to produce runnable code, but software must be designed and developed on a big scale and by every trick in the book.
- The *visualisation* (or more generally the *data exploration*): The data resulting from a simulation run must be interpreted. In some cases—e.g., for scalar quantities such as the drag coefficient in aero dynamics—this will be easy, in others—e.g., for highdimensional data sets—extracting the relevant information from the flood of numbers is a science of its own.
- The *validation*: Very important—how reliable are the results? Sources for errors lurk in the model, in the algorithm, in the code or in the interpretation of the results. Therefore, it is important to compare different models, different algorithms, and different codes, resp., as well as simulation results with inkind experiments. Depending on the source of the error, the process has to be restarted at the respective step and the pipeline has to be traversed once more starting from this point.
- The *embedding*: Simulations take place in a context—e.g., a development or production process—and should be integrated accordingly. This requires the definition of interfaces, a reasonable software engineering, simple testing environments, etc.

Let us take a look at a descriptive example—a little preview of part IV of this book. The subject of our interest is the automobile in a wind tunnel—or, better, the virtual automobile in the virtual wind tunnel. We want to figure out the wind resistance of the vehicle that, in technical terms, is denoted by the drag

coefficient c_w . For subsonic aerodynamics, the suitable physical or mathematical model, resp., is given by the Navier–Stokes equations, a system of nonlinear partial differential equations. Their discretization in space and time can, for example, be carried out through finite elements or finite volumes. The resulting large and sparse linear systems of equations can be solved efficiently using multigrid methods. In view of the method’s large computational time requirement, the method typically must be implemented on a parallel computer. The visualization of the three-dimensional velocity field will require techniques that exceed the well-known two-dimensional pictures using arrows, and the drag coefficient must be computed in a suitable way from the millions of computed discrete velocity and pressure values. The validation will rely on comparative calculations with other programs as well as experiments using prototypes in a wind tunnel.

The proper way to convey the results of an aerodynamics simulation to the design department (e.g., suggestions pertaining to details involving change in the shape of fenders) is an exciting task of embedding. Or said differently: How can simulation results be directly integrated into the CAD-model without causing the very time intensive design process to be restarted?

It is clearly illustrated through this example: The comprehensive solution of a simulation problem requires far more than “a little bit of computation”—all six steps pose an abundance of challenges to the different fields of science. In order to avoid an immediate misunderstanding: The figure depicting the pipeline should illustrate the diversity and sequential flow of the intermediate tasks, but it should on no account suggest that the individual steps can be worked separately by entirely different experts in a manner much like an assembly line production. In fact, everything is closely interwoven. For example, beginning early on in the design of numerical algorithms, one must keep an eye on an efficient implementation - this typically takes the target hardware into account one way or another.

The first two steps of the simulation pipeline—modeling and simulation in the tighter sense—are certainly of central importance. For this reason, and since they are absolutely necessary for an introduction to the topic, they will be covered in this book.

1.2 Introduction to Modeling

We will direct our attention to the first step of the simulation pipeline, the (mathematical) modeling, by discussing in sequence the following questions: What is actually a model, and what is its purpose? How does one obtain a suitable model? How can mathematical models be assessed? What is the difference between models, and how can they be classified? And finally—does there exist “the correct model”?

1.2.1 General Prerequisites

In general, a model is a (simplified) image of a (partial) reality. In this context, the models are always meant to be abstract, i.e., formal descriptions given mostly (but not always) through the methodological apparatus provided by mathematics or computer science. In the following, when talking about models, we will almost always think of mathematics or computer science models.

Mathematical modeling denotes the process of the formal derivation and analysis of a mathematical model for an effect, a phenomenon, or a technical system. The starting point is in general an informal description of the respective subject of modeling, for example in prose. This is typically converted next into a semiformal description, the model of the science application, using the tools of the application discipline. Finally, an additional step is required to derive a strictly formal model (i.e., unambiguous, consistent)—the mathematical model.

A simple example to illustrate this is the management of a timetable and the room scheduling of a school: To start, we have the textual description of the problem. This leads to the classical arrangement of index cards on the wall of the teachers' room which helps to avoid double bookings of rooms, but is limited in its ability to recognize possibilities for optimization. This situation changes when the problem is “mathematized”, for instance being formulated as a graph-based scheduling problem. Once such an abstraction is performed, one can now apply appropriate methods for the problem's analysis and optimization.

The extent that mathematical modeling is suggestive and established differs greatly in various areas of science. It has a very long tradition in the exact natural sciences. Several formulations of theoretical physics, for example, are per se mathematically rooted and in many areas recognized as such. This holds in particular where they have been validated through experimental data (as, e.g., in classical continuum mechanics). A completely different situation prevails for national economic policy. In view of the substantial impact the psychology of the moment contributes, the extent that mathematical models are supportable is questionable. And even if there is consensus, the choice of the “right” model is by no means obvious. Further, depending on the choice of model, one can derive economic-political rules of action that are diametrically opposite. As an example, one may think of the perpetual argument between the “monetarists”, who call for strict budgetary discipline even in times of recession, in contrast to the “Keynesians”, who hold in high esteem the “deficit spending” of their idol John Maynard Keynes and therefore demand national investment programs in times of recession. Needless to say, both camps refer to models!

But even among scientists, consensus does not necessarily exist or, at best, it is an arduous process. This is exemplified by the on-going discussion about climate change and global warming.

Game theory, which we will cover in more detail later, is another example that illustrates the difficulty in finding the right model. John von Neumann's models for 2-person zero-sum games which embrace conservative min-max

strategies—in short: always play so that the worst case loss is minimized assuming optimal performance by the opponent—may be an appropriate model to use for the head of a family on his one-time meander to the casino. Without doubt, however, these strategies do not suffice for the serious gambler.

Where do we model nowadays? A few important applications have already been mentioned, but no list can altogether be complete. However, one should be aware how commonplace the use of models, and thus modelling, have nowadays become:

- In *astrophysics*, one works to explain the origin and evolution of the universe as well as the life cycles of stars and galaxies.
- In *geophysics*, researchers want to understand processes which eventually lead to earthquakes.
- A central topic in *plasma physics* is fusion.
- The analysis of the spatial structure and therefore the functionality of proteins is a focus in *protein research*.
- *Theoretical chemistry* investigates the causes for certain material behavior on the atomic level.
- *Drug design* is concerned with the systematic design of agents having an exactly specified functionality.
- Also *medicine* increasingly utilizes models, e.g., in the research of aneurysms or the optimization of implants.
- The discussions in *climate research*, which are driven by models, have had a high audience appeal—ranging from global warming, holes in the ozone layer or the future of the gulf stream.
- On a shorter term, but no less current, is *weather forecasting*, which relies on a mix of computations and measurements.
- The *automobile industry* yields a whole wealth of examples: Whether one considers crash tests (structural mechanics), deep drawing (structure optimization), aerodynamics or air conditioning (fluid dynamics), sound emission (aeroacoustics), fuel injection (combustion), vehicle dynamics (optimal control) or sensor and actuator technologies (coupled systems, micro-electro-mechanical systems)—models are always involved.
- Also the *semiconductor industry* depends completely on models and simulations—examples are given by device simulation (transistors etc.), process simulation (production of highly purified crystals), circuit simulation as well as questions regarding the optimization of chip layout.
- Furthermore, several models have been and are being developed in *economics*—for the business cycle, for the economic and fiscal policy and for pricing mechanisms. The fact that the experts in the government's advisory boards are not necessarily always of one opinion, illustrates how far one is still from a consensus model.
- For *banks and insurance companies*, models are highly important for the assessment (i.e., pricing) of financial derivates such as options.

- *Traffic technology* requires quite diverse models—for example for the formation, dissolution and avoidance of traffic congestion, for the long term planning of traffic routes and for evacuation scenarios.
- *Providers* such as energy companies require load models in order to design their networks as fail-safe as possible.
- *Shipping companies* rely on model-based fleet management.
- Population models are used by *city planners, governments* (thinking of China's “one-child-policy”) as well as by *epidemiologists* (how do famines spread?).
- Without models, the statements of *pollsters* would resemble the prophecies of fortune tellers.
- And finally, what would *computer games* and *computer movies* be without illumination or animation models?

One sees: There are quite diverse application areas for modeling that exist with both “hard” (i.e., mathematical-formula heavy) and “soft” (i.e., more descriptive) models. And by now one suspects that these require a corresponding wide range of tools from mathematics and computer science. As a result, the following central questions arise in the context of modeling:

1. How does one obtain a suitable model?
2. Which descriptive tools will be used?
3. How does one subsequently assess the quality of the derived models?

We will now direct our attention to these questions.

1.2.2 *Derivation of Models*

We will begin with the task of deriving a model—which often appears to be a miracle to the newcomer. This derivation typically occurs in several steps.

At first, one has to determine what exactly should be modeled and thereafter simulated. Whoever is inclined to answer “well, the weather”, is not thinking on a grand enough scale: the weather in what time frame, the weather in what region and in what spatial resolution, resp., the weather in what precision? Several additional examples may serve as illustrations: Is one interested in a rough estimate of the efficiency of a car catalyst or rather in the detailed interior reaction processes, i.e., does the domain have to be resolved or not? Is one interested in the population growth in Kairo, in Egypt, or in all of Africa? Which resolution is therefore suitable or adequate? Should the throughput of a computer network or the average throughput time of a packet of data be determined by simulation, i.e., should data sets be considered separately as discrete entities, or is it sufficient to take averaged data flow quantities?

Next, one needs to determine which quantities play a *qualitative* role and how significant is their *quantitative* impact. We will illustrate this with a few examples: The optimal trajectory of the space shuttle is influenced by the gravitation of the

moon, the gravitation of Pluto and the gravitation of this book, but not all of these are relevant for the computation of the trajectory. The development of the Dow Jones index may depend on statements of the director of the American central bank, statements of the authors of this book as well as the bankruptcy of the Sultanate of Brunei. However, investors do not have to consider these statements on an equal footing. One sees that we can sometimes grasp causality and relevance intuitively, but in general (in particular for highly complex systems) these are all but obvious. Oftentimes, there are—despite the relevant expertise and ample material data—only hypotheses. And typically these early decisions significantly influence the later simulation results.

Once the set of relevant quantities has been determined, we have to direct our attention to the network of relationships among the model parameters that have been deemed to be important. Once again there is the qualitative aspect (logical dependencies of the type “if, then” or signs of derivatives) and the quantitative aspect (particular factors, magnitudes of derivatives). Typically, these networks of relationships are complex and multi-layered: In general, the CPU performance of a computer has a strong influence on the computational time of a job; however, in the case of strong thrashing or a low cache hit rate, it only plays a minor role. Such fluctuating dependencies have to be incorporated into the model as well.

Now, what is a suitable instrument to best formalize the interactions and dependencies previously identified? Mathematics and computer science provide a wealth of descriptive tools and instruments:

- *algebraic equalities or inequalities* to describe laws ($E = mc^2$) or constraints ($w^T x$);
- systems of *ordinary differential equations* (differential equations with only a single independent variable, typically the time t), for example for the description of growth behavior ($\dot{y}(t) = y(t)$);
- systems of *partial differential equations* (i.e., differential equations with several independent variables, such as different spatial directions or space and time), for example for the description of deformation of a clamped diaphragm under load ($\Delta u = f$) or for the description of wave propagation ($u_t = u_{xx}$);
- *automata and state transition diagrams*, for example for the modeling of queues (filling levels as states, arrival and service end as transitions, resp.), of text recognition (previous text structures as the states and the new symbols as the transitions) or of growth processes with cellular automata (overall occupancy situation as the states with rule-based transitions);
- *graphs*, for example for the modeling of round trip problems (problem of the traveling salesman with places as vertices and paths as edges), of ordering problems (partial jobs as the vertices, dependencies in time through the edges), of computing systems (components as the vertices, connecting paths as the edges) or of processes (data flows, workflows);
- *probability distributions*, to describe arrival processes in a queue, error terms as well as white noise or the approval of the government’s policy on the unemployment rate;

- *rule-based systems* or *fuzzy logic* for the modeling of control problems;
- *neural networks* for modeling learning;
- *language concepts* such as UML, in order to model complex software systems;
- *algebraic structures*, for example groups in quantum mechanics or finite fields in cryptology.

In this book we will on occasion see that finding a “best description” is quite a lofty goal, and that most cases do not require *the* model but rather *a suitable* model.

Finally, a perceptively trivial question whose answer is nevertheless important for goal-oriented modeling and simulation:: What is the concrete task? Shall we find *an arbitrary* solution of the model; shall we find *the only* solution of the model; shall we find *a particular* solution (which is optimal with respect to a certain criterion or which satisfies certain boundary conditions or constraints, respectively); shall we find a critical region (e.g., a bottleneck); or shall we show that at least one or even several solutions exist?

Once a model has been identified, it needs to be assessed. This will be the topic of the following section.

1.2.3 Analysis of Models

The analysis and assessment of models deals with the derivation of statements in terms of their manageability and usefulness, resp.

Here, the question of *solvability* takes center stage: Does a certain model have one or several solutions, or none?

In population dynamics, for example, one is interested in whether or not a certain model has a stationary limit state and whether or not it actually attains its limit value. In ordering problems, i.e., when, for example, a set of tasks has to be completed on a set of machines, the question arises whether or not the precedence graph, which describes possible constraints in the ordering of tasks, contains any cycles. In minimization problems, it is crucial to determine whether the target function actually assumes a minimum, or possibly only contains saddle points (minimal with respect to a subset of directions while maximal with respect to the remaining directions).

In the case of solvability, the subsequent question concerns the *uniqueness* of solutions: Is there exactly one solution, is there exactly one global minimum? Is there a stable limit state or rather oscillations, as we will later see in predator-prey-scenarios in population dynamics, or different pseudo-stable states, among which the solution jumps back and forth (such as in the form of spatial configurations or convolutions in proteins)? In the case of several solutions, are these all of equal weighting, or are there preferred solutions?

A third aspect is likely to be less obvious: Does the solution depend *continuously* on the *input data* (initial values, boundary values, material parameters, constraints, etc.), or could rather small changes in the input lead to a completely different

behavior in the solution? The idea of continuous dependency corresponds to a notion of the *sensitivity* or *conditioning* of a problem, resp.

In 1923, Hadamard chose the following three aspects as a launching point for his definition of a *well-posed problem*: A problem is therefore called well-posed if a solution exists, is unique, and furthermore depends continuously on all input data. However, Tikhonov and John have subsequently shown that this definition is quite restrictive—unfortunately, problems are mostly ill-posed. Prime examples for ill-posed problems which illustrate this notion are the so-called *inverse problems*. With an inverse problem, the result is essentially given and one is looking for the initial configuration: How should a pressing tool be configured so that a metal sheet is worked to produce the desired result? How much does carbon dioxide emission have to be reduced in the next ten years in order to avoid certain undesirable developments? In politics, what must be done today in order to reduce the percentage of unemployed persons below 10 % within three years? How do the components of a computer network have to be configured in order to guarantee a certain minimum throughput? Even if the corresponding forward problem is continuous (a marginal change in the corporate tax rate will hardly lead to a big jump in unemployment), the continuity typically no longer holds in the opposite direction: It cannot be expected that a slightly smaller unemployment rate can simply be obtained through a slightly reduced tax rate—even if one would like to believe this!

As one may already anticipate, such inverse problems are no rarity in practice—oftentimes a goal to be reached is given, and one is looking for a suitable way to do so. Even if this is an ill-posed problem, there are possible ways to “rescue” the model. A first option is the (meaningful) trial and adjustment, i.e., the solution of a sequence of forward problems. Here, the skill consists of reaching convergence quickly. An alternative approach is the so-called *regularization*. Here, one solves a related, well-posed (regularized) problem instead of the original problem. A generally helpful trick: If the problem is unpleasant, change it slightly!

We still have to discuss a fourth aspect of model assessment—one that is in fact often neglected by the pure modelers or at best treated as a distant relative: How difficult will it be to continue the processing of the model (i.e., the simulation)? In fact, the modeling is not done as an end in itself but rather as a means to perform simulations. This raises a couple of additional questions: Is the availability and quality of the required input data sufficient? In the end, what is the purpose of an ever so elegant model if I do not have access to the input data? For the solution of the model, do algorithms exist, and if so, what are their computational and storage complexities? Given this, is a solution realistic, in particular when keeping in mind the real-time requirements? After all, tomorrow’s weather forecast must be completed before tomorrow. Do we have to anticipate or expect fundamental problems during the solution process (ill-conditioning, chaotic behavior)? Is the model competitive, or do there exist models with possibly a better price-performance ratio? And finally, how involved is the expected implementation effort? These all are questions that by far exceed the pure modeling effort but must nonetheless be considered at this stage.

If all these questions could be addressed satisfactorily, one could then approach the simulation stage. Before doing this, however, we will attempt to bring some structure in the midst of the flood of existing and conceivable models.

1.2.4 Classification of Models

From the multitude of possibilities for classification we will take a closer look at two: discrete vs. continuous models as well as deterministic vs. stochastic models.

In modeling, *discrete models* exploit discrete or combinatorial descriptions (binary or integer quantities, state transitions in graphs or automata), while in contrast *continuous models* are based on real-valued or continuous descriptions (real numbers, physical quantities, algebraic equations, differential equations). Obviously, discrete models are naturally used to model discrete phenomena, whereas continuous models are employed for continuous phenomena. However, this is by no means mandatory as demonstrated by the example of traffic simulation which will also be studied later in this book. Here, the traffic flow through a city can be modeled discretely (single cars as entities which wait at lights, etc.) as well as continuously (densities, flows through channels). The approach deemed more suitable depends on the actual problem setting.

Examples of *deterministic models* include systems of classical differential equations which manage without random components. Ever more frequently, however, the systems contain probabilistic components—whether to integrate error terms (noise), or to account for uncertainties, or to explicitly build in randomness (stochastic processes). Once again, there is no mandatory correlation between the character of the process being modeled and the instrument employed. Non-deterministic experiments such as the roll of a die represent a probabilistic reality and are modeled as such; Crashtests are strictly causal-deterministic and are generally modeled deterministically. The weather forecast becomes more interesting: In a sense, everything happens strictly deterministically, obeying the laws of thermodynamics and fluid dynamics. However, several turbulence models contain stochastic components. Finally, the (hopefully) deterministic incoming order of jobs for a printer is mostly modeled via stochastic processes—from the point of view of the printer, the jobs arrive randomly, and furthermore, at least for system design, one is rather interested in average quantities (means) and not in the individual fates of printing jobs.

1.2.5 Scales

One idea should be quickly dismissed, namely the one involving the “correct” model. Modeling is rather a question of consideration of complexity, or cost and accuracy. The more details and single effects one integrates in a model, the higher the precision one naturally expects for the attainable results—however, at the

expense of increasing simulation cost. Phenomena always take place on certain *scales*—spatial (from nanometer up to the light year) and temporal (from the femtosecond up to billions of years), and models and simulations themselves are always based on certain scales. In principle, each molecule in the air makes a contribution to the weather—at the same time it would be crazy to consider all molecules individually when forecasting the weather. But one cannot neglect spatial resolution completely: A statement of the kind “tomorrow will be nice in Europe” is usually not very helpful. Thus the question pertaining to the level of detail (spatial or temporal) or resolution arises, i.e., which scales are appropriate—first in view of the desired accuracy of the result and second in view of the required solution cost.

A few examples may illustrate this. Let us begin with true high-technology. The heating of water in a cylindrically formed pot on a stove can be modeled and simulated in one spatial dimension (temperature as a function of time and height in the pot—after all, the pot is cylindrical and its content—water—is homogenous), in two spatial dimensions (temperature as a function of time, the height in the pot and the radial distance to the middle of the pot—after all, the room air is cooling the pot from the outside) or even in three spatial dimensions (additional dependence of the temperature on the circular angle—after all, no stove heats in perfect rotational symmetry); what is the appropriate approach? Or in population dynamics: Typically, the development of a species is described as a purely time-dependent process. However, this could not yield a reasonable description of the development of a population such as in the USA in the middle of the nineteenth century when the strong “go west!” drive prevailed for the migration of settlers.

The simulation of circuits provides another example. For many years, this simulation was performed as purely time-dependent (system simulators based upon Kirchhoff’s circuit laws). The increase in integration density leads to a growing occurrence of parasitic effects (current through a conductor induces current through a nearby other conductor) which are local phenomena and require a spatial model component. Finally the catalytic converter in our cars: Do I really need to resolve in detail the geometry of the catalytic converter for the computation of macroscopic quantities such as the degree of efficiency?

The previous question leads us to another aspect, the interplay of scales. Often, we have to deal with a so-called “multiscale property”. In this case, the scales cannot be separated without an unacceptable loss of accuracy because of shared interdependency. A classical example is given by turbulent flows. Phenomenological to turbulent flows, one has to deal with strong, erratic vortices of varying magnitudes—from tiny to very large. The flow is unsteady and inherently three-dimensional. Here, a strong energy transport takes place in all directions and between the scales. Depending on the viscosity of the fluid, one needs to compute the tiniest vortices even in larger domains in order to avoid incorrect results. The dilemma, therefore, is that for reasons of efficiency, one cannot resolve all that is needed to be resolved for reasons of accuracy. A remedy is found in turbulence models: They try to pack the fine-scale influence into suitable parameters of the large scale—through averaging (with respect to space or time) or through

Table 1.1 A hierarchy of possible simulations on different scales

Problem setting	Level of consideration	Possible model
Population increase globally	Countries/regions	Population dynamics
Population increase locally	Individuals	Population dynamics
Human physiology	Circuits/organs	System simulator
Blood circulation	Pump/canals/valves	Network simulator
Blood stream in the heart	Blood cells	Continuum mechanics
Cellular transport processes	Macromolecules	Continuum mechanics
Function of macromolecules	Atoms	Molecular dynamics
Atomic processes	Electrons, ...	Quantum mechanics

homogenization. Naturally, such multiscale phenomena set particular requirements to the models and simulations.

In view of the wide spectrum of relevant scales, one often encounters entire model hierarchies. As an illustration, we consider such a hierarchy centered around humans: Each level can represent certain things, but not others, and the models and simulation techniques differ from level to level (Table 1.1).

1.3 Introduction to Simulation

1.3.1 General Remarks

Our aim is not to derive and employ models just for the description of a circumstance, but for the subsequent simulation based on these models. To this end, the models have to be solved in concrete scenarios—for example differential equations plus initial and boundary conditions. This can be done through various methods.

An *analytic solution* not only includes existence and uniqueness proofs, but also the formal analytic construction of the solution—using “paper and pencil”, as it is referred to in mathematics. This is insofar the preferred case since no further simplifications or approximations are required. However, this approach works almost exclusively only in very simple (and thus mostly hopelessly unrealistic) special cases. For example, one can directly write down the solution $y = c e^t$ of the simplistic growth law $\dot{y}(t) = y(t)$ without any magic. A little less obvious, but still no trick, is the direct solution of the one-dimensional heat equation $u_{xx}(x, t) = u_t(x, t)$; here, a so-called separation approach yields $u(x, t) = \sin(cx)e^{-c^2 t}$. Finally, in mini-graphs, one can detect a shortest path through a simple exhaustive search. But what alternatives exist when an analytical solution is not feasible? Irrelevant here is whether this is due to fundamental reasons or due to the limited capabilities of the person working the problem.

The *heuristic solution approach* offers a first alternative in which, beginning with plausibility arguments, one uses certain strategies to get closer to the unknown

solution. Such heuristics are wide-spread primarily for problems in combinatorical or discrete simulation and optimization (e.g., greedy heuristics which always choose the best local alternative). In the knapsack problem, for example, one packs the item with the respective best weight-value-relation into the knapsack until nothing else fits. This does not necessarily lead to the best solution, and even if it does, it may take excessively long. But such a procedure, however, is still good as a heuristic.

In the *direct-numerical approach*, a numerical algorithm provides the exact solution modulo round-off error. The simplex algorithm for problems in linear optimization of the kind “solve $\max_x c^T x$ under the constraint $Ax \leq b$ ” is an example of this approach. For the *approximate-numerical approach*, however, one refers to an approximation method in order to approximate the solution of the model as accurately as possible. This task splits into two parts: first, the *discretization* of the continuous problem, and second, the *solution* of the discrete problem. Both parts are concerned with the question of convergence. The discretization should be of the type such that an increase in effort (i.e., an increase in resolution) will lead to asymptotically better approximations, and the (mostly iterative) solution technique for the discretized problem should first of all converge as well as converge rapidly to its solution.

The approximate-numerical approach is certainly the most important one for problems in numerical simulation; we will encounter it repeatedly in the following chapters.

1.3.2 Assessment

Of central significance in a simulation is the assessment of the computed results. The goal of *validation* is to determine whether we have used the correct (or rather, a suitable) model (“Do we solve relevant equations?”). By contrast, *verification*, takes a look at the algorithm and software program with the purpose to determine whether the given model has been solved correctly (“Do we solve the given equations correctly?”). Even in the case of two affirmative “Yes!” answers, the examination of aspects concerning the accuracy of the result as well as the invested effort is still remaining.

There are several possibilities for validating the computed simulation results. The classical procedure is the *comparison with experimental tests*—whether these are 1:1-experiments, as for example in crash tests, or scaled laboratory experiments, for example tests in a wind tunnel with downsized prototypes. Sometimes, however, this approach is prohibited for feasibility reasons or the required effort. But even when experiments can be performed one should use caution: First of all, it is easy for small differences to arise between the simulated and the experimental scenarios; secondly, one has to be very cautious with respect to the scaling of quantities (it is possible that certain effects do not appear on small scales); and thirdly, there may occur sporadic and systematic mistakes in measuring—computers and their operators do not have a monopoly on bugs!

A-posteriori observations provide an additional (and in general very inexpensive) possibility for validation—true to the motto that “one is always smarter afterwards”. *Reality tests* compare the predicted with the actual result; one may think of the weather, the stock market or military scenarios. *Satisfaction tests* determine whether the desired result has materialized to a sufficient degree. Examples for applications are systems for traffic control as well as illumination and animation models in computer graphics.

In contrast, the *plausibility tests* remain on a purely theoretical level, as frequently encountered in physics. Here, one checks whether the simulation results are in contradiction to other, previously verified theories. Naturally, one must not be too conservative—possibly, the common doctrine errs and the simulator is correct!

Finally, there exists the option to perform a *model comparison*, i.e., to compare the results of simulations which are based on different models.

No matter how one proceeds—one needs to use caution before one draws conclusions from validations. There are various sources for errors; pears literally wait to be compared to apples, and Muenchhausen has been known to pull himself out of the swamp by his own hair ...

The topic of verification leads to convergence proofs, etc. for the employed algorithms on one hand and to correctness proofs for the designed programs on the other hand. While the former are well established and are known to be a favorite pastime for numerical analysts, the later are still in their infancy. It is not so much the case that computer science has not achieved anything in this respect. It is rather the case that the simulation business—quite opposite to other software-intensive fields—is positioned extremely shirt-sleeved (not systematic in its approach): Here, one usually programs but hardly ever develops software. The pain threshhold seems to have been reached only recently, and requests for a formal framework (and thus for better possibilities for verification) become louder.

Even the aspect of *accuracy* is more complex than it appears at first sight. The first coming to mind is accuracy with respect to the quality of input data. If the input data is available in the form of measured data with an accuracy of three decimal places, then one cannot expect the result to be accurate to eight decimal places. In addition, one needs to keep an eye on the accuracy in relation to the problem—which can at times be problematic. In many cases, a model which produces errors below one per cent is considered completely sufficient. But in an election poll, for example, being half a percent off the mark can turn everything upside down—and thus render the modeling and simulation completely useless! Another factor is the need for security: Can one live with statements that reflect averaged values, or is it necessary to reflect a guaranteed worst-case-bound?

And finally the *cost question*—what effort (wrt. time for implementation, memory, computation or response time) was invested to reach the simulation result? In this context, it is important to neither consider the obtained benefit (e.g., the accuracy of the result) nor the invested effort individually, but rather in relation to each other. Basically, it is neither the best nor the cheapest car that one wants to buy, but rather the one with the best price-performance ratio.

Chapter 2

Required Tools in Short

In this chapter, we will give an itemized overview of the topics in mathematics and computer science which we will reference repeatedly in subsequent parts of this book. The selection of topics for this chapter is guided by the simple rule that everything occurring multiple times is introduced only once in this central location, whereas tools that are required only for individual application scenarios will be covered in that particular part. Since our book is primarily directed toward students studying computer science, engineering, or natural sciences who are upper division bachelor students, we assume that most topics of this chapter have been studied in their respective introductory courses in mathematics. On the other hand, the experience with courses on modeling and simulation has shown that a certain “warm-up” period in order to better adjust preliminary knowledge can do no harm—especially in view of the wide mix in academic backgrounds oftentimes observed (and intended so) in our audience.

This is the rationale behind the subsequent brief overview that follows. Those already familiar with the introduced notions and concepts may proceed quickly and confidently; those, however, who encounter unknown or somewhat forgotten topics should consult the cited references or find other resources to close the respective gaps before they begin their study of the various models and methods.

The following four sections will address elementary and discrete, continuous, stochastic and statistical as well as numerical aspects. Not everything will reappear explicitly at a later place—however, these contents are of significant importance for the more intensive treatment of one or more modeling or simulation topics and frequently appear implicitly. In order to illustrate these relations, the last section of this chapter will provide an overview of the network of relationships between tools and applications. To facilitate the orientation, each subsequent chapter begins with a reference to the relevant parts from this current chapter.

2.1 Elementary and Discrete Topics

We begin with a few topics taken from elementary and discrete mathematics. Two books (or their respective chapters) we recommend for further references and reading are “Concrete mathematics: A foundation for computer science” by Graham, Knuth and Patashnik [28] and “Discrete Mathematics with Applications” by Susanna Epp [17].

Sets

One of the most fundamental notions of mathematics is the one of a *set*. Sets consist of individual *elements*—one says that “the set M contains the element x ” and writes $x \in M$. If N contains only elements which are also contained in M , then N is called a *subset* of M ($N \subseteq M$). The set $\mathcal{P}(M)$ of all subsets of M is the *power set*. The *empty set* \emptyset contains no elements. The *cardinality* $|M|$ of a finite set is the number of its elements; however, a set could also be infinite. The essential operations on sets are the *union* $N \cup M$, the *intersection* $N \cap M$, the *difference* $N \setminus M$, the *complement* \bar{M} as well as the *Cartesian product* $M \times N$, the set of all pairs (m, n) with $m \in M$ and $n \in N$.

At this junction, let us recall a few important *topological* properties of sets: There exist *open* sets, *closed* sets and *compact* sets (colloquially, closed and bounded).

Numbers

Important examples for sets are sets of numbers, in particular the *binary numbers* or *Boolean values* $\{0, 1\}$, the *natural numbers* \mathbb{N} , the *integers* \mathbb{Z} , the *rational numbers* \mathbb{Q} as well as the uncountable sets \mathbb{R} (*real numbers*) and \mathbb{C} (*complex numbers*).

Symbols

∞ denotes infinity, \forall and \exists denote the quantifiers “for all” and “there exist”, respectively, $\delta_{i,j}$ denotes the *Kronecker delta symbol* (0 for $i \neq j$, 1 else). We will frequently encounter the so-called *Landau symbol* or *\mathcal{O} notation*; it describes the asymptotic behavior—in case of $\mathcal{O}(N^k)$ typically the effort for $N \rightarrow \infty$, and in case of $\mathcal{O}(h^l)$ in general the error as $h \rightarrow 0$.

Relations and Mappings

A *relation* R between two sets N and M is a subset of the Cartesian product $N \times M$; one writes aRb or $(a, b) \in R$. Important properties of relations are *symmetry* ($aRb \Rightarrow bRa$), *transitivity* ($aRb, bRc \Rightarrow aRc$) as well as *reflexivity* ($aRa \forall a$).

Relations will be treated in more detail in the chapter on group decisions. Of central importance are also the notions of a *mapping* f from a set M into a set N (one writes $f : M \rightarrow N$ or $f : m \mapsto n$ for $m \in M, n \in N$ or $n = f(m)$, resp., and refers to the *image* n as well as the *inverse image* m) as well as the notion of a *function* as a unique mapping. Some important properties of mappings that are possible are *injectivity* ($f(m_1) = f(m_2) \Rightarrow m_1 = m_2$), *surjectivity* ($\forall n \in N \exists m \in M : n = f(m)$) as well as *bijectivity* (injective and surjective). For any bijective mapping, there exists an *inverse mapping* that is also bijective.

Graphs

Graph theory is a discrete mathematical modeling tool of great significance. A graph is a tuple (V, E) , consisting of a finite set V , the *nodes*, and a relation E on V , the *edges*. Edges can be *directed* or *undirected*. The *neighborhood* of a vertex v is the subset of nodes that are connected to v by an edge; their number determines the *degree* of v . For a *connected* graph, any two arbitrary different nodes are connected through a sequence of edges—a *path*. In the disconnected case, the graph has several *connected components*. A connected graph with directed edges that does not contain any *cycles* (non empty paths with equal start and end node) is called a *DAG* for “directed acyclic graph”. *Trees* are a particular type of connected graphs that are acyclic when considered as undirected graphs. One often needs to completely traverse graphs and in particular trees or their respective nodes—the most important strategies are the *breadth-first search* and the *depth-first search*. A *Hamiltonian cycle* is a cycle which visits each node exactly once; a *Eulerian tour* is a path that contains each edge exactly once and ends again at the starting node. A *node coloring* assigns a color to each node so that neighboring nodes never have the same color; the minimum number of required colors is called the *chromatic number* of the graph.

2.2 Continuous Aspects

The continuous world is naturally of central importance to numerical simulation. The basics are provided by linear algebra and analysis. A multitude of textbooks are available for either one. As examples, we mention the books “Introduction to Linear Algebra” by Gilbert Strang [52] and “Calculus” by James Stewart [51].

The two accelerated courses in linear algebra and analysis are succeeded by a brief summary explaining why these two fields are so important for modeling as well as simulation.

2.2.1 Linear Algebra

We will begin by assembling some of the essential terms from linear algebra.

Vector Spaces

The structure central to linear algebra is the *vector space*, an additive group over a set (the *vectors*), in which multiplication by scalars is defined as well. Although this was inspired by the geometric interpretation of a vector—it is not necessary. A set of vectors $\{v_1, \dots, v_n\}$ can be combined linearly: Each *linear combination* $\sum_{i=1}^n \alpha_i v_i$ with real-valued factors α_i is also an element of the vector space; the set of all vectors that can be represented by such a linear combination is called the *span* of v_1, \dots, v_n . A set of vectors is called *linearly independent* if the only linear combination that produces the zero vector is the case in which $\alpha_i = 0 \forall i$; otherwise, the set is called *linearly dependent*. A *basis* of a vector space is a set of linearly independent vectors in which every vector in the vector space can be represented uniquely as a linear combination of these vectors. The number of vectors making up the basis—finite or infinite—is called the *dimension* of the vector space.

Linear Mappings

A (*vector space*) *homomorphism* f is a linear mapping from a vector space V into a vector space W , i.e., $f : V \rightarrow W$ with $f(v_1 + v_2) = f(v_1) + f(v_2)$ and $f(\alpha v) = \alpha f(v) \forall v \in V$ as well as $\forall \alpha \in \mathbb{R}$. The set $f(V)$ is called the *image* of the homomorphism and the set of all vectors $v \in V$ with $f(v) = 0$ is called the *kernel* of f . An important means to represent homomorphisms of the type $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are *matrices* $A \in \mathbb{R}^{m,n}$; the linear mapping corresponds to the assignment $x \mapsto Ax$. The matrix of an injective mapping consists of linearly independent columns while the matrix of a surjective mapping consists of linearly independent rows; bijectivity therefore necessitates $m = n$ as well as n linearly independent columns and rows. The number of linearly independent rows of A is called the *rank* of the matrix. The *transpose* A^T of a matrix is obtained by reflecting the entries with respect to the diagonal. If in addition to performing a transpose one forms the complex conjugates of all matrix entries, then one obtains the *Hermitian conjugate* A^H . Thus, the matrix-vector-product Ax means the application of the linear mapping given by A to the vector x ; solving a *linear system of equations* $Ax = b$ with unknown x corresponds to looking for the inverse image of b .

Linear Systems of Equations

We restrict our attention to the case of a square coefficient matrix $A \in \mathbb{R}^{n,n}$. The system $Ax = b$ has a unique solution for every b if and only if A has *full rank* (i.e., rank n). The mapping is then invertible, and the matrix A is *non-singular* or *invertible*, i.e., there exists a matrix A^{-1} in which the product $A^{-1}A$ is the identity I . An equivalent characterization of a full rank matrix is for its so-called *determinant* of A , $\det(A)$, to be non-zero.

Eigenvalues and Eigenvectors

If it holds that $Ax = \lambda x$ for λ a scalar and $x \neq 0$, then λ is called an *eigenvalue* of A and x is called an *eigenvector* of A . The eigenvalues are the n (possibly complex) roots of the *characteristic polynomial* $\det(A - \lambda I) = 0$; their entirety is called the *spectrum* of A . This spectrum characterizes the essential properties of the matrix—we will study the eigenvalues of matrices at various places in subsequent chapters in order to obtain certain statements about models or solution methods.

Scalar Products and Norms

An important mapping $V \times V \rightarrow \mathbb{R}$ from the vector space $V = \mathbb{R}^n$ to \mathbb{R} is the *scalar product* $x^T y := \sum_i x_i y_i$. Furthermore, we will use *vector norms* $\|\cdot\|$ to measure or estimate errors, i.e., a positive, homogeneous mapping from V to \mathbb{R} , which also satisfies the triangle inequality $\|x + y\| \leq \|x\| + \|y\|$. Examples are the *Euklidian norm* $\|x\|_2 := \sqrt{\sum_i x_i^2}$, the *maximum norm* $\|x\|_\infty := \max_i |x_i|$ as well as the *sum norm* $\|x\|_1 := \sum_i |x_i|$. In the case of the scalar product and the Euklidian norm, there holds the *Cauchy-Schwarz inequality* $|x^T y| \leq \|x\|_2 \cdot \|y\|_2$. Sometimes, one also requires norms for matrices—associated *matrix norms* can be induced from vector norms by means of $\|A\| := \max_{\|x\|=1} \|Ax\|$. Furthermore we recall the concept of *orthogonality*: Two vectors are orthogonal if $x^T y = 0$.

Classes of Matrices

A number of matrix classes deserves special attention. $A \in \mathbb{R}^{n,n}$ is called *symmetric* or *skew symmetric*, if $A^T = A$ or $A = -A^T$, respectively. For $A \in \mathbb{C}^{n,n}$ and $A = A^H$, A is called *Hermitian*. Real symmetric or complex Hermitian square matrices are also called *positive definite* if $x^T Ax > 0$ or $x^H Ax > 0$, respectively, $\forall x \neq 0$. For *positive semidefinite* matrices, we have that $x^T Ax$ or, respectively $x^H Ax$, is nonnegative $\forall x$. *Orthogonal matrices* satisfy $A^{-1} = A^T$, while *unitary matrices* satisfy $A^{-1} = A^H$. Also of interest are *normal matrices* ($A^T A = AA^T$ or, respectively $A^H A = AA^H$), since these are the only matrices in which there exists a basis of pairwise orthogonal eigenvectors.

2.2.2 Analysis

Now we will turn our attention to analysis since material under this subject heading will also accompany us through this book.

Continuity

Of the various common *definitions of continuity* we restrict our attention to the one most widely used: A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is called *continuous* at x_0 , if for every $\varepsilon > 0$ there exists a $\delta > 0$ such that $\forall \|x - x_0\| < \delta$ it holds that $|f(x) - f(x_0)| < \varepsilon$. If this holds not only at a point x_0 but over the entire domain of f , then one speaks of *global continuity*.

Limits

Central to this subject of study is the *convergence behavior* of a function f in the case of an arbitrary approach to a point (which may also be infinite). One then speaks of *limits* of functions and writes $\lim_{x \rightarrow 0} f(x) = \dots$ or $\lim_{x \rightarrow \infty} f(x) = \dots$. The limit can also be infinite: $\lim f(x) = \pm\infty$.

Sequences and Series

A *sequence* is understood to be a function f defined on \mathbb{N} . Using $a_n := f(n)$, one writes a sequence in short as (a_n) . Sequences can be *bounded* or *monotonic*, and as with functions we are interested in the *convergence behavior* for sequences as $n \rightarrow \infty$ or the existence of a limit $\lim_{n \rightarrow \infty} a_n$, resp. Infinite *series* $\sum_{n=0}^{\infty} a_n$ are sequences of *partial sums* $(\sum_{n=0}^N a_n)$, where once again the central topic is the question of convergence for case $N \rightarrow \infty$.

Differentiability

A single variable function f is called *differentiable* (at a point x_0 or on an interval or globally), if its derivative exists there. The *derivative* of a function describes the change in behavior of the function values as a result of a change in its argument. It is defined as the limit of *difference quotients* $(f(x) - f(x_0))/(x - x_0)$ as $x \rightarrow x_0$. One writes f' , \dot{f} or df/dx . *Higher derivatives* $f^{(k)}$ are defined analogously. The differential operator is linear: It holds that $(\alpha f + \beta g)' = \alpha f' + \beta g'$ for $\alpha, \beta \in \mathbb{R}$. Furthermore, the product rule is given by $(fg)' = f'g + fg'$, the quotient rule by $(f/g)' = (f'g - fg')/g^2$ and the chain rule by $(f(g(x)))'|_{x=x_0} = f'(g(x_0))g'(x_0)$.

Extrema, Inflection Points and Convexity

Functions can assume global and local *extrema (minima, maxima)*. In optimization one is naturally interested in these points. If the function is differentiable, then a necessary condition for a local extrema is for the first derivative to vanish. There is

an *inflection point* if the second derivative vanishes as well as changes its sign at this point. One speaks of *convexity* if the secant line through any two points on the graph of the function f always lies above the graph.

Classes of Functions

Important classes of special functions are *polynomials* $\sum_{i=0}^n a_i x^i$ as well as *rational functions* (quotients of polynomials). Furthermore we will deal with the *exponential function* e^x , the *logarithm* $\log(x)$ as well as the *trigonometric functions* such as, e.g., $\sin(x)$ and $\cos(x)$. The latter are actually defined by the way of a mathematical model, the *harmonic oscillator* $y^{(2)} + y = 0$.

Integrals

Integration can be thought of as the inverse mapping of differentiation. The *indefinite integral* $\int f(x)dx$ of an integrable function $f : \mathbb{R} \rightarrow \mathbb{R}$ denotes an *antiderivative* F , i.e., $F'(x) = f(x)$; the *definite integral* $\int_a^b f(x)dx$ computes the area under the graph of f from a to b . The notion of integrability is a little more complicated. Differentiation and integration are tied together via the *Fundamental Theorem of Calculus*. While the computation of derivatives may be considered a technical skill, integration is more like an art. Some help comes from techniques such as *integration by parts* or *integration by substitution*.

Local and Global Approximation

The *local approximation of a function using polynomials or series* plays a major role in many areas. In numerical simulation, for example, one employs respective techniques to prove the convergence of discretization schemes. Given an n -times continuously differentiable function f , the n -th *Taylor-polynomial* at a denotes the (unique) polynomial of degree n whose zeroth up to n -th derivatives at a coincide with the respective derivatives of f at a . If f is infinitely differentiable (e.g., the exponential function), then one obtains the *Taylor series*.

The central term with respect to *global approximation* is *uniform convergence*, $\|f_n - f\| \rightarrow 0$, which is a stronger condition when contrasted to *pointwise convergence*, $f_n(x) \rightarrow f(x)$. The *Weierstrass approximation theorem* states that every continuous function on a compact set can be approximated to arbitrary accuracy by a polynomial.

Periodic Functions and Fourier Series

For *periodic functions* $f(x)$ with period 2π , it holds that $f(x + 2\pi) = f(x)$. Periodic functions (under certain conditions on smoothness) can be approximated elegantly through *trigonometric polynomials*, i.e., weighted sums of terms comprised of the type $\cos(kx)$ and $\sin(kx)$, resp., each term allowing for a different *frequency* k . A noteworthy example are transmitted signals which can be approximated as an overlay of a finite number of pure sine and cosine oscillations. In this context one speaks of *Fourier polynomials* and *Fourier series*.

Differential Equations

Differential equations are equations that relate functions and their derivatives and use this relation to identify certain functions—the solutions. One distinguishes *ordinary* and *partial* differential equations. In the former, there appears only one independent variable (typically in time t —in which one typically uses \dot{y} , \ddot{y} to denote the derivatives instead of y' , y'' etc.), in the later there are several independent variables (different spatial coordinates or space and time). The *order* of a differential equation is determined by the highest derivative present. A differential equation or a system of differential equations typically describes the underlying physics; additional conditions in the form of *initial conditions* or *initial values*, respectively, or *boundary conditions* or *boundary values*, resp., are necessary for a specific (and uniquely solvable) scenario. Accordingly, a model then can be formulated as an *initial value problem* or as a *boundary value problem*. In the case of partial differential equations (in short: PDEs) we require the use of differential calculus of several variables; in Sect. 2.4 on numerics we will briefly touch on PDEs. Here, we will provide some essential basics of the analysis of ordinary differential equations (in short: ODEs); ODEs will also reappear in Sect. 2.4.

ODEs can only be solved analytically in simple cases—which is one of the reasons numerics is so significant in modeling and simulation. Sometimes, for example for an ODE of the type $\dot{y}(t) = y(y)$, the solution is obvious. Sometimes, for example for ODEs of the type $y'(x) = g(x) \cdot h(y)$, techniques such as the *separation of variables* yield the solution. Here, terms involving the independent variable (x) are algebraically separated from those in the dependent variable (y):

$$\frac{dy}{h(y)} = g(x)dx .$$

Subsequently, both sides are integrated. However, even when all analytic solution attempts fail, in many cases one can still make certain helpful statements with respect to the solvability of the system, the uniqueness of its solutions or the characteristic of the solution(s).

A simple class of ODEs to serve as an example is the *linear ODE of first order* $y'(x) = a(x) \cdot y(x) + b(x)$ with continuous functions a and b , whereas the equation

$y'(x) = a(x) \cdot y(x)$ is called the associated *homogeneous* equation. If $A(x)$ and $u(x)$ denote antiderivatives of $a(x)$ and $b(x)e^{A(x)}$, resp., then $y(x) = c e^{A(x)}$ solves the homogeneous equation and $y(x) = (u(x) + c)e^{A(x)}$ solves the general equation. The initial condition $y(x_0) = y_0$ (an initial condition for an ODE of first order) can be satisfied by way of the constant c —in this simple case one therefore has existence and uniqueness of the solution.

Next, we will take a look at the *linear ODE with constant coefficients*, i.e., an equation of the type

$$y^{(n)} + a_{n-1}y^{(n-1)} + \dots + a_1y' + a_0y = q(x)$$

with constants a_i and a function $q(x)$. Once again, the equation with zero right hand side is called the associated homogeneous equation. Together with a set of n initial values $y(x_0), y'(x_0), \dots, y^{(n-1)}(x_0)$, there is once again exactly one solution $y(x)$ satisfying the ODE and the initial conditions. The solution of the homogeneous equation can be determined by assuming $y(x) = e^{\lambda x}$, while the explicit solution of the inhomogeneous equation succeeds at least in the case of special right hand sides $q(x)$.

A nice example for a linear ODE with constant coefficients is given by the *damped harmonic oscillator* $\ddot{y}(t) + 2d\dot{y}(t) + ky(t) = 0$ with a damping constant $d \geq 0$ and an elasticity constant $k > 0$. Depending on the sign of $d^2 - k$, one has weak, strong, or critical damping (decreasing oscillation, simple oscillation, intensifying oscillation etc.). A right hand side of the form $K \cos \omega t$ leads to a *driven oscillation*.

We point out that differential equations of higher orders can be reduced to systems of differential equations of first order by introducing auxiliary quantities.

A general class of *explicitly given* ODEs of first order (the equation is solved for the highest derivative present) can be defined in the form of

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0.$$

Later, in the section on numerics, we will get to know different approximation methods pertaining to this type of ODEs. The analysis for this type of ODEs also allows for general statements such as the famous theorem of *Picard-Lindelöf*: If f and y satisfy a *Lipschitz condition*

$$\|f(t, y_1) - f(t, y_2)\| \leq L \cdot \|y_1 - y_2\|$$

for the initial value problem $\dot{y}(t) = f(t, y(t))$, $y(a) = y_a$, $t \in [a, b]$, then the existence and uniqueness of the solution are guaranteed. Two special cases with respect to the right hand side f deserve special attention: If f depends only on t , then the problem is in fact no differential equation but rather an integration problem. If, however, f depends only on y , then the ODE is called *autonomous*.

Direction fields and *trajectories* have also proven to be useful for better understanding ODEs. However, since these strongly steer the direction of conversation

to the numerical solution of ODEs, we save this topic for the numerics of ODEs in Sect. 2.4.

Differential Calculus with Several Variables

Let us now consider a subdomain Ω of \mathbb{R}^n to be the domain of f . Here, the *notion of differentiability* becomes more complicated than in one dimension, in that it now is defined through the existence of a linear mapping, the *differential*. “The derivative” no longer exists—one rather needs to specify the direction the arguments $\mathbf{x} \in \mathbb{R}^n$ shall vary.

The *partial derivatives* $f_{x_i} = \partial_i f = \frac{\partial f}{\partial x_i}$ etc., are the canonical *directional derivatives* which describe the changing behavior of f along a coordinate direction. Also prominent is the *normal derivative* $\frac{\partial f}{\partial n}$, which is, for example, given at surfaces of geometrical objects and describes the change of a function defined in space perpendicular to the surface of the objects.

The *gradient* ∇f of a scalar function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ can be interpreted as a vector of partial derivatives or as an operator whose application yields the partial derivatives. The *Jacobi matrix* of a *vector field* $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, a vector-valued function on \mathbb{R}^n such as, for example, velocity in space, contains in row i and column j the derivative of the i -th component of \mathbf{F} with respect to x_j . One can also perform Taylor approximations for $f : \mathbb{R}^n \rightarrow \mathbb{R}$ in higher spatial dimensions; instead of the first derivative, one now uses the gradient, the second derivative is replaced by the *Hessian matrix* of the mixed second partial derivatives f_{x_i, x_j} . In higher dimensions, the analysis is searching as well for extreme values such as minima or maxima. A new notion, however, is that of a *saddle point*: Analogous to a saddle or a pass in the mountains, a saddle point is understood to be a point which represents a minimum with respect to some coordinate directions and a maximum with respect to all the other coordinate directions. We will also reencounter saddle points, e.g., in games. Furthermore, in models of continuum mechanics, one will find three additional operators: The *divergence* of a vector field $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is defined as

$$\operatorname{div} \mathbf{F} := \sum_{i=1}^n \partial_i F_i ,$$

the *rotation* of \mathbf{F} is given by

$$\operatorname{rot} \mathbf{F} := \begin{pmatrix} \partial_2 F_3 - \partial_3 F_2 \\ \partial_3 F_1 - \partial_1 F_3 \\ \partial_1 F_2 - \partial_2 F_1 \end{pmatrix} ,$$

and the *Laplace operator*, applied to a scalar function f , is the divergence of the gradient:

$$\Delta f := \operatorname{div} \nabla f = \sum_{i=1}^n \frac{\partial^2 f}{\partial x_i^2}.$$

Integration in Higher Dimensions

As soon as one departs from the one dimensional setting, integration becomes more complex and complicated. The first case is the so-called *volume integral*, the integral of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ over an integration domain $\Omega \subseteq \mathbb{R}^n$. In the case of a product $\Omega = X \times Y$ with lower dimensional X and Y , one can often utilize (under special prerequisites, of course) *Fubini's theorem*, which clarifies the composition and change of order in integration:

$$\int_{X \times Y} f(\mathbf{x}, \mathbf{y}) d(\mathbf{x}, \mathbf{y}) = \int_X \left(\int_Y f(\mathbf{x}, \mathbf{y}) dy \right) dx = \int_Y \left(\int_X f(\mathbf{x}, \mathbf{y}) dx \right) dy,$$

with a close connection to *Cavalieri's principle*. Another important theorem is the *substitution theorem for multiple variables*. This generalization of integration by substitution permits the transformation of coordinates, once again only under suitable prerequisites:

$$\int_U f(T(\mathbf{x})) \cdot \| \det T'(\mathbf{x}) \| d\mathbf{x} = \int_V f(\mathbf{y}) dy,$$

where $V = T(U)$.

One can also integrate—i.e., sum up—over lower dimensional structures in \mathbb{R}^n . For *line integrals*, a quantity is integrated along a curve in space. *Surface integrals*

$$\int_{\partial G} \mathbf{F}(\mathbf{x}) \vec{dS}$$

sum up a vector field \mathbf{F} —for example a force or a velocity—over surfaces or hypersurfaces, in space, e.g., over the surface ∂G of a domain G . In particular, they play a role in modeling if, as in the case of fluid mechanics, the heat or mass transfer across the boundaries of a domain is to be described. Of central importance for the derivation of many models of physics which are based on *conservation laws* (conservation of energy, momentum, and mass), is *Gauß' theorem*, also called *Divergence theorem*, which defines the transformation of surface integrals to volume integrals and vice versa and will appear in various places of this text. Under certain regularity assumptions, it holds that,

$$\int_G \operatorname{div} \mathbf{F}(\mathbf{x}) dx = \int_{\partial G} \mathbf{F}(\mathbf{x}) \vec{dS}.$$

Perspectives of Functional Analysis

Functional analysis is concerned with *functionals*, i.e., functions which themselves have functions as arguments. Hereby, *function spaces* play a central role, such as, for example, the space $\mathcal{C}^n([a, b])$ of n -times continuously differentiable functions on $[a, b]$. A classical example of such function spaces are *Hilbert spaces*. Hilbert spaces are understood to be complete inner product spaces, i.e., vector spaces with inner products in which every so-called *Cauchy sequence* (a certain class of sequences) converges toward an element of the space. In modern mathematical modeling, certain classes of Hilbert spaces such as *Sobolev spaces* have achieved central significance.

2.2.3 Significance for Modeling and Simulation

Analysis is used mostly for the mathematical modeling as well as for *numerical analysis*, i.e., the analysis of numerical methods and their properties such as the quality of the approximation or the convergence behavior. The former is obvious, since the well-known models in mathematical physics, for example, are hardly imaginable without the use of differential and integral calculus in higher dimensions. The idea of convexity, for example, frequently plays a major role in the analysis of a model, as with statements regarding the existence and uniqueness of the solutions of a model.

But also the field of numerical analysis makes use of analysis. In this context, *smoothness* is a central idea having a qualitative nature: The greater the number of times a function is continuously differentiable (i.e., differentiable with continuous derivative), the *smoother* it is. *Smoothness conditions*, often referred to as *regularity conditions*, play an essential role for continuous mathematical models: In general, the more smoothness that can be assumed for the model's solution, the easier it is to make statements regarding approximation or convergence. Often it is the smoothness of the problem which determines the choice of the numerical method: Approximation techniques of higher order approximate “better”, however, they often work only under certain restrictive assumptions—namely these smoothness conditions. In addition, Taylor and other approximations are oftentimes an indispensable tool for the assessment of the quality of an approximation method.

Furthermore, the analysis also provides inspiration for numerical methods. The first approaches for the approximation of derivatives (i.e., their *discretization*), e.g., via difference quotients, is based on the definition of the derivate—a fact from which many algorithms for the numerical solution of models based on differential equations benefit today. A further example is given by Fubini's theorem, whose statement has been the starting point for numerous methods of multivariate (multidimensional) quadrature.

As well, linear algebra is indispensable for modeling as much as for simulation. For linear systems of ordinary differential equations, for example, the eigenvalues of

the system matrix are essential for the behavior of the system. And it goes without saying that numerics, whose essence consists of *discretizing*, i.e., of pressing continuous models into a finite corset, cannot exist without linear algebra.

Given this gallop through the fields of linear algebra and analysis, we will now turn our attention to those subjects of Applied Mathematics central to modeling and simulation—first stochastics and statistics, then numerical analysis.

2.3 Stochastic and Statistical Issues

From a historical perspective, randomness plays a large role in our context due primarily to the success of *Monte-Carlo-simulations*. Presently, probabilistic elements are almost ubiquitous in modeling and simulation. *Stochastic processes* are used for the modeling and simulation of sequences of events, while *stochastic differential equations* permit the integration of effects such as noise into models of differential equations, and *stochastic finite elements* enable their realization into discretization schemes. For this reason, we will frequently encounter the description of randomness and randomness itself in this book. For a reference and additional study, we refer to “Statistics for engineering and the sciences” by William Mendenhall and Terry Sincich [40].

2.3.1 Why Randomness?

Randomness and *probability* in modeling and simulation? What good are simulation results that can discern only up to a certain probability? These doubts, however, are justified only at a first glance.

First of all, there is the phenomenon of *uncertainty*: While everything may be deterministic, it is information that is missing. It is, for example, completely unknown to the printer when the next print job arrives—from its perspective, its arrival is random or only predictable in probabilistic quantities. For this reason, arrival processes in queueing systems are modeled using stochastic processes. Furthermore, one is often mostly interested in averaged quantities such as, for example, the response times of a system for a randomly chosen input or its mean over 100 operating hours. For circumstances in which measured data plays a role, whether for the initial or boundary values of a simulation or for its experimental validation, a statistical assessment of the data is generally required. And finally, there are the already mentioned *Monte-Carlo-simulations*—scenarios, in which the same method is performed multiple times under random constellations, i.e., constellations obeying a certain distribution, in order to obtain the desired result by averaging at the end. This procedure enjoys great popularity, particularly in areas where conventional simulation methods do not exist or are unacceptably costly (e.g., in very high-dimensional problem settings).

We offer an example to illustrate the significance of randomness. Let us assume the following task: A computer center receives a monetary donation from a benefactor—what shall the manager do with the money? He can route faster connections, install better switches, buy a supercomputer, provide more workplaces for users, improve the network connectivity to the outside, etc. In order to reach a well-founded decision, he would want to use simulations to determine the bottlenecks of the existing system and thus determine starting points for improvements. To this end, he must first generate representative input data, e.g., arrival or login times of users—which requires the generation of (pseudo-) random numbers. Second, a suitable mathematical model which allows for realistic computations of the computer center must be set up—this is usually carried out through stochastic processes and queueing networks. Last, the long columns of computed values must be correctly interpreted, and a corresponding purchase decision is to be made—for this, the available (random-based) data has to be analysed and evaluated.

2.3.2 Discrete Probability Spaces

Events

Probability theory facilitates the modeling of situations having an uncertain (not decisively predictable) outcome. We call such situations *non-deterministic experiments*. Non-deterministic experiments have possible *results* or *outcomes* w_i , which are summarized in the *sample space* $\Omega = \{w_1, w_2, \dots\}$. The notion indicates that we will initially only consider *discrete*, i.e., finite or countably infinite sample spaces. An event $E \subseteq \Omega$ is an arbitrary subset of Ω . One says that “ E occurs” if we obtain $w_i \in E$ as the outcome of our non-deterministic experiment. The empty set \emptyset is called an *impossible* event, Ω is called a *certain* event. Defining events in terms of sets allows us to work with its associated framework. In particular, given two events A and B , we obtain the derived events $A \cap B$, $A \cup B$, \bar{A} and $A \setminus B$:

- $A \cap B$: Events A and event B both occur. In the case of the empty intersection, we call A and B *disjoint*.
- $A \cup B$: Event A or event B occur.
- \bar{A} describes the event *complementary* to A , i. e., the event A does not occur.
- $A \setminus B$: Event A occurs, but event B does not occur.

Frequency and Probability

The term *frequency* is very concrete and relates to the past—we observe non-deterministic experiments and keep a tally of their outcomes, of the *absolute frequency* of an event E (number of cases in which E occurred), or of the *relative frequency* of E (quotient of absolute frequency and total number of all performed

observations). The relative frequency determines our expectation for the future event. This is also known as the *probability* of the event.

Probability Spaces

A *discrete probability space* consists of a discrete sample space Ω made up of *elementary events* $\{w_i\}$, $i = 1, \dots$, with assigned *elementary probabilities* p_i , in which

$$0 \leq p_i \leq 1, \quad \sum_{w_i \in \Omega} p_i = 1.$$

The probability of an arbitrary event E is defined as

$$P(E) := \sum_{w_i \in E} p_i.$$

If Ω is finite, then one speaks of a *finite probability space*. Such a probability space can be viewed as a mathematical model for a non-deterministic experiment. Among the things we are concerned with in the following are techniques that forecast or predict the occurrence of events in certain discrete cases—especially in cases in which counting is not so easy.

The following *addition theorem* holds for pairwise disjoint events A_1, A_2, \dots :

$$P\left(\bigcup_i A_i\right) = \sum_i P(A_i).$$

If the A_i are not pairwise disjoint, it becomes somewhat more complicated. One obtains the so-called *sieve principle* or the *inclusion/exclusion principle*. We restrict our attention to the case of two events:

$$P(A_1 \cup A_2) = P(A_1) + P(A_2) - P(A_1 \cap A_2).$$

Conditional Probabilities

Because the initial situation, and hence the probability of an event, changes when one has some partial knowledge or when one becomes aware of part of the previous past, one introduces so-called *conditional events* with *conditional probabilities*, and writes $A|B$ as well as $P(A|B)$ and says “ A under the condition B ”. In contrast to conditional probability, one also refers to $P(A)$ as the *absolute* or *total* probability. Now let A and B be events with $P(B) > 0$. Then one defines

$$P(A|B) := \frac{P(A \cap B)}{P(B)}.$$

It holds that $P(A|B) \geq 0$, $P(A|B) \leq 1$, $P(B|B) = 1$, $P(B|\bar{B}) = 0$ as well as $P(A|\Omega) = P(A)$. For an arbitrary, but fixed event B with $P(B) > 0$, the conditional probabilities with respect to B form a new probability space with new probability P_B , which expresses the changed level of information. The conditional probability leads to the *multiplication theorem*: Let n events A_1, \dots, A_n be given with $P(A_1 \cap \dots \cap A_n) > 0$. Then,

$$P(A_1 \cap \dots \cap A_n) = P(A_1) \cdot P(A_2|A_1) \cdot P(A_3|A_1 \cap A_2) \cdot \dots \cdot P(A_n|A_1 \cap \dots \cap A_{n-1}).$$

The first central theorem to use conditional probabilities is the *Law of total probability*. It states how knowledge of the conditional probabilities of an event lets one determine the unconditional (total) probability. Let the pairwise disjoint events A_1, A_2, A_3, \dots be given and let $B \subseteq \bigcup_i A_i$ be an additional event. Then there holds (in the finite as well as in the infinite case)

$$P(B) = \sum_i P(B|A_i) \cdot P(A_i).$$

Bayes' theorem is the another important theorem that must be mentioned in the context of probabilities. Again, let the pairwise disjoint events A_1, A_2, A_3, \dots be given and let $B \subseteq \bigcup_i A_i$ be an additional event with $P(B) > 0$. Then it holds (in the finite as well as in the infinite case) for arbitrary $j = 1, 2, 3, \dots$,

$$P(A_j|B) = \frac{P(A_j \cap B)}{P(B)} = \frac{P(B|A_j) \cdot P(A_j)}{\sum_i P(B|A_i) \cdot P(A_i)}.$$

Thus, Bayes' theorem allows one to exchange the order of the condition: Starting with the probabilities of “ B given the condition A_i ”, one makes statements about “ A_i given the condition B ”.

Independence

Independence is a central term of probability theory. Many statements only hold, or are significantly easier, in the case of independence. Two events A and B are called *independent* if it holds that,

$$P(A \cap B) = P(A) \cdot P(B).$$

The events A_1, \dots, A_n are called independent if for all index sets $I \subseteq \{1, \dots, n\}$ it holds that,

$$P\left(\bigcap_{i \in I} A_i\right) = \prod_{i \in I} P(A_i).$$

In the case of an infinite number of events, one requires the above condition for all finite index sets, i.e., for every finite combination of events.

Random Variables

A *random variable* X is understood to be a mapping $\Omega \rightarrow \mathbb{R}$. In the case of a finite or countably infinite sample space Ω , the random variable is also called *discrete* since then its range W_X is also discrete. A random variable is used to define events—after all, one is interested in the probability that X assumes a certain value. One writes, e.g., in short $P(X \leq x)$ (capital letters for random variables, lower case letters for their values) which means,

$$P(X \leq x) = P(\{w \in \Omega : X(w) \leq x\}) .$$

Two special cases are of particular importance and receive individual names:

$$f_X : \mathbb{R} \rightarrow [0, 1], \quad x \mapsto P(X = x) , \quad F_X : \mathbb{R} \rightarrow [0, 1], \quad x \mapsto P(X \leq x) .$$

f_X is called a (*discrete*) *density (function)* of X , while F_X is called the (*discrete*) *distribution (function)* of X .

Analogous to the notion of the independence of events one also speaks of independent random variables. The random variables X_1, \dots, X_n are called *independent* if for every n -tuple of values (x_1, \dots, x_n) of the joint range of $W_{X_1} \times \dots \times W_{X_n}$ it holds that,

$$P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i) ,$$

i.e., if the joint density equals the product of individual densities.

Expected Value

Since it is somewhat impractical to characterize a random variable X via f_X and F_X , one introduces characteristic numbers, the so-called *moments*. The *expected value*, or equivalently the *first moment*, $E(X)$, of a discrete random variable X is defined by,

$$E(X) := \sum_{w_i \in \Omega} X(w_i) \cdot p_i = \sum_{x \in W_X} x \cdot P(X = x) = \sum_{x \in W_X} x \cdot f_X(x) .$$

The expected value indicates which result can be expected on average. In the case of an infinite range W_X , the expected value $E(X)$ is defined only if the sequence in the

above definition converges. It follows directly from its definition that the expected value is linear: Given any two random variables X_1 and X_2 and $a \in \mathbb{R}$, it holds that $E(aX_1 + X_2) = aE(X_1) + E(X_2)$.

If X is a random variable, it holds that $Y := f(X)$ is also a random variable as long as f is defined on the range W_X and maps it into \mathbb{R} . For $E(f(X))$, it then holds that,

$$\begin{aligned} E(f(X)) &= \sum_y y \cdot P(Y = y) \\ &= \sum_y y \cdot \sum_{x:f(x)=y} P(X = x) = \sum_x f(x) \cdot P(X = x). \end{aligned}$$

Conditional probabilities also play a large role in the context of random variables. For an event A having positive probability, one can introduce the *conditional random variable* $X|A$. It has density,

$$f_{X|A} := P(X = x|A) = \frac{P(X = x \cap A)}{P(A)}.$$

The *conditional expected values* $E(X|A)$ are defined accordingly.

Variance

The expected value does not completely characterize a random variable or its respective distribution. Also of importance is the quantity of *scattering* of the data around the expected value. For a random variable X having expected value $\mu := E(X)$, we define the *second moment* as $E(X^2)$ and the *variance* or, equivalently the *second central moment*, $V(X)$ as

$$V(X) := E((X - \mu)^2),$$

and the *standard deviation* $\sigma(X)$ as

$$\sigma(X) := \sqrt{V(X)}.$$

The following connection between variance and second moment holds:

$$V(X) := E(X^2) - (E(X))^2 = E(X^2) - \mu^2.$$

Furthermore, for an arbitrary random variable X and $a, b \in \mathbb{N}$, the following important property holds as well,

$$V(aX + b) = a^2 \cdot V(X).$$

In general, one defines the *k-th moment* of a random variable X as $E(X^k)$ and the *k-th central moment* as $E((X - E(X))^k)$. The collection of all moments completely characterizes a random variable. However, only the first two moments and central moments are relevant in practice.

For the “relative variance” or equivalently, the “relative standard deviation”, one defines the *coefficient of variation* $\varrho(X)$:

$$\varrho(X) = \frac{\sigma(X)}{E(X)}.$$

Bernoulli Distribution

A binary random variable X (i.e., $W_X = \{0, 1\}$) with density function,

$$f_X(x) = \begin{cases} p & \text{for } x = 1, \\ 1 - p & \text{for } x = 0 \end{cases}$$

is called *Bernoulli distributed* with parameter p . p is also called the *probability of success*, for $1 - p$ oftentimes the notation q is used. For the expected value of a Bernoulli distributed random variable there holds $E(X) = p$, for its variance there holds $V(X) = pq$.

Binomial Distribution

Let n independent random variables X_1, \dots, X_n be given having a Bernoulli distribution along with probability of success p (equal for all X_i). A *binomially distributed* random variable with parameters n and p (in symbols: $X \sim \mathcal{B}(n, p)$) is then obtained through the sum

$$X := X_1 + \dots + X_n.$$

The binomial distribution requires two parameters for its characterization, n and p . The density of the binomial distribution is given by,

$$f_X(k) = \binom{n}{k} \cdot p^k \cdot q^{n-k}.$$

Through the additivity property of expected value and variance we finally obtain $E(X) = np$ and $V(X) = npq$. The sum of independent, binomially distributed random variables is itself binomially distributed.

Geometric Distribution

The *geometric distribution* is also based on a Bernoulli experiment. We will again refer to p as the probability of success and repeat the Bernoulli experiment independently as many times as needed until we obtain the first case of success. The random variable X denotes the number representing the repeated attempts for the first successful outcome and now has a geometric distribution. The density function is then given by $f_X(i) = p \cdot q^{i-1}$, while the expected value is given by $E(X) = \frac{1}{p}$, and the variance by $V(X) = \frac{q}{p^2}$.

Poisson Distribution

The final discrete distribution we will consider is the *Poisson distribution*. It has a parameter $\lambda > 0$ and its range is all of \mathbb{N}_0 . The density of the Poisson distribution is defined as

$$f_X(i) = \frac{e^{-\lambda} \lambda^i}{i!}, \quad i \in \mathbb{N}_0.$$

For the expected value, it holds that $E(X) = \lambda$, while for the variance it can be shown that $V(X) = \lambda$. One may actually interpret the Poisson distribution as the limiting case of a binomial distribution. One additional and convenient property: Sums of independent random variables having a Poisson distribution themselves have a Poisson distribution in which the defining parameters are found through simple addition.

In practice, the Poisson distribution applies to the modeling of the counting of certain events (a car passes by, a print job arrives at the printer, a cavalry horse kicks a soldier fatally ...). This is an important application in connection with the simulation of traffic in computer systems via stochastic processes.

2.3.3 Continuous Probability Spaces

Continuous Random Variables

At times, however, the phenomena to be analysed are *continuous*: the distances between two arriving print jobs, the intensity of the arriving light in an optical fiber or the temperature in a cooled mainframe computer. Furthermore, discrete events oftentimes occur in such large numbers that a continuous description (e.g. using limit processes) becomes practical.

In the continuous case, one no longer considers the sample space Ω but rather takes the probability space defined by a random variable directly, i.e., $\Omega = \mathbb{R}$. In contrast to discrete random variables one now speaks of *continuous* random

variables. A continuous random variable X and the underlying probability space are defined through a *density (function)* $f_X : \mathbb{R} \rightarrow \mathbb{R}_0^+$ which has to be integrable and must satisfy

$$\int_{-\infty}^{\infty} f_X(x)dx = 1.$$

It is not meaningful to state the probability of an elementary event (“the rain hits a particular spot”). Instead, one allows as events only certain subsets A of the real axis (countable unions of pairwise disjoint intervals of an arbitrary kind). This choice has mathematical motivations. Such an event A occurs if X assumes a value $x \in A$, and its probability is defined as

$$P(A) = \int_A f_X(x)dx.$$

The prototype of such an event is the interval open to the left and closed to the right $] -\infty, x]$. For the probability of such intervals, this new notion of a *distribution (function)* is introduced:

$$P(X \in] -\infty, x]) = P(X \leq x) = F_X(x) = \int_{-\infty}^x f_X(t)dt.$$

In contrast to the density function, which does not have to be continuous, this distribution is always continuous and monotonically increasing. Furthermore, it holds that,

$$\lim_{x \rightarrow -\infty} F_X(x) = 0, \quad \lim_{x \rightarrow \infty} F_X(x) = 1.$$

Functions of Continuous Random Variables

The notion of the continuous probability space was defined in exactly this way so that the most relevant properties and computation rules of the discrete case still hold. This is also true for the terms *independence* and *conditional probability*. For a given function $g : \mathbb{R} \rightarrow \mathbb{R}$ and a given random variable X , one can define a new random variable $Y := g(X)$. Its distribution is given by,

$$F_Y(y) = P(Y \leq y) = P(g(X) \leq y) = \int_C f_X(t)dt,$$

where C denotes the set of all real numbers t which satisfy $g(t) \leq y$. Naturally, C must be an event (in the sense of our definition).

Moments of Continuous Random Variables

Analogous to the previous considerations, the notions of expected value and variance can be generalized to the continuous case essentially by replacing the sums with integrals. The *expected value* of a continuous random variable X is defined as

$$E(X) = \int_{-\infty}^{\infty} t \cdot f_X(t) dt .$$

Here, the integral on the right hand side has to converge absolutely. Analogous to the previous situation, the generalization of the definition of *variance* is given by,

$$V(X) = E((X - E(X))^2) = \int_{-\infty}^{\infty} (t - E(X))^2 \cdot f_X(t) dt .$$

For the expected value of a composite random variable $g(X)$,

$$E(g(X)) = \int_{-\infty}^{\infty} g(t) \cdot f_X(t) dt .$$

In the case of a linear function g , i.e., $g(X) := aX + b$, this implies, in particular, that

$$E(aX + b) = \int_{-\infty}^{\infty} (at + b) \cdot f_X(t) dt .$$

Uniform Distribution

The *uniform distribution* is the easiest continuous distribution. It is based on the Laplace principle. Let the interval $\Omega = [a, b]$ be given as the sample space. On Ω , the density function for a uniformly distributed random variable X is given by,

$$f_X(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] , \\ 0 & \text{else.} \end{cases}$$

Determining the antiderivative yields the distribution F_X :

$$F_X(x) = \int_{-\infty}^x f(t) dt = \begin{cases} 0 & \text{for } x < a , \\ \frac{x-a}{b-a} & \text{for } a \leq x \leq b , \\ 1 & \text{for } x > b . \end{cases}$$

For the expected value, variance, standard deviation, and coefficient of variation, there hold the formulas

$$E(X) = \frac{a+b}{2}, \quad V(X) = \frac{(b-a)^2}{12}, \quad \sigma(X) = \frac{b-a}{2\sqrt{3}}, \quad \varrho(X) = \frac{b-a}{\sqrt{3}(a+b)}.$$

Normal Distribution

The *normal distribution* may be the most important continuous distribution, this holding in particular for statistics. It is oftentimes well suited for the modeling of quantities of chance which fluctuate around a certain value. A random variable X with range $W_X = \mathbb{R}$ is called *normally distributed* with the parameters $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$ (in symbols $X \sim \mathcal{N}(\mu, \sigma^2)$), if it has the density

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

The distribution function is once more obtained through the antiderivative

$$F_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \int_{-\infty}^x e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt.$$

However, f_X cannot be integrated in closed form— F_X can only be computed approximately using numerical integration.

Of particular importance is the *standard normal distribution* $\mathcal{N}(0, 1)$ with parameters 0 and 1. Its distribution function

$$\Phi(x) := \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^x e^{-\frac{t^2}{2}} dt$$

is called the *Gaussian function* and its graph is called the *Gaussian bell curve*. The values of Φ have been tabulated. A nice property of the normal distribution is the fact that the linear transformation $Y := aX + b$ ($a \neq 0$) of a (μ, σ^2) -normally distributed random variable is itself normally distributed, i.e.,

$$Y \sim \mathcal{N}(a\mu + b, a^2\sigma^2).$$

Using the above relationship, we can *scale* $\mathcal{N}(\mu, \sigma^2)$ and trace it back to the standard normal distribution $\mathcal{N}(0, 1)$:

$$X \sim \mathcal{N}(\mu, \sigma^2) \quad \Rightarrow \quad Y := \frac{X - \mu}{\sigma} \sim \mathcal{N}(0, 1).$$

It holds then, that $E(X) = \mu$ and $V(X) = \sigma^2$. Therefore, the standard normal distribution has expected value 0 and variance 1. Thus, the two parameters of the normal distribution are just its two first central moments.

The importance of the normal distribution stems from the fact that many non-deterministic experiments are *asymptotically* normally distributed. For example, this holds for the binomial distribution for increasing n .

Exponential Distribution

A random variable X is called *exponentially distributed* with the parameter $\lambda > 0$, if it has as its density function,

$$f_X(x) = \begin{cases} \lambda \cdot e^{-\lambda x} & \text{in the case that } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Integration of f_X yields its distribution function,

$$F_X(x) = \int_0^x \lambda \cdot e^{-\lambda t} dt = 1 - e^{-\lambda x}$$

for $x \geq 0$ and $F_X(x) = 0$ for $x < 0$. Employing integration by parts, one obtains the expected value and variance:

$$E(X) = \frac{1}{\lambda}, \quad V(X) = \frac{1}{\lambda^2}.$$

With its exponentially decreasing characteristic, the exponential distribution can be viewed to some extent as the continuous counterpart of the geometric distribution. The exponential distribution is *the* important distribution for the description of intervals between certain occurring events (print jobs at the printer...) or in general, of waiting times. Because of this, it is therefore of exceptional importance for the areas of queueing theory and stochastic processes. For an exponentially distributed random variable X with parameter λ and $a > 0$, the random variable $Y := aX$ is also exponentially distributed, however, here with respect to the parameter λ/a .

The exponential distribution is *memoryless*: For all positive x and y , it holds that,

$$P(X > x + y | X > x) = \frac{P(X > x + y)}{P(X > x)} = \frac{e^{-\lambda(x+y)}}{e^{-\lambda x}} = e^{-\lambda y} = P(X > y).$$

The exponential distribution is the only continuous memoryless distribution. This distribution can be interpreted as the limit of the geometric distribution. This is an example how discrete and continuous random variables or distributions, resp., can be related through limit processes.

2.3.4 Asymptotics

Markov's and Chebyshev's Inequalities

The assumption of complete knowledge about a random variable is oftentimes unrealistic. In particular, its distribution in many cases cannot be given in closed form. Sometimes, one can at least compute the expected value or possibly the variance. In such cases, one is interested in approximate information on probabilities or estimates. In this case, *Markov's inequality* proves helpful: Let X be a random variable with $X \geq 0$ and expected value $E(X)$. Then, for all positive real t , it holds that

$$P(X \geq t) \leq \frac{E(X)}{t}.$$

Chebyshev's inequality establishes an additional asymptotic estimate as well. Let X be a random variable and let t be a positive real number. It holds then, that

$$P(|X - E(X)| \geq t) \leq \frac{V(X)}{t^2}.$$

This means that the smaller the variance, the larger the probability that X assumes values within a certain interval around $E(X)$ —which agrees well with our intuition for variance being a good measure for scattering.

Law of Large Numbers

A very important asymptotic relationship is formulated and quantified by the famous *law of large numbers*: Let independent and identically distributed random variables $X_i, i = 1, 2, \dots$ be given (one also speaks of an *iid sequence*—that is, an independent and identically distributed—of random variables) with expected value μ and variance σ^2 . Furthermore, let positive numbers ε, δ as well as $n \in \mathbb{N}$ with $n \geq \frac{\sigma^2}{\varepsilon \delta^2}$ be given. The arithmetic mean Z_n of X_1, \dots, X_n is then defined:

$$Z_n := \frac{1}{n} \cdot \sum_{i=1}^n X_i.$$

It then holds that,

$$P(|Z_n - \mu| \geq \delta) \leq \varepsilon.$$

For sufficiently large n , i.e., for a sufficiently large number of repetitions of a non-deterministic experiment, the average of the outcomes lies with arbitrarily small

probability ε outside of an arbitrarily small interval (δ) centered around the expected value of the one-time experiment. Thus, if we are prepared for tremendous effort, we can come arbitrarily close to the expected value.

Monte Carlo Methods

In several numerical problem settings the so-called *Monte Carlo methods* have become customary. As the name already indicates, chance plays a role here. The “justification” of such an approach lies rooted in the law of large numbers. A particularly popular field for applying Monte Carlo methods is numerical quadrature, especially for high dimensional integrals as they appear in mathematical finance, physics, or in probability theory (expected values are after all integrals!). We consider a simple two dimensional example: Let the characteristic functions $\chi_A(x, y)$ and $\chi_B(x, y)$ for the circle $A := \{(x, y) : x^2 + y^2 \leq 1\}$ and rectangle $B := [-1, 1]^2$, respectively, be given (with values 1 within and value 0 outside the domain). We are interested in the integral of $\chi_A(x, y)$, i.e., the number π . Our non-deterministic experiment: Randomly choose a point $(x_i, y_i) \in B$ (uniform distribution!). Our random variables Z_i : $Z_i = \chi_A(x_i, y_i)$. Then, the law of large numbers says that for increasing n , $\frac{1}{n} \cdot \sum_{i=1}^n Z_i$ converges toward $\pi/4$ in the probabilistic sense (but not “guaranteed”!).

Central Limit Theorem

This theorem also represents an asymptotic result. It makes a statement pertaining to the question why the normal distribution is so normal, and, as well, it explains the outstanding significance of the normal distribution in statistics. Let an iid-sequence of random variables X_i , $i = 1, 2, \dots$ be given, with expected value μ and variance $\sigma^2 > 0$. Additionally, let the random variables Y_n , $n = 1, 2, \dots$ be defined to be the n -th partial sums of the X_i , i.e.,

$$Y_n := \sum_{i=1}^n X_i .$$

Furthermore, given these Y_n , let the scaled random variables Z_n be defined by

$$Z_n := \frac{Y_n - n\mu}{\sigma\sqrt{n}} .$$

Then the Z_n are *asymptotically standard normally distributed*, i.e.,

$$\lim_{n \rightarrow \infty} Z_n \sim \mathcal{N}(0, 1) .$$

In particular, the sequence of distribution functions F_{Z_n} associated with the Z_n , satisfy

$$\lim_{n \rightarrow \infty} F_{Z_n}(x) = \Phi(x) \quad \forall x \in \mathbb{R}.$$

One can even go one step further and drop the assumption requiring it to be identically distributed. Now let the X_i be independent with expected values μ_i and variances $\sigma_i^2 > 0$. Under a very weak additional assumption, the asymptotic standard normal distribution also holds for the

$$U_n := \frac{Y_n - \sum_{i=1}^n \mu_i}{\sqrt{\sum_{i=1}^n \sigma_i^2}}.$$

2.3.5 Statistical Inference

In modeling, one often is confronted with systems that are complex and known only in terms of some of their parts. Nevertheless, one still would like to derive statements regarding the system behavior (for example to determine model parameters for a subsequent simulation)—and these should also be as reliable and meaningful as possible. Generally, there is nothing else to do but to observe (“to take a sample”) and then to assess the obtained data statistically. To this end, three central tools are available:

- *Estimators* or *estimator variables* are functions whose values are based on observations that are used in estimating characteristics such as expected value or variance.
- An estimator is a point quantity. Frequently, one is rather interested in quantities that can cover a range of values, i.e., intervals, in which the expected value or variance lie with high probability. Such intervals are called *confidence intervals*.
- Finally, one frequently has to make decisions: Is the router defective or not? Is the average access time for disc 1 shorter than for disc 2? To this end, one performs *tests*—one tests hypotheses for their plausibility based on the available data.

Therefore, in the following we take *samples* as the basis for further analyses. The individual measurements are called *sampling variables*, their number is called the *sample size*.

Estimators

Let X be a given random variable with density function f_X and define a parameter θ which is to be estimated. An *estimator* for θ is simply a random variable Y which is constructed from several sampling variables (typically iid). An estimator Y is called

unbiased if $E(Y) = \theta$ holds. All meaningful estimators should satisfy this property. An estimator is called *variance reducing*, if it holds that $V(Y) < V(X)$. In this context, the variance is also called *mean square deviation*. Given two unbiased estimators, the one with the smaller variance is called *more efficient*.

The most obvious estimator for $\theta = E(X)$ is the *mean of the sample* or *sample mean* \bar{X} :

$$\bar{X} := \frac{1}{n} \cdot \sum_{i=1}^n X_i .$$

The sample mean \bar{X} is unbiased, more efficient than X and consistent in the quadratic mean (the variance goes toward zero as $n \rightarrow \infty$):

$$V(\bar{X}) = V\left(\frac{1}{n} \cdot \sum_{i=1}^n X_i\right) = \frac{1}{n^2} \cdot \sum_{i=1}^n V(X_i) = \frac{1}{n} \cdot V(X) .$$

Chebyshev's inequality yields

$$P(|\bar{X} - \theta| \geq \varepsilon) \leq \frac{V(X)}{n\varepsilon^2} \rightarrow 0 \quad \text{for } n \rightarrow \infty .$$

This again shows: For a sufficiently large sample size, the sample mean lies arbitrarily close to the value θ to be estimated with arbitrarily high probability.

As a standard estimator for the variance (now $\theta = V(X)$) we define the *sample variance* S^2 :

$$S^2 := \frac{1}{n-1} \cdot \sum_{i=1}^n (X_i - \bar{X})^2 .$$

We point out two deviations from the definition of variance: In the sum, the expected value $E(X)$ is replaced by the sample mean \bar{X} , and the factor in front of the sum is now $\frac{1}{n-1}$ instead of $\frac{1}{n}$. The sample variance is unbiased.

Confidence Intervals

Estimators are often unsatisfactory since they do not make any statements about how close the estimated value comes to the parameter θ which is to be estimated. Therefore, one frequently considers *two* estimator variables U_1 and U_2 which serve to bound θ with high probability:

$$P(U_1 \leq \theta \leq U_2) \geq 1 - \alpha$$

for (small) $\alpha \in]0, 1[$. The probability $1 - \alpha$ is called the *confidence level*, the interval $[U_1, U_2]$ or, for a concrete sample, $[u_1, u_2]$ is called the *confidence interval*. With α one can account for one's own readiness to assume risk: A small α implies only a small risk that θ is still outside the confidence interval. Consequently, this must be pretty large (and thus possibly not very meaningful). Conversely, a larger α implies only a moderate probability that our confidence interval does, in fact, contain θ . In such a case one can choose a proportionately smaller confidence interval. The confidence level can be chosen by the person performing the experiment—it expresses his security need. Since one aims for preferably small confidence intervals, one usually starts their construction with “ $\dots = 1 - \alpha$ ” instead of “ $\dots \geq 1 - \alpha$ ”.

Often, one employs only one estimator variable U and uses it to construct a symmetric confidence interval $[U - \delta, U + \delta]$. Let X be a continuous random variable with distribution function F_X . Then every number x_γ with

$$F_X(x_\gamma) = P(X \leq x_\gamma) = \gamma$$

is called a γ -*quantile* of X or of F_X . We say “every number” because quantiles are not necessarily unique (one considers, e.g., a distribution that is constant over an interval). In the case of the normal distribution, one denotes the γ -quantile by z_γ . For important distributions such as the standard normal distribution, the quantiles are tabulated.

Tests

The performance of tests is the supreme discipline of statistics. Through a test one generally wants to check certain conjectures related to the distribution (or its parameters).

We now consider the principles and components of tests. We assume that the random variable X has a Bernoulli distribution with $P(X = 1) = p$. First, we formulate the hypothesis which is to be checked, the so-called *null hypothesis* H_0 , and in addition it occasionally makes sense to formulate an *alternative* H_1 . One then collects a sample upon which the null hypothesis is accepted or rejected. This typically is done by making use of a *test statistic* which is derived from the sample variables and for which a usually simple *rejection region* (in general one or several intervals) is determined. It is also true for tests that nothing is for sure—all statements can only be made with respect to a certain probability. Thus, there always exists a chance for errors. One distinguishes two basic types of errors: A *type I error* is committed if the null hypothesis is rejected as a result of the test even though it is actually true. The probability of this is called the *risk of the first type*; a *type II error* is committed if the null hypothesis is not rejected even though it does not hold. The probability for this is called the *risk of the second type*. The goal of the test design is to keep both types of error risks as small as possible. However, we cannot minimize both simultaneously: If one chooses an empty rejection region, and therefore never

rejects the null hypothesis, then one can never commit an error of the first type. However, if the null hypothesis is wrong, then the error of second type occurs for sure. A corresponding statement holds in the case of a chronic rejection of the null hypothesis. Therefore, a reasonable compromise between both errors has to be sought. The maximal admissible *risk of first type* is denoted by α and is called the *statistical significance* of the tests. Typically one prescribes α (e.g., 0.05 or 0.01) and then formulates the decision rule so that the type I risk assumes exactly the value α , since then in general the risk of a type II error is minimal.

For hypotheses we now define $H_0 : p \geq p_0$ as well as $H_1 : p < p_0$, as the test quantities, the binomially distributed sum T_n is used:

$$T_n := \sum_{i=1}^n X_i .$$

In this case, the rejection region is most appropriately chosen as an interval of the type $[0, k]$ with suitable k (the larger p , the larger values of T_n can be expected)—i.e., we therefore construct a *one-sided test*. One can show by use of the central limit theorem that the normed random variable,

$$Z_n := \frac{T_n - np}{\sqrt{np(1-p)}}$$

is asymptotically standard normally distributed. With the notation $P(T_n \leq k; p)$ for the respective probability of $T_n \leq k$, if the probability of success is assumed to be the value p , we obtain for the type I risk,

$$\begin{aligned} \alpha &= \max_{p \geq p_0} P(T_n \leq k; p) = P(T_n \leq k; p_0) = P\left(Z_n \leq \frac{k - np}{\sqrt{np(1-p)}}\right) \\ &= P\left(Z_n \leq \frac{k - np_0}{\sqrt{np_0(1-p_0)}}\right) \approx \Phi\left(\frac{k - np_0}{\sqrt{np_0(1-p_0)}}\right). \end{aligned}$$

Thus, with the definition of the quantile it follows

$$\alpha \approx \Phi\left(\frac{k - np_0}{\sqrt{np_0(1-p_0)}}\right) \Rightarrow z_\alpha \approx \frac{k - np_0}{\sqrt{np_0(1-p_0)}}$$

and therefore $k \approx z_\alpha \cdot \sqrt{np_0(1-p_0)} + np_0$.

The test above that was constructed from scratch is rather the exception. In general, one will select the suitable test out of the multiple existing tests for the respective task and then execute this one according to the book. For example, one distinguishes *one sample tests* (only one random variable is analysed) and *two*

sample tests (two random quantities with possibly different distributions are to be compared), *parameter tests*, *distribution tests* or *independence tests*. We make explicit mention here of some examples that are of particular importance: the *approximate binomial tests*, the *Gaussian test*, the *t-test*, the χ^2 -*goodness-of-fit test* as well as the *Kolmogorov Smirnov test*.

After this excursion into stochastics and statistics, we now turn toward numerical analysis, which provides suitable simulation tools for numerous classes of models.

2.4 Numerical Aspects

Numerical Analysis, as stochastics, is an area within Applied Mathematics, and is concerned with the design and analysis of computational methods for continuous models, in particular from the area of linear algebra (solving linear systems of equations, computing eigenvalues etc.) and analysis (finding roots or extrema etc.). This is typically associated with *approximations* (solving differential equations, computing integrals) and is therefore rather atypical for traditional mathematics. The analysis of numerical algorithms revolves around the aspects of approximation accuracy, convergence speed, storage requirements and computing time. In particular, the later two topics lie at the center of *numerical programming*, which is an area within computer science that is also particularly concerned with implementation aspects. And everything works toward the goal to perform numerical simulations, in particular on high performance and supercomputers.

Whoever recognizes gaps in their understanding when reading the following sections on numerics and would like to close these before diving into the individual topics of modeling and simulation, is referred to, for example, “An introduction to numerical computations” by Sidney Yakowitz and Ferenc Szidarovszky [60] or to “Numerical Methods in Scientific Computing” by Germund Dahlquist and Åke Björck [14].

2.4.1 Basics

Discretization

In numerical analysis, we are confronted with problem settings that are *continuous*. Computers, however, are initially restricted to deal with *discrete* objects. This concerns numbers, functions, domains as well as operations such as differentiation. The magic word for the necessary transition, “continuous → discrete”, is *discretization*. Discretization always sits at the forefront of all numerical activities. One discretizes the real numbers through the introduction of *floating point numbers*, domains (for example, time intervals at the numerical solution of ordinary differential equations or spatial domains at the numerical solution of partial differential equations) through

the introduction of a *grid* consisting of discrete *grid points* and operators such as the derivative $\frac{\partial}{\partial x}$ by forming difference quotients of function values at neighboring grid points.

Floating Point Numbers

The set \mathbb{R} of real numbers is *unbounded* and *continuous*. The set \mathbb{Z} of integers, also unbounded, is discrete with constant unit distance 1 between two neighboring numbers. The set of *exactly representable numbers* on a computer, however, is inevitably finite and therefore discrete and bounded. Probably the easiest realization of such a number set with its corresponding computations is *integer arithmetic*. This, as well as *fixed point arithmetic*, works with fixed number ranges and fixed resolution. *Floating point arithmetic*, in contrast, permits varying the placement of the decimal point and therefore allows for variable magnitude, variable location of the representable number range as well as variable resolution.

The *normalized t-digit floating point numbers w.r.t. the basis B* ($B \in \mathbb{N} \setminus \{1\}$, $t \in \mathbb{N}$) is now defined:

$$\mathbb{F}_{B,t} := \{M \cdot B^E : M = 0 \text{ or } B^{t-1} \leq |M| < B^t, M, E \in \mathbb{Z}\}.$$

Here, M is called the *mantissa*, and E the *exponent*. The normalization (no leading zero) guarantees the uniqueness of the representation. The introduction of an admissible range for the exponent leads to the *machine numbers*:

$$\mathbb{F}_{B,t,\alpha,\beta} := \{f \in \mathbb{F}_{B,t} : \alpha \leq E \leq \beta\}.$$

The quadruple (B, t, α, β) completely characterizes the system of machine numbers. Therefore, a concrete number has to store M and E . The terms floating point number and machine number are often used synonymously; in context, B and t are usually obvious so that in the following we only write \mathbb{F} .

The *absolute distance* between two neighboring floating point numbers is not constant but depends on the respective magnitude of the numbers. The maximal possible *relative distance* between two neighboring floating point numbers is called the *resolution* ϱ . It holds:

$$\frac{(|M| + 1) \cdot B^E - |M| \cdot B^E}{|M| \cdot B^E} = \frac{1 \cdot B^E}{|M| \cdot B^E} = \frac{1}{|M|} \leq B^{1-t} =: \varrho.$$

The representable range is furthermore characterized by the *smallest positive machine number* $\sigma := B^{t-1} \cdot B^\alpha$ as well as the *largest machine number* $\lambda := (B^t - 1) \cdot B^\beta$.

A famous and important example is the number format of the IEEE (Institute of Electrical and Electronics Engineers), which is defined in the US-Norm ANSI/IEEE-Std-754-1985 and which can be traced back to a patent of Konrad

Zuse in the year 1936. It allows for the accuracy levels of *single precision*, *double precision* and *extended precision*. Single precision corresponds to about 6–7 decimal places, whereas double precision ensures about 14 places.

Rounding

Since floating point numbers are discrete, certain real numbers may slip through our fingers. These must be assigned to suitable floating point numbers in a meaningful way—one *rounds*. Important rounding strategies are *rounding down*, *rounding up*, *truncating* as well as *rounding correctly*, which always rounds toward the closest floating point number.

When rounding, one inevitably commits mistakes. These we can distinguish as the *absolute round-off error*, $\text{rd}(x) - x$, and the *relative round-off error*, $\frac{\text{rd}(x) - x}{x}$, for $x \neq 0$. The goal for the construction of floating point numbers is to obtain a high relative accuracy. It is the relative round-off error that plays a central role for all analyses and it must be estimated in order to assess the possible influence of round-off errors in a numerical algorithm. The relative round-off error is directly tied to the resolution.

Floating Point Arithmetic

For the simple rounding of numbers, one knows the exact value. Here, this immediately changes for the simplest computations since, starting with the first arithmetic operation, one exclusively works with approximations. The exact execution of the basic arithmetic operations $* \in \{+, -, :, /\}$ in the system \mathbb{F} of floating point numbers is, in general, impossible—even for elements in \mathbb{F} : How shall one write the sum of 1234 and 0.1234 exactly using four places? Therefore, we need a “clean” floating point arithmetic that prevents a build-up of accumulated errors. The ideal case (and required by the IEEE-standard for the basic arithmetic operations and for the square root) is the *ideal arithmetic* in which the computed result corresponds to the rounding of the exact result.

Round-off Error Analysis

A numerical algorithm is a finite sequence of basic arithmetic operations with a uniquely determined succession. Floating point arithmetic provides a significant source for errors in numerical algorithms. For a numerical algorithm, the most important objectives concerning floating point arithmetic are therefore a *small discretization error*, *efficiency* (run-times as small as possible) as well as a *small influence of round-off errors*. The later objective requires an *a-priori round-off error*

analysis: Which bounds can be stated for the total error given that a certain quality is assumed for the elementary operations?

Condition

Conditioning is a central, but usually only qualitatively defined term in numerical analysis: How large is the sensitivity of the solution of a problem toward changes in the input data? In the case of high sensitivity, one speaks of *ill conditioning*, or of an *ill conditioned problem*. In the case of little sensitivity, one correspondingly speaks of *good conditioning* and *well-conditioned problems*. Very important: Conditioning is a property of the considered *problem*, not of the algorithm to be used.

Perturbations δx in the input data must therefore be studied since inputs oftentimes are available only inaccurately (obtained from measurements or from previous calculations) and thus such perturbations are common even in exact arithmetic. Ill-conditioned problems are not only difficult to treat numerically, but could possibly not even be treated at all in the most extreme case.

Among the basic arithmetic operations, only subtraction has the potential to be ill-conditioned. This is due to the so-called *loss of significance* which is the effect that could occur when subtracting two numbers of equal sign in which leading identical digits cancel each other out, i.e., leading non-zeros can disappear. Thus, the number of relevant digits can decrease considerably. Loss of significance is impending in particular when both numbers are of similar magnitude.

Typically, the conditioning of a problem $p(x)$ having input x is not defined as before by computing a simple difference (i.e., the relative error), but via the derivative of the result with respect to the input:

$$\text{cond}(p(x)) := \frac{\partial p(x)}{\partial x} .$$

When a problem p is decomposed into two or more subproblems, then the chain rule implies

$$\text{cond}(p(x)) = \text{cond}(r(q(x))) = \frac{\partial r(z)}{\partial z} \Big|_{z=q(x)} \cdot \frac{\partial q(x)}{\partial x} .$$

Naturally, the total conditioning of $p(x)$ is independent of the decomposition, but the partial conditionings depend on the decomposition. This can lead to problems: Let p be well-conditioned with an excellently conditioned first part q and a miserably conditioned second part r . If there then occur errors in the first part, these can have catastrophic effects in the second.

A prime example for conditioning is the computation of the intersecting point of two non-parallel lines. If the two lines are almost orthogonal, then the problem of finding the intersection is well-conditioned. However, if they are almost parallel, then we have an ill conditioned problem.

Stability

Using the notion of conditioning allows us to characterize problems. Next, we turn to the characterization of numerical algorithms. As we have readily seen, the input data may be perturbed. Mathematically formulated, this means that they are only determined up to a certain tolerance, e.g., they lie in some neighborhood,

$$U_\varepsilon(x) := \{\tilde{x} : |\tilde{x} - x| < \varepsilon\}$$

with respect to the exact input x . Every such \tilde{x} must therefore be considered as equivalent to x . Thus, the following definition appears obvious: An approximation \tilde{y} for $y = p(x)$ is called *acceptable* if \tilde{y} is the exact solution for one of the above \tilde{x} , i.e.,

$$\tilde{y} = p(\tilde{x}).$$

Here, the error $\tilde{y} - y$ has different sources: Round-off errors as well as method or discretization errors (series and integrals are often approximated by sums, derivatives by difference quotients, iterations are aborted after a few iteration steps).

Stability is another central notion of numerics. Here, a numerical algorithm is called (*numerically*) *stable*, if it produces acceptable results under the influence of round-off and method errors for all admissible and up to the computational accuracy perturbed input data. A stable algorithm can, by all means, yield large errors—for example, if the problem to be solved is ill-conditioned. The established implementations of the basic arithmetic operations are numerically stable. The composition of stable methods, however, are not necessarily stable—otherwise everything would be numerically stable.

Stability is an important and central topic for methods involving the numerical solution of ordinary and partial differential equations.

2.4.2 Interpolation and Quadrature

Interpolation and *integration* or *quadrature*, are central tasks of numerical analysis. In subsequent chapters on modeling and simulation they are required indirectly at most and as such we will make it brief in the following.

Polynomial Interpolation

For reasons of simplicity we restrict ourselves to the one-dimensional case. In interpolation or intermediate value calculations, an (entirely or partially unknown or just too complicated) function $f(x)$ is to be replaced by a function $p(x)$ that is easy to construct and to work with (evaluate, differentiate, integrate), whereby p

assumes prescribed *node values* $y_i = f(x_i)$ at the given *nodes* $x_i, i = 0, \dots, n$, which define the distance of the *mesh widths* $h_i := x_{i+1} - x_i$. The pairs (x_i, y_i) are called nodal *data points*, while p is called the *interpolant* of f . In view of their simple structure, a class of particularly popular interpolants is given by *polynomials* $p \in \mathbb{P}_n$, the vector space of all polynomials with real coefficients of degree smaller than or equal to n in the single variable x . However, polynomial interpolation is by no means the only one: One can patch together piecewise polynomials from which one obtains the so-called *polynomial splines* which offer several distinct and essential advantages, and one can also interpolate using rational, trigonometric or exponential functions.

If one uses the interpolant p instead of the function f between the nodes, then one commits an *interpolation error* there. The difference $f(x) - p(x)$ is called the *error term* or *remainder*, and it holds that,

$$f(x) - p(x) = \frac{D^{n+1}f(\xi)}{(n+1)!} \cdot \prod_{i=0}^n (x - x_i)$$

at some intermediate location ξ ,

$$\xi \in [\min(x_0, \dots, x_n, x), \max(x_0, \dots, x_n, x)].$$

In the case of sufficiently smooth functions f , this relationship allows for the estimation of the interpolation error.

There exist different possibilities for the construction and representation of polynomial interpolants: the classical approach of the *point* or *incidence test*, the approach of *Lagrange polynomials*,

$$L_k(x) := \prod_{i:i \neq k} \frac{x - x_i}{x_k - x_i}, \quad p(x) := \sum_{k=0}^n y_k \cdot L_k(x),$$

the recursive *scheme by Aitken and Neville* as well as the recursive *Newton interpolation formula*.

For equidistant nodes with mesh width $h := x_{i+1} - x_i$, one can easily estimate the interpolation error:

$$|f(\bar{x}) - p(\bar{x})| \leq \frac{\max_{[a,b]} |D^{n+1}f(x)|}{n+1} \cdot h^{n+1} = \mathcal{O}(h^{n+1}).$$

One observes that classical polynomial interpolation with equidistant nodes is very ill-conditioned for larger n (here, the impact of “larger” already begins below 10)—small errors in the central nodal values are, for example, amplified drastically at the boundary of the considered interval by polynomial interpolation. For this reason, one has to attend to methods that are better in this respect.

Polynomial Splines

In order to circumvent the two major disadvantages of the polynomial interpolation (the number of nodes and the degree of polynomial are tightly connected; useless for larger n), one “glues” together polynomial pieces of lower degree in order to also construct a global interpolant and for a larger number of nodes. This leads to *polynomial splines* or *splines* in short. Once again let $a = x_0 < x_1 < \dots < x_n = b$ and $m \in \mathbb{N}$. Here, the x_i are called *nodes*. We consider here only the special case of *simple nodes*, i.e., $x_i \neq x_j$ for $i \neq j$. A function $s : [a, b] \rightarrow \mathbb{R}$ is called a *spline of order m* or *of degree m - 1*, resp., if $s(x) = p_i(x)$ is $m - 2$ -times continuously differentiable on $[a, b]$ with $p_i \in \mathbb{P}_{m-1}$, $i = 0, 1, \dots, n - 1$ on $[x_i, x_{i+1}]$. Thus, s is a polynomial of degree $m - 1$ between two neighbouring nodes, and globally (i.e., in particular at the nodes themselves!) s is $m - 2$ -times continuously differentiable.

For $m = 1$, s is a step function (piecewise constant) while for $m = 2$ it is piecewise linear, etc. The *cubic splines* ($m = 4$) are popular. Splines not only overcome the disadvantages of polynomial interpolation, they can also be constructed efficiently (linear effort in n).

Trigonometric Interpolation

With *trigonometric interpolation*, one considers complex-valued functions that are defined on the unit circle in the complex number plane—and also refers to the *representation in the frequency domain*. Let there be n nodes given on the unit circle of the complex number plane,

$$z_j := e^{\frac{2\pi i}{n}j}, \quad j = 0, 1, \dots, n - 1,$$

as well as n node values v_j . One desires to find the interpolant

$$p(z), \quad z = e^{2\pi i t}, t \in [0, 1],$$

with

$$p(z_j) = v_j, \quad j = 0, 1, \dots, n - 1, \quad p(z) = \sum_{k=0}^{n-1} c_k z^k = \sum_{k=0}^{n-1} c_k e^{2\pi i k t}.$$

$p(z)$ is hence set up as a linear combination of exponential functions or—after separation into real and imaginary part—of sine and cosine functions. Finding this *p de facto* implies the need to compute the coefficients c_k , and these are of course nothing else but the coefficients of the (*discrete*) *Fourier transformation (DFT)*. A famous and efficient algorithm for this task is the *Fast Fourier Transform (FFT)*.

With the previously introduced p and the abbreviation $\omega := e^{2\pi i/n}$ we obtain the following problem setting: Find n complex numbers c_0, \dots, c_{n-1} which satisfy

$$v_j = p(\omega^j) = \sum_{k=0}^{n-1} c_k \omega^{jk} \quad \text{for } j = 0, 1, \dots, n-1.$$

With some analysis ($\bar{\omega}$ the complex conjugation of ω) one can show that

$$c_k = \frac{1}{n} \sum_{j=0}^{n-1} v_j \bar{\omega}^{jk} \quad \text{for } k = 0, 1, \dots, n-1.$$

We use c and v to denote the n -dimensional vectors of the discrete Fourier coefficients and the DFT input, resp. Furthermore, we let the matrix M be defined as $M = (\omega^{jk})_{0 \leq j, k \leq n-1}$. Hence, it holds using matrix-vector notation that

$$v = M \cdot c, \quad c = \frac{1}{n} \cdot \overline{M} \cdot v.$$

This formula for the computation of the coefficients c_k defines precisely the *discrete Fourier transformation (DFT)* of the input data v_k . The formula for the computation of the values v_j from the Fourier coefficients c_k is called the *inverse discrete Fourier transformation (IDFT)*.

Apparently, the numbers of required arithmetic operations for the DFT and IDFT are of the order $\mathcal{O}(n^2)$.

The FFT algorithm, however, gets by with $\mathcal{O}(n \log n)$ operations for suitable n . The main idea is a recursive reordering of the coefficients into even and odd indices with subsequent exploitation of the recurrence of certain partial sums (*sorting phase* and *combination phase* by means of the so-called *butterfly operation*).

Close relatives, the *discrete cosine transformation* and the *fast cosine transformation*, are used, for example, in JPEG methods for image compression.

Numerical Quadrature

Numerical quadrature is known to be the numerical computation of a certain integral of the type

$$I(f) := \int_{\Omega} f(\mathbf{x}) d\mathbf{x}$$

for a given function $f : \mathbb{R}^d \supseteq \Omega \rightarrow \mathbb{R}$, the *integrand*, and a given *integration domain* Ω . Here, we will exclusively be concerned with *univariate* quadrature, i.e., with the case $d = 1$ over an interval $\Omega = [a, b]$. Obviously, the greater challenges lie in the higher-dimensional case of the *multivariate* quadrature,

occurring for example, in statistics, physics or in mathematical finance and requires sophisticated numerical techniques. Numerical quadrature should only be used if all other solution or simplification techniques such as closed-form formulas or subdivision of the integration domain have failed. Most of the numerical integration techniques require sufficient smoothness (differentiability) of the integrand.

Almost all *quadrature rules*, i.e., rules for numerical quadrature, can be written as *weighted sums of function values (samples)*:

$$I(f) \approx Q(f) := \sum_{i=0}^n g_i f(x_i) =: \sum_{i=0}^n g_i y_i$$

with *weights* g_i and pairwise distinct *nodes* x_i , where $a \leq x_0 < x_1 < \dots < x_{n-1} < x_n \leq b$. Since the evaluation of the integrand is often an expensive endeavor, one is interested in rules which allow for high accuracy (a small *quadrature error*) given moderate n .

The standard approach for the derivation of quadrature rules is to replace the integrand f by an easy to construct and easy to integrate approximation function \tilde{f} which is then integrated *exactly*, i.e.,

$$Q(f) := \int_a^b \tilde{f}(x) dx .$$

For simplicity, the approximant \tilde{f} is frequently chosen as a *polynomial interpolant* $p(x)$ of $f(x)$ for the nodes x_i . In this case, the representation of $p(x)$ using the Lagrange polynomial $L_i(x)$ of degree n yields the weights g_i practically without cost:

$$Q(f) := \int_a^b p(x) dx = \int_a^b \sum_{i=0}^n y_i L_i(x) dx = \sum_{i=0}^n \left(y_i \cdot \int_a^b L_i(x) dx \right) ,$$

where the weights g_i are given explicitly through,

$$g_i := \int_a^b L_i(x) dx .$$

The integrals of the Lagrange polynomial clearly can be computed a priori—they depend on the chosen grid (the nodes), but not on the integrand f . As a result of the uniqueness of the interpolation problem it holds that $\sum_{i=0}^n L_i(x) \equiv 1$ and therefore $\sum_{i=0}^n g_i = b - a$. This implies that the sum of the weights always equals $b - a$ if a polynomial interpolant is used for the quadrature. For reasons of conditioning, one in general considers only rules with positive weights.

Simple Quadrature Rules

One distinguishes between *simple* and *composite* quadrature rules. A simple rule treats the entire integration domain $[a, b]$ of length $H := b - a$ in its entirety. A composite rule, however, decomposes the integration domain into subdomains in which simple rules are then applied, their individual sum then forms the overall approximation—a procedure that strongly resembles the spline interpolation.

The *Newton–Cotes-Formula* represent an important class of simple rules:

$$Q_{NC(n)}(f) := I(p_n),$$

where p_n is the polynomial interpolant of f of degree n for the $n + 1$ equidistant nodes $x_i := a + H \cdot i/n$, $i = 0, \dots, n$. The simplest representative is the *rectangle rule*

$$Q_R(f) := H \cdot f\left(\frac{a+b}{2}\right) = I(p_0).$$

For its *remainder term* $R_R(f) := Q_R(f) - I(f)$ one can show the relationship,

$$R_R(f) := -H^3 \cdot \frac{f^{(2)}(\xi)}{24}$$

for some intermediate value $\xi \in]a, b[$ if f is twice continuously differentiable on $]a, b[$. Hence, polynomials of degree 0 or 1 is integrated *exactly*.

If one replaces the constant polynomial interpolant p_0 by its linear counterpart p_1 , then one obtains the *trapezoidal rule*

$$Q_T(f) := H \cdot \frac{f(a) + f(b)}{2} = I(p_1)$$

with remainder,

$$R_T(f) = H^3 \cdot \frac{f^{(2)}(\xi)}{12}.$$

The polynomial of maximal degree that can be integrated exactly by a quadrature rule is called the *degree of accuracy*, or in short, the *accuracy* of the method. Hence, the rectangle rule as well as the trapezoidal rule have accuracy 1. For $p = 2$, one obtains *Simpson's rule*

$$Q_F(f) := H \cdot \frac{f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)}{6} = I(p_2)$$

with remainder

$$R_F(f) = H^5 \cdot \frac{f^{(4)}(\xi)}{2880}.$$

Assuming sufficiently high differentiability, one therefore obtains in the Newton–Cotes approach methods of higher order and higher accuracy for increasing n . However, for $n = 8$ and $n \geq 10$, there appear negative weights. As previously mentioned, the Newton–Cotes formulas become practically useless in these cases.

As a matter of principle, the problem existing for negative weights g_i for higher polynomial degree n does not imply that polynomial interpolants of higher degree are unsuitable for purposes of numerical quadrature. A possible remedy requires a deviation from the assumption that the nodes are equidistant. This is exactly what is rooted in the principle of the *Clenshaw–Curtis rules* in which the integration interval $[a, b]$ is now replaced by a uniform subdivision of the angles over the semi circle $[0, \pi]$.

As one can easily show, the nodes at the boundaries of the integration interval lie closer to each other compared to the nodes in the middle. The use of such guidelines in the construction of quadrature rules is beneficial since the weights appearing are always positive.

Composite Quadrature Rules

The most important composite rule is the *composite trapezoidal rule*. The integration interval $[a, b]$ is first subdivided into n subintervals of length $h := (b - a)/n$. The equidistant points $x_i := a + ih$, $i = 0, \dots, n$ serve as nodes. The trapezoidal rule is then applied on each subinterval $[x_i, x_{i+1}]$. Finally, each contribution of the computed integral values are added to produce the integral value over the original interval:

$$Q_{\text{ts}}(f; h) := h \cdot \left(\frac{f_0}{2} + f_1 + f_2 + \dots + f_{n-1} + \frac{f_n}{2} \right),$$

where $f_i := f(x_i)$. It holds that the trapezoidal sum has remainder,

$$R_{\text{ts}}(f; h) = h^2 \cdot (b - a) \cdot \frac{f^{(2)}(\xi)}{12} = h^2 \cdot H \cdot \frac{f^{(2)}(\xi)}{12}.$$

Compared to the trapezoidal rule, the accuracy remains 1 while the order is reduced to 2 (an order of magnitude is lost by the summation). One gains, however, a rule that is easy to implement for which one can arbitrarily increase n without encountering numerical difficulties.

Just as the composite trapezoidal rule represents a composite quadrature rule based upon the trapezoidal rule, the *Composite Simpson Rule* is the natural

generalization of Simpson's rule. Starting from the same subdivision of the integration interval $[a, b]$ as before, one now applies Simpson's rule over two combined neighboring subintervals:

$$Q_{ss}(f; h) := \frac{h}{3} \cdot (f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \dots + 2f_{n-2} + 4f_{n-1} + f_n).$$

Further Approaches for Numerical Quadrature

At this junction, let us refer to further important approaches for numerical quadrature. In *extrapolation*, one combines different computed approximations of lower order (e.g., trapezoidal sums for h , $h/2$, $h/4$, etc.) in a clever linear way in order to eliminate certain error terms thus resulting in a significantly better approximation. However, a prerequisite for the extrapolation technique is a high degree of smoothness for the integrand. Additional terms that are important in this context are the *Euler-Maclaurin sum formula* as well as *Romberg quadrature*.

For *Monte-Carlo quadrature*, a stochastic approach is applied. It is, in particular, very popular in the high-dimensional case where classical numerical methods often fail because of the so-called *curse of dimensionality* (a product approach making use of a rule with only 2 points in one dimension already requires 2^d points in d spatial dimensions). Colloquially, one can visualize the process as one in which nodes in the integration domain are selected from some uniform distribution. This defines the nodes where the integrand is evaluated, and simple averaging yields the overall result (obviously under consideration of the size of the domain). The error of the Monte-Carlo quadrature does not depend on the dimensionality; however, the convergence behavior in terms of the number of nodes n is only $\mathcal{O}(1/\sqrt{n})$.

Quasi Monte-Carlo methods or *Methods of minimal discrepancy* move in the same direction, however, they employ suitable sequences of deterministic nodes instead of randomly selected nodes.

The idea behind *Gauß quadrature* consists of placing the nodes so that polynomials of degree as high as possible can still be integrated exactly—i.e., one aims for a degree of accuracy as high as possible:

$$I(p) = \int_{-1}^1 p(x) dx \stackrel{!}{=} \sum_{i=1}^n g_i p(x_i)$$

for all $p \in \mathbb{P}_k$ with k as large as possible. One keeps a series expansion for f at the back of one's mind in which as many leading terms as possible shall be integrated exactly (high error order for decreasing interval width). The Gauß quadrature, which employs the roots of Legendre polynomial as nodes, attains the (hereby largest possible) accuracy degree $2n - 1$.

A very old and yet—especially considered from an algorithmic point of view—elegant approach for numerical quadrature originates from Archimedes. He is one of

the forefathers of the algorithmic paradigm *divide et impera* ubiquitous in computer science. The area under the integrand is thereby exhausted through a sequence of hierarchical (i.e., smaller and smaller) triangles. Similar to the Monte-Carlo quadrature, the main attraction of this approach is rooted in the highdimensional integrals. Here, competitive *sparse grid* algorithms can be formulated.

2.4.3 Direct Solution of Linear Systems of Equations

Linear Systems of Equations

Another important application area for numerical methods is *numerical linear algebra* that is concerned with the numerical solution of problems in linear algebra (matrix-vector product, computation of eigenvalues, solution of linear systems of equations). Of central importance here is the *solution of systems of linear equations*, i.e., for $A = (a_{i,j})_{1 \leq i,j \leq n} \in \mathbb{R}^{n,n}$ and $b = (b_i)_{1 \leq i \leq n} \in \mathbb{R}^n$, find $x \in \mathbb{R}^n$ such that $Ax = b$, which, among other applications, are generated in the discretization of models based on differential equations. One distinguishes between *fully populated* matrices (the number of non-zeros in A is of the order of the total number of matrix entries, i.e., $\mathcal{O}(n^2)$) and *sparse* matrices (typically $\mathcal{O}(n)$ or $\mathcal{O}(n \log(n))$ non-zeros). Often, sparse matrices have a *sparsity structure* (diagonal matrices, tridiagonal matrices, general banded structure), which simplifies the solving of the system.

With respect to solution techniques, one distinguishes between *direct* solvers, which yield the exact solution x (modulo round-off error), from *indirect* solvers which, starting from an initial approximation $x^{(0)}$, *iteratively* compute a (hopefully convergent) sequence of approximations $x^{(i)}$ without actually reaching x .

We first address the direct solvers; let the matrix A be non-singular. The explicit computation of the inverse A^{-1} is not performed due to complexity reasons. For the analysis of the conditioning of the problem $Ax = b$, as well as for the analysis of the convergence behavior of iterative methods, one needs the notion of the *condition number* $\kappa(A)$:

$$\|A\| := \max_{\|x\|=1} \|Ax\|, \quad \kappa(A) := \|A\| \cdot \|A^{-1}\|,$$

where $\|\cdot\|$ denotes a suitable vector or matrix norm, resp. One can show that the following inequality holds,

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{2\varepsilon\kappa(A)}{1 - \varepsilon\kappa(A)}$$

where ε denotes an upper bound for the relative input perturbations $\delta A/A$ or $\delta b/b$, resp. The larger the condition number $\kappa(A)$, the larger our upper bound on the right becomes for estimating the impact on the result, and therefore the worse

the conditioning of the problem “solve $Ax = b$ ”. The term “condition number” is therefore a meaningful choice—it represents a measure for the conditioning. Only if $\varepsilon\kappa(A) \ll 1$, which represents a restriction to the order of magnitude of the admissible input perturbation, can the numerical solution of the problem make sense. We then would have the conditioning under control.

An important quantity is the *residual* r . For the approximation \tilde{x} of x , r is defined as

$$r := b - A\tilde{x} = A(x - \tilde{x}) =: -Ae$$

with *error* $e := \tilde{x} - x$. Error and residual can be of very different orders of magnitude. In particular, a small residual does not necessarily imply a small error—in fact, the correlation also involves the condition number $\kappa(A)$. Nevertheless, the residual is helpful: $r = b - A\tilde{x} \Leftrightarrow A\tilde{x} = b - r$ shows that an approximate solution with a small residual represents an acceptable result.

Gaussian Elimination and the *LU* Decomposition

From linear algebra comes the well-known classical method for solving linear systems of equations, *Gaussian elimination*, and it is the natural generalization of solving two equations in two unknowns: Solve one of the n equations (say, the first) for an unknown (say, x_1); substitute the resulting (and dependent on x_2, \dots, x_n) expression for x_1 in the other $n - 1$ equations— x_1 is thus *eliminated* from these; solve the resulting system of $n - 1$ equations in the remaining $n - 1$ unknowns correspondingly and continue until there remains only x_n in a single equation which can therefore be explicitly computed; now plug in x_n in the elimination equation for x_{n-1} so that one obtains x_{n-1} explicitly; continue until finally the elimination equation of x_1 yields the value for x_1 by substituting the (known) values for x_2, \dots, x_n . Visually, the elimination of x_1 means that A and b are modified such that the first column contains only zeros underneath $a_{1,1}$, where the new system (consisting of the first equation and the remaining x_1 -free equations) is naturally solved by the same vector x as the old one!

Gaussian elimination is equivalent to the so-called *LU decomposition*, in which A is *faktored* into the product $A = LU$ with unit lower triangular matrix L (ones along the diagonal) and upper triangular matrix U . In the special case of a positive definite A , the *Cholesky decomposition* yields a symmetric factorization $A = \tilde{L} \cdot \tilde{L}^T$ which saves about half the cost. The complexity is cubic in the number of unknowns in all three cases.

For Gaussian elimination and *LU* decomposition, one divides by the diagonal elements (the so-called *pivots*), which may possibly be zero. If a zero occurs, one has to modify the algorithm to enforce an admissible situation, i.e., a non-zero on the diagonal, by row or column exchange (which is of course possible if A is non-singular). Possible exchange partners for a zero on the diagonal can either be found in column i below the diagonal (*column pivoting*) or in the entire remaining part of

the matrix (everything on or after row and column $i + 1$, *complete pivoting*). Finally: Even if no zeros occur—for numerical reasons, pivoting is always advisable.

2.4.4 Iteration Methods

Notion of Iteration

Linear systems of equations which need to be solved numerically often originate in the discretization of ordinary (for boundary value problems) or partial differential equations. The direct solution methods previously discussed are generally out of the question. First, n is usually so large (in general, n is directly correlated with the number of grid points and this leads to very large n , in particular, for unsteady partial differential equations (three spatial variables and one temporal variable)) that a cubic effort is unacceptable. Second, such matrices are typically sparse and exhibit a certain structure which naturally corresponds to a decreasing effect on storage and computational time; elimination methods typically destroy this special structure and thus reverse the advantages. Furthermore, the accuracy provided from the exact direct solution often exceeds that required for the simulation.

Therefore, *iterative* methods are preferred for large and sparse matrices or linear systems of equations. These start with (in general, not just for the situation of linear systems of equations) an *initial approximation* $x^{(0)}$ and generate a sequence of approximation $x^{(i)}$, $i = 1, 2, \dots$ which, in the case of convergence, converge toward the exact solution x . In this context one also refers to the *method function* Φ of the iteration, $\Phi(x_i) := x_{i+1}$, which determines the new iteration value from the current one. In the case of convergence ($\lim x_i = x$), the iteration has a *fixed point* $\Phi(x) = x$. For sparse matrices, an iteration step typically costs (at least) $\mathcal{O}(n)$ computational operations. Thus, the construction of iterative algorithms is affected by how many iteration steps will be required to reach a certain given accuracy.

Relaxation Methods

Perhaps the oldest iterative methods for the solution of linear systems of equations $Ax = b$ with $A \in \mathbb{R}^{n,n}$ and $x, b \in \mathbb{R}^n$ are the so-called *relaxation methods*: the *Richardson* method, the *Jacobi* method, the *Gauß–Seidel* method as well as the *overrelaxation (SOR)* method. The starting point for each of these methods is the residual $r^{(i)} := b - Ax^{(i)} = -Ae^{(i)}$. Since $e^{(i)}$ is unavailable (the error cannot be computed without knowledge of the exact solution x), it is reasoned by the above relationship to make use of the vector $r^{(i)}$ as the *direction* in which we want to search for an improvement of $x^{(i)}$. The Richardson method takes the residual directly as a correction for $x^{(i)}$. The Jacobi- and Gauß–Seidel methods try a little harder. Their idea for the correction of the k -th component of $x^{(i)}$ is the elimination of $r_k^{(i)}$.

The SOR method and its counterpart, *damped* relaxation, additionally account for the fact that such a correction often overshoots or undershoots its mark, respectively.

In the algorithmic formulation, the four methods are represented as follows:

- *Richardson iteration:*

for $i = 0, 1, \dots$

$$\quad \text{for } k = 1, \dots, n: \quad x_k^{(i+1)} := x_k^{(i)} + r_k^{(i)}$$

Here, the residual $r^{(i)}$ is simply used componentwise as a correction to the current approximation $x^{(i)}$.

- *Jacobi iteration:*

for $i = 0, 1, \dots$

$$\quad \text{for } k = 1, \dots, n: \quad y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}$$

$$\quad \text{for } k = 1, \dots, n: \quad x_k^{(i+1)} := x_k^{(i)} + y_k$$

In every partial step k of step i , a correction y_k is computed and stored. If applied immediately, this would lead to the (temporary) disappearance of the k -component of the residual $r^{(i)}$ (easily verified by insertion). Equation k would be solved exactly with this current approximation for x —a progress which would be immediately lost again in the subsequent partial step $k + 1$. However, these component corrections are not performed immediately but only at the end of an iteration step (second k -loop).

- *Gauß-Seidel iteration:*

for $i = 0, 1, \dots$

$$\quad \text{for } k = 1, \dots, n: \quad r_k^{(i)} := b_k - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k}^n a_{kj} x_j^{(i)}$$

$$\quad y_k := \frac{1}{a_{kk}} \cdot r_k^{(i)}, \quad x_k^{(i+1)} := x_k^{(i)} + y_k$$

Here, the same correction as in the Jacobi method is computed, however, the update is now always performed immediately and not only at the end of the iteration step. Thus, at the update of component k , the modified new values for the components 1 to $k - 1$ are already available.

- Sometimes, in each of the three sketched methods a *damping* (multiplication of the correction with a factor $0 < \alpha < 1$) or an *overrelaxation* (factor $1 < \alpha < 2$), resp., leads to better convergence behavior:

$$x_k^{(i+1)} := x_k^{(i)} + \alpha y_k .$$

In the Gauß-Seidel case, the version with $\alpha > 1$ is predominantly used which is referred to as the *SOR method*, in the Jacobi case, on the other side, damping is mostly used.

For a brief convergence analysis of the above methods we need an algebraic formulation (instead of the algorithmic). All the approaches presented are based on the simple idea of writing the matrix A as a sum $A = M + (A - M)$, where $Mx = b$ is very easy to solve and the difference $A - M$ should not be too

large wrt. a matrix norm. By means of such a suitable M , the Richardson, Jacobi, Gauß–Seidel- and SOR methods can be written as

$$Mx^{(i+1)} + (A - M)x^{(i)} = b$$

or, solved for $x^{(i+1)}$, as

$$x^{(i+1)} := x^{(i)} + M^{-1}r^{(i)}.$$

Furthermore, we decompose A additively into its diagonal part D_A , its strictly lower triangular part L_A as well as its strictly upper triangular part U_A :

$$A =: L_A + D_A + U_A.$$

Thus we can show the following relationships:

- Richardson: $M := I$,
- Jacobi: $M := D_A$,
- Gauß–Seidel: $M := D_A + L_A$,
- SOR: $M := \frac{1}{\alpha}D_A + L_A$.

Concerning the convergence, there exists a direct consequence from the approach $Mx^{(i+1)} + (A - M)x^{(i)} = b$: If the sequence $(x^{(i)})$ converges, then the limit is the exact solution x of our system $Ax = b$. For the analysis, let us further assume that the *iteration matrix* $-M^{-1}(A - M)$ (i.e., the matrix that is applied to $e^{(i)}$ in order to obtain $e^{(i+1)}$) is symmetric. Then the spectral radius ϱ (i.e., the absolute value of the largest eigenvalue) is the critical quantity for the determination of the convergence behavior:

$$\left(\forall x^{(0)} \in \mathbb{R}^n : \lim_{i \rightarrow \infty} x^{(i)} = x = A^{-1}b \right) \Leftrightarrow \varrho < 1.$$

To see this, one subtracts $Mx + (A - M)x = b$ from the above equation of the general approach:

$$Me^{(i+1)} + (A - M)e^{(i)} = 0 \Leftrightarrow e^{(i+1)} = -M^{-1}(A - M)e^{(i)}.$$

If all eigenvalues are less than 1 in absolute value and hence $\varrho < 1$ holds, then all error components are reduced at each iteration step. For the case that $\varrho > 1$, generally at least one error component will increase. It is then natural, even intuitive, that the objective of the construction of iterative methods is a spectral radius of the iteration matrix as small as possible (as close to zero as possible).

There are a number of results regarding the convergence of the various methods, of which a few of the more significant ones shall be mentioned below:

- Necessary for the convergence of the SOR method is $0 < \alpha < 2$.
- If A is positive definite, then the SOR method (for $0 < \alpha < 2$) as well as the Gauß–Seidel iteration converge.

- If both A and $2D_A - A$ are positive definite, then the Jacobi method converges.
- If A is strictly diagonally dominant (i.e., $a_{ii} > \sum_{j \neq i} |a_{ij}|$ for all i), then the Jacobi and the Gauß–Seidel methods converge.
- In certain cases the optimal parameter ϱ can be determined (ϱ minimal so that the error reduction per iteration step becomes maximal).

Obviously, ϱ is not only critical for the question whether the iteration directive converges at all, but also for its quality, i.e. its convergence speed: The smaller ϱ , the faster are all the components of the error $e^{(i)}$ reduced in each iteration step. In practice, the above results are unfortunately of rather theoretical value since ϱ is often so close to 1 so that—despite convergence—the number of iteration steps required to reach a sufficient accuracy is far too large. An important example scenario is the discretization of partial differential equations. Here it is typical that ϱ depends on the problem size n and thus on the resolution h of the underlying grid, i.e. for example

$$\varrho = \mathcal{O}(1 - h_l^2) = \mathcal{O}\left(1 - \frac{1}{4^l}\right)$$

for a mesh width $h_l = 2^{-l}$. This is an enormous disadvantage: The finer and hence more accurate our grid, the more miserable the resulting convergence behavior becomes for our iterative methods. It is therefore a must to develop faster iterative solvers such as *multigrid methods*, for example.

Minimization Techniques

One of the best-known methods for the solution of linear systems of equations, the method of *conjugate gradients* (*cg*), is based on a different principle than that of relaxation. In order to see this, we take an indirect approach for the problem of solving linear systems of equations. In the following, let $A \in \mathbb{R}^{n,n}$ be symmetric and positive definite. In this case, the solution of $Ax = b$ is equivalent to the minimization of the quadratic function

$$f(x) := \frac{1}{2}x^T Ax - b^T x + c$$

for an arbitrary scalar constant $c \in \mathbb{R}$. Since A is positive definite, the hypersurface that is built from $z := f(x)$ is a paraboloid in \mathbb{R}^{n+1} with n -dimensional ellipsoids as isosurfaces $f(x) = \text{const.}$, with f having a global minimum at x . The equivalence of the problems is obvious:

$$f'(x) = \frac{1}{2}A^T x + \frac{1}{2}Ax - b = Ax - b = -r(x) = 0 \Leftrightarrow Ax = b .$$

By means of this reformulation, one can now employ *optimization methods* and thus extend the spectrum of potential solution techniques. Let us consider techniques for searching for a minimum. A straightforward possibility is provided by the *method of steepest descent*: For $i = 0, 1, \dots$, repeat,

$$\begin{aligned} r^{(i)} &:= b - Ax^{(i)}, \\ \alpha_i &:= \frac{r^{(i)T} r^{(i)}}{r^{(i)T} Ar^{(i)}}, \\ x^{(i+1)} &:= x^{(i)} + \alpha_i r^{(i)}, \end{aligned}$$

or begin with $r^{(0)} := b - Ax^{(0)}$ and repeat for $i = 0, 1, \dots$

$$\begin{aligned} \alpha_i &:= \frac{r^{(i)T} r^{(i)}}{r^{(i)T} Ar^{(i)}}, \\ x^{(i+1)} &:= x^{(i)} + \alpha_i r^{(i)}, \\ r^{(i+1)} &:= r^{(i)} - \alpha_i Ar^{(i)}, \end{aligned}$$

which saves one of the two matrix-vector products (the only really expensive step in the algorithm). The method of steepest descent searches for an improvement in the direction of the negative gradient $-f'(x^{(i)}) = r^{(i)}$ which in fact points to the steepest descent (thus the name). It would obviously be better to search in the direction of the error $e^{(i)} := x^{(i)} - x$, but unfortunately this is not known. But the direction itself is insufficient for one also needs a suitable step length. To this end, we look for the minimum of $f(x^{(i)} + \alpha_i r^{(i)})$ when interpreted as a function of α_i (partial derivative with respect to α_i is set to zero), which after a brief calculation yields the above value for α_i . If, instead of the residuals $r^{(i)}$, one chooses alternatingly the unit vectors in the coordinate directions as search vectors and then once again determines the optimal step widths with respect to these search directions, one actually ends up with the Gauß–Seidel iteration. Thus, despite all differences there are relationships between the relaxation and minimization approach.

The convergence behavior of the method of steepest descent is modest. One of the few trivial special cases is the identity matrix: There, the isosurfaces are spheres, the gradient always points to the center (the minimum), and one reaches the goal in one step! In general, one eventually lands arbitrarily close to the minimum, however, this may take arbitrarily long (since we can always destroy part of what we have previously attained). In order to alleviate this disadvantage we stick with our minimization approach but in addition we continue to look for better search directions. If all search directions would be orthogonal, and if the error after i steps would be orthogonal to all previous search directions, then the optimality previously attained would never be lost and one would be at the minimum after at most n steps—just as for the case of a direct solver. For this reason one

calls the cg-method and cg-based methods *semi-iterative methods*. One obtains the cg-method if one pursues the idea of improved search directions, in particular employs *conjugate directions* (two vectors x and y are called *A-orthogonal* or *conjugate* if $x^T A y = 0$ holds) and combines it all with an inexpensive strategy for computing such conjugate directions. The convergence behavior is clearly better than for the steepest descent, but the “slow-down” effect for increasing problem size is still not overcome.

Nonlinear Equations

This much about linear solvers: Many realistic models, unfortunately, are not linear. Thus, one has to also contemplate the solution of *nonlinear equations*. Typically, this can no longer be done directly, but only iteratively. Here in the following, we restrict ourselves to the easier case in which $n = 1$ (i.e., *one* nonlinear equation). Therefore, consider a continuously differentiable (nonlinear) function $f : \mathbb{R} \rightarrow \mathbb{R}$ which has a root $\bar{x} \in]a, b[$ (one may think of a search for roots or extrema). Consider an iteration rule (to be determined) which yields a sequence of approximations, $(x^{(i)})$, $i = 0, 1, \dots$, that (hopefully) converge toward the simple root \bar{x} of $f(x)$. As a measure of the *convergence speed* one considers the reduction of the error in each step and, in the convergent case

$$|x^{(i+1)} - \bar{x}| \leq c \cdot |x^{(i)} - \bar{x}|^\alpha,$$

speaks of different types of convergence depending on the maximal possible value of the parameters α : *linear* ($\alpha = 1$ and in addition $0 < c < 1$) or *quadratic* ($\alpha = 2$) convergence etc. There exist *conditionally* or *locally* convergent methods for which convergence is guaranteed only for an already sufficiently good starting value $x^{(0)}$, and *unconditionally* or *globally* convergent methods in which the iteration leads to a root of f independent of the choice of the starting point.

The simple methods are the *bisection method* (start with $[c, d] \subseteq [a, b]$ with $f(c) \cdot f(d) \leq 0$ —continuity of f guarantees the existence of (at least) one root in $[c, d]$ —and cut each successive search interval in half), the *Regula falsi* (a variant of the bisection method in which the x-axis-intersection of the line through the two points $(c, f(c))$ and $(d, f(d))$ is selected instead of the interval midpoint as an end point of the new and smaller interval), as well as the *secant method* (begin with *two* initial approximations $x^{(0)}$ and $x^{(1)}$; in the following, one determines $x^{(i+1)}$ out of $x^{(i-1)}$ and $x^{(i)}$ by computing the root of the line through $(x^{(i-1)}, f(x^{(i-1)}))$ and $(x^{(i)}, f(x^{(i)}))$ (of the *secant s(x)*)).

The most famous method is the *Newton method*. Here one begins with *one* initial approximation $x^{(0)}$ and subsequently determines $x^{(i+1)}$ out of $x^{(i)}$ by searching for the root of the *tangent line* $t(x)$ of $f(x)$ at $x^{(i)}$ (*linearization*: replace f by its tangent or its Taylor polynomial of first degree, resp.):

$$\begin{aligned}
t(x) &:= f(x^{(i)}) + f'(x^{(i)}) \cdot (x - x^{(i)}) , \\
0 &= t(x^{(i+1)}) = f(x^{(i)}) + f'(x^{(i)}) \cdot (x^{(i+1)} - x^{(i)}) , \\
x^{(i+1)} &:= x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})} .
\end{aligned}$$

The *order of convergence* of the methods introduced is globally linear for bisection and regula falsi, locally quadratic for Newton and locally 1.618 for the secant method. However, since a Newton step requires an evaluation of f and f' , resp., it needs to be compared to *two* steps of the other methods. Furthermore, the computation of the derivative often poses a problem.

In most practical applications it holds that $n \gg 1$ (for example, the unknowns are the function values at the grid points!), so that we are confronted with very large nonlinear systems of equations. In the higher dimensional case, the simple derivative $f'(x)$ is replaced by the *Jacobi matrix* $F'(x)$, the matrix of partial derivatives of all vector components of F with respect to all variables. The Newton iteration rule then reads as

$$x^{(i+1)} := x^{(i)} - F'(x^{(i)})^{-1} F(x^{(i)}) ,$$

where obviously the matrix $F'(x^{(i)})$ is not inverted but the respective linear system of equations with the right hand side $F(x^{(i)})$ is solved (directly):

```

compute  $F'(x)$ ;
decompose  $F'(x) := LR$ ;
solve  $LRs = F(x)$ ;
update  $x := x - s$ ;
evaluate  $F(x)$ ;

```

The repeated computation of the Jacobi matrix is highly work-intensive, or is often only possible approximately, since most of the time differentiation must be performed numerically. In addition, the direct solution of a linear system of equations at each Newton step is expensive. Therefore, the Newton method has only been the starting point for a multitude of algorithmic developments. In the *Newton-Chord* or *Shamanskii method*, resp., the Jacobi matrix is not computed and inverted at every Newton step, but $F'(x^{(0)})$ is always used (Chord) or $F'(x^{(i)})$ is always used for several Newton steps (Shamanskii). In the *inexact Newton method*, the linear system of equations is not solved directly (i.e., exactly or via *LU* decomposition) in each Newton step, but iteratively; one speaks of an *inner* iteration within the (*outer*) Newton iteration. Finally, in the *Quasi-Newton method*, a sequence $B^{(i)}$ of approximations for $F'(\bar{x})$ is generated, and this is not done by expensive new computations but rather by inexpensive *updates*. One exploits the

fact that a *rank-1-update* ($B + uv^T$) with two arbitrary vectors $u, v \in \mathbb{R}^n$ (invertible if and only if $1 + v^T B^{-1}u \neq 0$) is easy to invert:

$$(B + uv^T)^{-1} = \left(I - \frac{(B^{-1}u)v^T}{1 + v^T B^{-1}u} \right) B^{-1}.$$

Broyden gave a suitable choice for u and v (s as above in the algorithm):

$$B^{(i+1)} := B^{(i)} + \frac{F(x^{(i+1)})s^T}{s^T s}.$$

2.4.5 Ordinary Differential Equations

Differential Equations

One of the most important application areas for numerical methods are *differential equations*, see also Sect. 2.2.2 on analysis. In *ordinary differential equations (ODE)*, whose numerical properties we want to discuss in the following, there is only one independent variable appearing. Simple applications include, for example, the oscillating pendulum

$$\ddot{y}(t) = -y(t)$$

having solution $y(t) = c_1 \cdot \sin(t) + c_2 \cdot \cos(t)$, or the exponential growth

$$\dot{y}(t) = y(t)$$

with the solution $y(t) = c \cdot e^t$. In *partial differential equations (PDE)*, there appear several independent variables. Here, simple application examples are the *Poisson equation* in 2 D, which, for example, describes the deformation u of a membrane which is fixed at the boundary under an outer load f :

$$\Delta u(x, y) := u_{xx}(x, y) + u_{yy}(x, y) = f(x, y) \quad \text{on } [0, 1]^2$$

(here, the explicit description of the solution is already becoming much harder) or the *heat equation* in 1 D, which, for example, describes the temperature distribution T in a metal wire with given temperature at the endpoints:

$$T_t(x, t) = T_{xx}(x, t) \quad \text{auf } [0, 1]^2$$

(here one has an *unsteady equation* due to the time dependency).

As it has already been mentioned in Sect. 2.2.2, the differential equation alone, in general, does not uniquely determine the solution, rather it requires additional

conditions. Such conditions appear as *initial conditions* (such as the population size at the beginning of the common era) or as *boundary conditions* (a space shuttle, after all, should start and land at well-defined locations). For such cases, one is looking for a function y that satisfies the differential equation *and* these conditions. Correspondingly, one speaks of *initial value problems (IVP)* or *boundary value problems (BVP)*. In this section, we will only consider IVPs of ODEs, in particular of the type

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0.$$

Here we require *one* initial condition since it is an ODE of *first order* (only the first derivative).

As an example we consider the ODE,

$$\dot{y}(t) = -\frac{1}{10}y(t) + \frac{1}{10}, \quad (2.1)$$

which, as a linear differential equation with constant coefficients, is one of the simplest conceivable differential equations of the above type. Its solutions can be stated directly as $y(t) = 1 + c \cdot e^{-t/10}$ with parameter $c \in \mathbb{R}$ as a degree of freedom which allows the flexibility to satisfy an initial condition $y(0) = 1 + c \stackrel{!}{=} y_0$. Figure 2.1 shows solution curves for different initial values y_0 .

By means of this simple example we can think about the ways to make at least qualitative statements pertaining to the solutions in complicated cases in which an explicit solution cannot be stated without further ado. Simultaneously, these considerations are the entry point to the numerical solution of ODEs for which reason this discussion is placed in the section on numerical properties instead of the section on analysis.

The idea is simple: We follow a particle in the $t - y$ -plane which moves along a curve $\{(t, y(t)), t \in \mathbb{R}_+\}$ where $y(t)$ satisfies the differential equation. Such a curve is called the *trajectory* or *path*. If we assume uniform motion with velocity 1 in t -direction, then the velocity in the y -direction has to be just $\dot{y} = f(t, y(t))$ in order to stay on the trajectory: For each point $(t, y(t))$, the right hand side of the differential equation gives us the direction in which to proceed. If one supplies the $t - y$ -plane with a *direction field*—i.e., at every point (t, y) with the vector with component 1 in t -direction and $f(t, y)$ in the y -direction, Fig. 2.1 shows some direction arrows for our example problem—then one can sketch a solution (or approximate it numerically) by exploiting the fact that at every point the direction vector is tangential to the curve.

An example of a qualitative property of a differential equation, such as all solutions converging to 1 for arbitrary initial values, could have been derived from the direction field even without knowledge of the explicit solution. The fact that the differential equation is *autonomous* due to its constant coefficients (the right hand side is of the form $f(y(t))$ without direct dependence of t), is also not necessary for such qualitative considerations (this will be different for the systems of differential

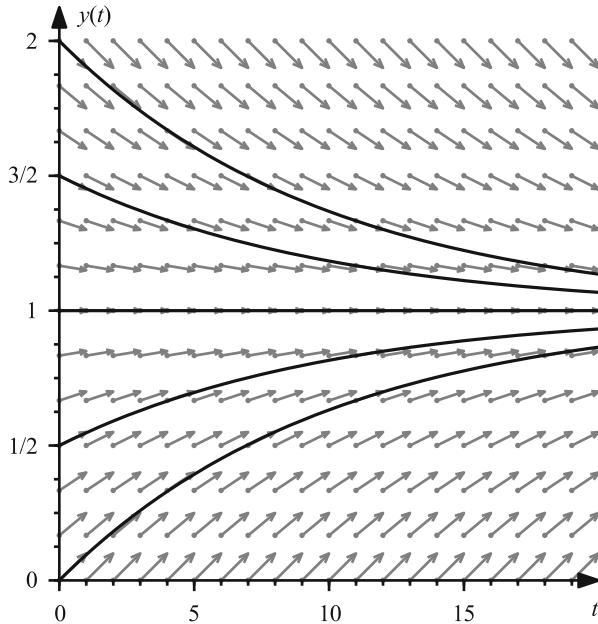


Fig. 2.1 Solutions and direction field of the differential equation (2.1)

equations which will be subsequently discussed). Altogether, direction fields are a useful tool to understand an ODE even before one attempts an explicit or numerical solution.

Finally, we will briefly turn to a very simple class of *systems of ODEs*, the *linear systems with constant coefficients*. As before, we distinguish the *homogeneous* case

$$\dot{x}(t) = A \cdot x(t), \quad x(t_0) = x_0,$$

as well as the *nonhomogeneous* case

$$\dot{x}(t) = A \cdot x(t) + c, \quad x(t_0) = x_0,$$

where $A \in \mathbb{R}^{n,n}$, $c \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$. For the homogeneous system, the approach $x(t) := e^{\lambda t} \cdot v$ with initially undetermined parameters $\lambda \in \mathbb{R}$ and $v \in \mathbb{R}^n$ yields after substitution into the ODE that there must hold $Av = \lambda v$. Thus, if one manages successfully to represent the initial vector x_0 as a linear combination of eigenvectors v_i associated with eigenvalues λ_i of A , $x_0 = \sum_{i=1}^n v_i$, then

$$x(t) := \sum_{i=1}^n e^{\lambda_i t} \cdot v_i$$

satisfies the ODE as well as the initial condition. Somewhat obtained as a freebie, we also learn that the eigenvalues of the system matrix A determine the behavior of the solution. In the case in which all eigenvalues have negative real parts, the solution will converge to zero over time; at least one positive real part leads to $\|x(t)\| \rightarrow \infty$ for $t \rightarrow \infty$; non-vanishing imaginary parts imply the presence of oscillations.

In the non-homogeneous case one chooses (assuming that the matrix A is regular) $x_\infty := -A^{-1}c$ and with one solution $z(t)$ of the homogeneous equation obtains the solution $x(t) := z(t) + x_\infty$ to the nonhomogeneous system because $\dot{x}(t) = \dot{z}(t) = A \cdot z(t) = A \cdot (x(t) + A^{-1}c) = A \cdot x(t) + c$. Compared to the homogeneous case, the solution is only shifted by x_∞ , which has no influence on the convergence, divergence or oscillation behavior, resp.

As an example we consider the following system:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} -1/10 & 1/20 \\ 1/20 & -1/10 \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} + \begin{pmatrix} 3/40 \\ 0 \end{pmatrix}. \quad (2.2)$$

The system matrix has the eigenvectors

$$v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

associated with the eigenvalue $\lambda_1 = -0.05$ and

$$v_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

associated with the eigenvalue $\lambda_2 = -0.15$. Since both eigenvalues are real and negative, the solutions thus converge to the *equilibrium point*

$$\bar{x} = -A^{-1} \begin{pmatrix} 3/40 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \end{pmatrix}$$

for arbitrary initial values. Figure 2.2 shows several solution curves in the $x - y$ -plane for arbitrary initial values, while Fig. 2.3 shows the components $x(t)$, $y(t)$ of the solution and their dependence on t . The direction field can still be drawn in the case of autonomous systems of differential equations with two components if one imagines the t -axis (which makes no contribution to the direction field) to be perpendicular to the drawing plane and then marks the direction vector (\dot{x}, \dot{y}) for a point in the $x - y$ -plane.

A completely different behavior is displayed by the solution curves of the system

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} -1/20 & 1/10 \\ 1/10 & -1/20 \end{pmatrix} \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} + \begin{pmatrix} 0 \\ -3/40 \end{pmatrix}, \quad (2.3)$$

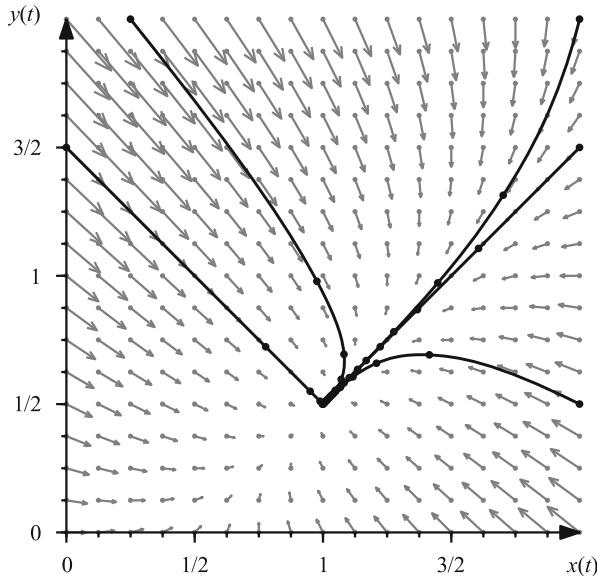


Fig. 2.2 Solutions corresponding to the initial values $(x_0, y_0) = (0, 3/2), (1/4, 2), (2, 2), (2, 3/2)$ and $(2, 1/2)$ and the direction field of the system of differential equations (2.2), filled circle symbols represent the step $\delta t = 10$

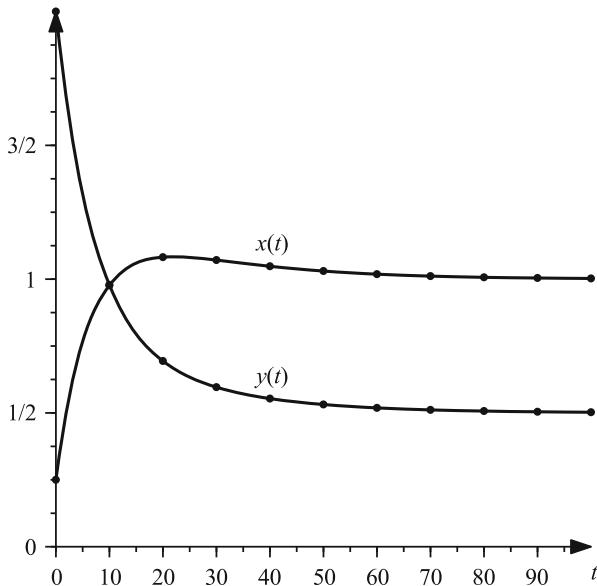


Fig. 2.3 Solution components of the system of differential equations (2.2) with initial values $(x_0, y_0) = (1/4, 2)$

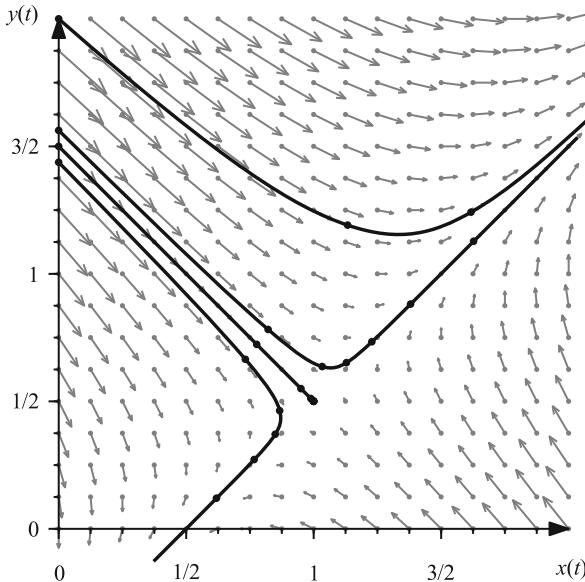


Fig. 2.4 Solutions for the initial values $(x_0, y_0) = (0, 3/2)$, $(0, 3/2 \pm 1/16)$ and $(0, 2)$ and the direction field of the system of differential equations (2.3)

in which the system matrix has the same eigenvectors v_1 and v_2 as in the previous example, but the eigenvalue $\lambda_1 = +0.05$ associated with v_1 is now positive (the other eigenvalue $\lambda_2 = -0.15$ and the equilibrium point x_∞ remain unchanged). Figure 2.4 shows the effects from the change in sign of λ_1 : now only those solution curves converge toward x_∞ for which the initial values lie exactly on the line in the direction v_2 from the equilibrium point. In the case of a minor perturbation, the solution curve initially runs almost parallel until the positive eigenvalue λ_1 has a sudden and almost explosive effect—the solution curve is redirected in the direction of v_1 and the velocity by which the solution distances itself from the equilibrium point grows exponentially. Hence, the existence of a positive eigenvalue leads to an unstable equilibrium which will be lost by an arbitrarily small perturbation.

Now let us address the numerical properties of initial value problems of ordinary differential equations. Particularly troublesome, of course, are—as always—the ill-conditioned problems which we will not address in the following. Nevertheless, to serve as a deterrent, we provide an ill-conditioned IVP as an example. Let the equation $\ddot{y}(t) - N\dot{y}(t) - (N+1)y(t) = 0$, and $t \geq 0$ be given with the initial conditions $y(0) = 1$, $\dot{y}(0) = -1$ and the solution $y(t) = e^{-t}$. If we now perturb just the one initial condition $y_\varepsilon(0) = 1 + \varepsilon$, a new solution $y_\varepsilon(t) = (1 + \frac{N+1}{N+2}\varepsilon)e^{-t} + \frac{\varepsilon}{N+2}e^{(N+1)t}$ results. Obviously, $y(t)$ and $y_\varepsilon(t)$ have a completely different character; in particular, $y(t)$ goes to zero as $t \rightarrow \infty$, whereas $y_\varepsilon(t)$ grows without bound for $N+1 > 0$, and this occurs for arbitrary $\varepsilon > 0$ (i.e., in particular arbitrarily small)! This shows that the smallest of perturbations

in the input data (here one of the two initial conditions) can thus have a disastrous effect on the solution of the IVP—a clear case of miserable conditioning!

Finite Differences

We now turn our attention to specific solution algorithms. In the following we consider the general IVP of first order $\dot{y}(t) = f(t, y(t))$, $y(a) = y_a$, $t \in [a, b]$ just mentioned and assume it has a unique solution. If f does not depend on its second argument y , then this is a simple integration problem! The starting point is, as always, the *discretization*, i.e., here: Replace derivatives or *differential quotients* with *difference quotients* or *finite differences*, resp. For example, the *forward*, *backward* or *central difference*

$$\frac{y(t + \delta t) - y(t)}{\delta t} \quad \text{or} \quad \frac{y(t) - y(t - \delta t)}{\delta t} \quad \text{or} \quad \frac{y(t + \delta t) - y(t - \delta t)}{2 \cdot \delta t}$$

for $\dot{y}(t)$ respectively or, for IVPs of second order,

$$\frac{\frac{y(t + \delta t) - y(t)}{\delta t} - \frac{y(t) - y(t - \delta t)}{\delta t}}{\delta t} = \frac{y(t + \delta t) - 2 \cdot y(t) + y(t - \delta t)}{(\delta t)^2}$$

for $\ddot{y}(t)$, etc. From above, the first of the approximations for $\dot{y}(t)$ leads to

$$y(a + \delta t) \approx y(a) + \delta t \cdot f(t, y(a)), \quad \text{also}$$

$$y_{k+1} := y_k + \delta t \cdot f(t_k, y_k),$$

$$t_k = a + k \delta t, \quad k = 0, 1, \dots, N, \quad a + N \cdot \delta t = b$$

and is the simplest rule to produce discrete approximations y_k for $y(t_k)$. Hence, at the time step t_k one takes the approximation y_k that has been already computed and then, together with f , calculates an approximation for the slope (derivative) of y and uses this for an estimate of y in the next time step t_{k+1} . This method is called *Euler's method*. One can also interpret or derive Euler's method via Taylor approximations of the solution $y(t)$. To this end, consider the Taylor expansion

$$y(t_{k+1}) = y(t_k) + \delta t \cdot \dot{y}(t_k) + R \approx y(t_k) + \delta t \cdot f(t_k, y_k)$$

in which all terms of higher order (i.e., with $(\delta t)^2$ etc.) are neglected.

In addition to this, there exist a number of related methods for IVPs of ODEs. The method by *Heun* is one such example:

$$y_{k+1} := y_k + \frac{\delta t}{2} (f(t_k, y_k) + f(t_{k+1}, y_k + \delta t f(t_k, y_k))).$$

The basic pattern, however, remains the same: One takes the approximation y_k at t_k that has already been computed and uses this to determine an approximation of the slope \dot{y} . This is then used to compute an approximation for the value of the solution y at the next time step t_{k+1} through multiplication with the step size δt . What is new is how to estimate the slope. In Euler's method, one simply takes $f(t_k, y_k)$. Heun's method attempts to better approximate the slope over the entire interval $[t_k, t_{k+1}]$ by computing the average of two estimates for \dot{y} taken at t_k and t_{k+1} . The difficulty arising from y_{k+1} not being determined yet can be avoided by using an Euler estimate as the second argument of f . It is easily seen that the individual time step has become more work-intensive (two function evaluations of f and more elementary computational operations when compared to the simple Euler's method).

The method by *Runge* and *Kutta* goes one additional step further along these lines:

$$y_{k+1} := y_k + \frac{\delta t}{6} (T_1 + 2T_2 + 2T_3 + T_4)$$

with

$$\begin{aligned} T_1 &:= f(t_k, y_k) , \\ T_2 &:= f\left(t_k + \frac{\delta t}{2}, y_k + \frac{\delta t}{2} T_1\right) , \\ T_3 &:= f\left(t_k + \frac{\delta t}{2}, y_k + \frac{\delta t}{2} T_2\right) , \\ T_4 &:= f(t_{k+1}, y_k + \delta t T_3) . \end{aligned}$$

This rule also follows the basic principle

$$y_{k+1} := y_k + \delta_t \cdot (\text{Approximation_of_the_slope}) ,$$

however, the computation for the approximation of \dot{y} has now become even more complicated. Starting with the simple Euler approximation $f(t_k, y_k)$, a skillfully nested procedure generates four suitable approximations which when—appropriately weighted—will then be used for the approximation. The appeal of these complicated rules is naturally rooted in the higher accuracy of the produced discrete approximations for $y(t)$. To quantify this, we must get a better handle on the notion of the *accuracy* of a method for the discretization of ordinary differential equations. Two issues have to be carefully separated: first, the error that accumulates *locally* at every point t_k and is independent from the use of any approximate solutions but is present due to the use of the difference quotients (based on the exact $y(t)$) used in the algorithm rather than the derivatives $\dot{y}(t)$ of the exact solution $y(t)$; second, the error that accumulates *globally* over the course of the computations from a to b , i.e., over the entire time interval being considered. In like manner, we distinguish two types of discretization errors, the *local discretization*

error $l(\delta t)$ (i.e., the part that is produced as a new error at each time regardless whether the difference quotient is determined using the exact $y(t)$) as well as the *global discretization error*

$$e(\delta t) := \max_{k=0,\dots,N} \{ |y_k - y(t_k)| \}$$

(i.e., the greatest possible amount by which one's calculations performed over the entire time interval may be off the mark).

If $l(\delta t) \rightarrow 0$ for $\delta t \rightarrow 0$, then the discretization scheme is called *consistent*. Consistency is obviously the minimum that needs to be required. An inconsistent discretization is utterly useless: If the approximations per time step are not even locally reasonable, meaning that more and more computational effort does not lead to better results, one cannot expect our given IVP to be reasonably solved. If $e(\delta t) \rightarrow 0$ for $\delta t \rightarrow 0$, then the discretization scheme is called *convergent*. The investment of more and more computational effort (smaller and smaller time steps δt) then leads to better and better approximations of the exact solution (vanishing error). Consistency is the weaker of the two concepts, rather of a technical nature and oftentimes this is relatively easy to prove. Convergence, in contrast, is the stronger concept (Convergence implies consistency, but not vice versa!), it is of fundamental and practical significance and oftentimes not quite trivial to show.

All of the three methods that have been introduced so far are consistent and convergent; Euler's method is of first order, Heun's method of second order and the Runge–Kutta method of fourth order, i.e.,

$$l(\delta t) = \mathcal{O}((\delta t)^4), \quad e(\delta t) = \mathcal{O}((\delta t)^4).$$

Here, their difference in quality is made apparent: The higher the order of the method, the more is gained from an increase in computational effort. When dividing the step size δt by two, for example, the error in Euler's, or Heun's, or the Runge–Kutta method is asymptotically reduced by a factor of 2, 4 and 16, respectively. The more expensive methods are thus the more effective (at least asymptotically). But we are still not satisfied. The number of *evaluations* of the function f for different arguments has strongly increased (compare the Runge–Kutta formulas: T_2 , T_3 and T_4 in which each require an additional evaluation of f). In numerical practice, f is typically very complicated (oftentimes a single evaluation of f requires another differential equation to be solved), so that one evaluation of f already involves a high computational effort.

The previous methods are examples of the so-called *one-step methods*: For the computation of y_{k+1} , no time points prior to t_k are exploited (but rather—as mentioned—new evaluation points). This is different for the *multistep methods*: Here, no new evaluation points of f are produced, but rather older

(and already computed) function values will be reused, for example in t_{k-1} for the *Adams–Bashforth method of second order*:

$$y_{k+1} := y_k + \frac{\delta t}{2} (3f(t_k, y_k) - f(t_{k-1}, y_{k-1}))$$

(second order consistency can easily be shown). Methods of even higher order can be constructed analogously by utilizing time steps t_{k-i} , $i = 1, 2, \dots$ that lie even further in the past. The principle here is a familiar relative to quadrature: Replace f by a polynomial p of suitable degree which interpolates f at the points (t_i, y_i) being considered and then use this p in accordance to

$$y_{k+1} := y_k + \int_{t_k}^{t_{k+1}} f(t, y(t)) dt \approx y_k + \int_{t_k}^{t_{k+1}} p(t) dt ,$$

in order to compute y_{k+1} (the polynomial p is easy to integrate). At the beginning, i.e., as long as there are not sufficiently many “old” values, one typically uses a suitable one-step method. The multistep methods that are based on this interpolation principle are called *Adams–Bashforth methods*.

Stability

The methods of Euler, Heun and Runge–Kutta are consistent and convergent. This is also true for the class of multistep methods of the Adams–Bashforth type just introduced. However, it turns out that for multistep methods, consistency and convergence do not always hold simultaneously. In order to be convergent, a consistent method must additionally be *stable* (in the sense of our definition of a numerically stable algorithm). The proof of stability is thus of eminent importance. The basic rule holds which says,

$$\text{consistency} + \text{stability} \Rightarrow \text{convergence} .$$

An example for an unstable discretization rule is found in the *midpoint rule* $y_{k+1} := y_{k-1} + 2\delta t f_k$ which is a consistent 2-step method (easy to show via Taylor expansion). Applying the midpoint rule, for example, to the IVP

$$\dot{y}(t) = -2y(t) + 1 , \quad y(0) = 1 , \quad t \geq 0 ,$$

having solution,

$$y(t) = \frac{1}{2}(e^{-2t} + 1),$$

leads to *oscillations* and *divergence*—no matter how small the chosen δt .

Stiffness, Implicit Methods

As a final phenomenon we consider *stiff* differential equations. These are understood to be scenarios in which an insignificant local property of the solution imposes an extremely high resolution upon the entire domain. An example is provided by the (well-conditioned) IVP $\dot{y} = -1000y + 1000, t \geq 0, y(0) = y_0 = 2$ having the solution $y(t) = e^{-1000t} + 1$. If one applies the (convergent!) Euler's method, then $\delta t \geq 0.002$ leads to no convergence. This is a contradiction only at first glance. The concepts of consistency, convergence and stability are of *asymptotic* nature: $\delta t \rightarrow 0$ and $\mathcal{O}(\delta t)$ always implies “for sufficiently small δt ”, and in the current case, this “sufficiently small” turns out to be “unacceptably small”. A cure for a problem's stiffness is given through the use of *implicit methods* in which the unknown y_{k+1} also appears on the right hand side of the method so that it has to be first solved for y_{k+1} . The simplest example of an implicit method is the *implicit Euler's* or *backward Euler's method*:

$$y_{k+1} = y_k + \delta t f(t_{k+1}, y_{k+1}) .$$

One notes the appearance of the (still to be determined) y_{k+1} on the right hand side of the rule. For the stiff problem considered above, an application of the implicit Euler's method leads to a situation where $\delta t > 0$ can now be chosen arbitrarily.

Why is that? Overly simplified, explicit methods approximate the solution of an IVP using polynomials whereas implicit methods use rational functions. Polynomials, however, cannot approximate e^{-t} for $t \rightarrow \infty$, for example, while rational functions can do so very well. Therefore, implicit methods are indispensable for stiff differential equations. However, the equation cannot always be solved easily for y_{k+1} . In the general case this may require a (nonlinear) iteration technique. Oftentimes, the *predictor-corrector approach* is of help: First employing the use of an implicit method, one determines an initial approximation for y_{k+1} , substituting this approximation into the right hand side of the implicit rule—the rule of thumb, twice explicitly thus turns into once implicitly, holds under certain prerequisites. Basically it holds that an implicit time step is more expensive than an explicit one. But in view of non-existent or at least less restrictive bounds for the step widths, implicit methods often require significantly fewer time steps.

Additional topics (not mentioned here) involving numerical properties of ODEs include *systems of ODEs*, *ODEs of higher order* as well as *boundary value problems*.

2.4.6 Partial Differential Equations

Classification

Partial differential equations (PDE) are likely to be even less analytically approachable than ODEs—not only because solutions in closed-form can hardly ever be

stated for practical and relevant cases, but also because the question regarding existence and uniqueness of solutions is often an open one. This turns the numerical treatment of PDEs into a must—despite the high computational effort that includes, in particular, the case of full spatial resolution. Concrete models are formulated as *boundary value problems* (e.g., in the steady case) or as *boundary-initial value problems*. Important boundary conditions are the *Dirichlet boundary conditions* (here, the function values are prescribed on the boundary) as well as the *Neumann boundary conditions*, in which the normal derivative is prescribed on the boundary.

An important class of PDEs is the *linear PDE of second order* in d dimensions:

$$\sum_{i,j=1}^n a_{i,j}(x) \cdot \partial_{i,j}^2 u(x) + \sum_{i=1}^n a_i(x) \cdot \partial_i u(x) + a(x) \cdot u(x) = f(x).$$

Within this class, one distinguishes between *elliptic* PDEs (the matrix A of coefficient functions $a_{i,j}(x)$ is positive or negative definite), *hyperbolic* PDEs (the matrix A has a positive and $n - 1$ negative eigenvalues or vice versa) as well as *parabolic* PDEs (an eigenvalue of A is zero, all the others have the same sign, and the rank of A together with the vector of coefficient functions $a_i(x)$ is maximal or full—if there is no second derivative with respect to a variable, then there is at least a first derivative). Simple yet well-known examples include the *Laplace- or potential equation* $\Delta u = 0$ for the elliptic case, the *heat equation* $\Delta T = T_t$ for the parabolic case and the *wave equation* $\Delta u = u_{tt}$ for the hyperbolic case. This distinction is not reserved for the accounting department—these three types display different analytic properties (e.g., shock phenomena such as the Mach cone exist only for hyperbolic PDEs) and require different numerical approaches for their solutions.

Discretization Approaches

Particularly true for the higher dimensional case, the discretization of the underlying domain is already work intensive. One distinguishes between *structured grids* (e.g., cartesian grids), in which the coordinates of grid points or cells, resp., do not have to be stored explicitly but can be determined via index computations, and *unstructured grids*, in which the entire geometry (coordinates) and topology (neighborhoods, etc.) have to be administered and stored. Examples for unstructured grids include irregular triangulations or tetrahedral grids.

Discretization techniques for PDEs have a variety of approaches that have been established over the course of years. *Finite difference methods* discretize the PDE directly in the sense that all derivatives are replaced by approximating difference quotients—a straightforward and easy to implement approach which, however, has its weaknesses with respect to its theoretical background and is furthermore basically restricted to structured or orthogonal grids. *Finite volume methods* are popular, in particular, in the context of fluid flow simulations. They discretize continuum mechanical conservation laws on small control volumes.

In contrast, *Finite-element methods* apply a variational approach—they look for a minimal-energy solution which is in direct relation to the underlying physics. Here the implementation is often more work intensive, but the pay-off is found from a high flexibility with respect to the employed grids (from structured to unstructured) as well as a very nice and rich mathematical theory. In addition there exist further concepts such as *spectral methods* or *meshfree methods* in which the perspective of moving particles is assumed instead of a fixed reference grid.

Finite Differences

Let a domain $\Omega \subseteq \mathbb{R}^d$, $d \in \{1, 2, 3\}$ be given. On it, we introduce a (structured) grid Ω_h with *grid width* $h = (h_x, h_y, h_z)$ (in 3 D). The definition of *finite differences* or difference quotients, follows analogously with the previous section on ODEs. For the first derivative, as in the case for the ODEs, *forward*, *backward* or *central differences* are customary,

$$\frac{\partial u}{\partial x}(\xi) \doteq \frac{u(\xi + h_x) - u(\xi)}{h_x}, \quad \frac{u(\xi) - u(\xi - h_x)}{h_x}, \quad \frac{u(\xi + h_x) - u(\xi - h_x)}{2h_x},$$

and for the second derivative, for example, there is the *3-point stencil*

$$\frac{\partial^2 u}{\partial x^2} \doteq \frac{u(\xi - h_x) - 2u(\xi) + u(\xi + h_x)}{h_x^2}.$$

The Laplace operator Δu correspondingly leads to the *5-point stencil* in 2 D and to the *7-point stencil* in 3 D. The notation “stencil” indicates which and how many neighboring points are used in the discretization. Wider stencils involve more points and therefore result in a higher approximation quality (elimination of further terms in the Taylor expansion).

For each interior gridpoint the PDE has now been replaced by a *difference equation*. The unknowns (*degrees of freedom*) are thereby the approximate values of the function values $u(\xi)$ at the discrete grid points ξ . Thus, one obtains one unknown per each grid point and per each unknown scalar quantity. Points on or near the boundary require special treatment. In the case of *Dirichlet boundary conditions*, one does not set up a difference equation in the boundary points (there, the function values are given and not unknown). In points next to the boundary, one obtains a “degraded” discrete equation since the stencils in parts involve known values which are moved to the right hand side. In the case of *Neumann boundary conditions*, one also prescribes difference equations at the boundary points. However, these equations have a modified form due to the condition for the normal derivative.

Overall, one obtains a large (one row for every scalar quantity in each (inner) grid point), sparse (only a few non-zeros resulting from the stencil) system of linear equations which subsequently needs to be solved efficiently.

Resulting is a certain *order of consistency* for the discretization that depends on the selected discrete operators; In addition, stability must also be proven. The areas to first consider for improvements could be the choice of stencils of higher order or local grid refinement (*adaptivity*). For the case of higher dimensions, as they appear in quantum mechanics or in mathematical finance, for example, the *curse of dimensionality* comes to the forefront: In d spatial dimensions, $\mathcal{O}(h^{-d})$ grid points and unknowns are required, which is naturally no longer manageable for $d = 10$ or $d = 100$.

Finite Elements

With *finite element methods (FEM)*, the derivatives are not discretized directly. Rather, the PDE is transformed into its so-called *weak form* (integral form). We summarize below the five essential steps:

1. *Substructuring* and *grid generation*: decompose the domain into individual parcels of a given pattern and of finite extension (*finite elements*);
2. *weak form*: no longer satisfy the PDE pointwise everywhere but only weakened (in form of an inner product) or averaged (as an integral), resp.;
3. *finite dimensional trial space*: replace the continuous solution in the weak form by a suitable finite dimensional approximation;
4. *system of linear equations*: use test functions to generate an equation for each degree of freedom;
5. *Solution of the linear system*: employ a suitable iteration method for the solution of the linear system.

The generation of finite elements can be interpreted as a top-down or a bottom-up process. Top-down decomposes the domain into standard components whose behavior is easy to describe; bottom-up originates from particular elements and uses these to approximate the given domain. In 3 D, one thus obtains a *finite element net* of *elements* (3 D atoms, e.g., cubes or tetrahedra), *surfaces* (2 D surface structures, e.g., triangles or squares), *edges* (1 D boundary structures of the element) as well as *grid points* or *nodes* in which the unknowns typically (but not necessarily) live. Each node is associated with a *trial function* φ_k , a basis function with finite support (non-zero only in neighbouring elements). Together, all trial functions span the linear and finite dimensional *trial space* V_n and form a basis. Within this trial space one seeks the approximation to the solution of the PDE.

We now address the weak form of the PDE whose representation we give through the general form $Lu = f$ with differential operator L (e.g., Δ), right hand side f and continuous solution u . Instead of $Lu = f$ on Ω , one now considers

$$\int_{\Omega} Lu \cdot \psi_l d\Omega = \int_{\Omega} f \cdot \psi_l d\Omega$$

for a basis of *test functions* ψ_l in a finite dimensional *test space* W_n . This procedure is also called the *method of weighted residuals* or, equivalently, the *Galerkin approach*. If the test and trial space are chosen to be the same ($V_n = W_n$), one then speaks of a *Ritz–Galerkin approach*, otherwise of a *Petrov–Galerkin approach*. One now writes the above equation in a slightly different way using both a *bilinear form* $a(.,.)$ and a *linear form* $b(.)$,

$$a(u, \psi_l) = b(\psi_l) \quad \forall \psi_l \in W_n ,$$

hence obtaining n discrete linear equations.

Lastly, one replaces the continuous solution u by its discrete approximation

$$u_n := \sum_{k=1}^n \alpha_k \varphi_k \in V_n$$

thus obtaining,

$$a(u_n, \psi_l) = \sum_{k=1}^n \alpha_k \cdot a(\varphi_k, \psi_l) = b(\psi_l) \quad \forall \psi_l \in W_n .$$

This yields a linear system of equations in the n unknowns α_k : The matrix consists of the entries $a(\varphi_k, \psi_l)$, the right hand side of the $b(\psi_l)$; all quantities do not depend on the solution but only on the problem.

As for finite differences, the FEM discretization has led to a discrete problem $Ax = b$. Here, one recognizes a tight coupling between the discretization and the solution: Some properties of the matrix are desirable (positive definite, sparse, as “close” as possible to diagonality) to ensure the efficient solution of the linear system. This is attempted through a “good” choice for the test and trial space.

Given the numerous application fields involving FEM, a multitude of different elements has been introduced over the years, each differing with respect to the form as well as with respect to the location and number of unknowns. For an increase in efficiency, *adaptive refinement* is very popular in which *local error estimation* (its importance is rooted with the decision where to refine) as well as *global error estimation* (whose importance is rooted to the decision when one can stop) are of great significance. One also refers to the *h-version* of FEM. Conversely, one can successively increase the approximation quality of the elements (*p-version* of FEM) or combine these two (*hp version*). With regard to the geometries that are to be described, the complexity increases if they are very *complicated* or if they are *changing* with respect to time, for example, if the problem involves free surfaces. To close, the remark pertaining to finite differences and the curse of dimensionality naturally applies here as well.

Table 2.1 Relations between tools and application scenarios: *filled circle* stands for a strong, *empty circle* for a weak relation

	Discrete structures	Higher mathematics	Stochastics	Numerical analysis
3 game theory			○	
4 group decisions	●			
5 scheduling	●		○	
6 Wiener processes			●	
7 traffic macroscopically		●		●
8 traffic microscopically	●		○	
9 traffic stochastically	●	○	●	
10 population dynamics		●	●	
11 control	●	●		
12 chaos theory		●		●
13 molecular dynamics	●			●
14 heat transfer	●			●
15 fluid dynamics	●			●
16 computer graphics	●		○	●

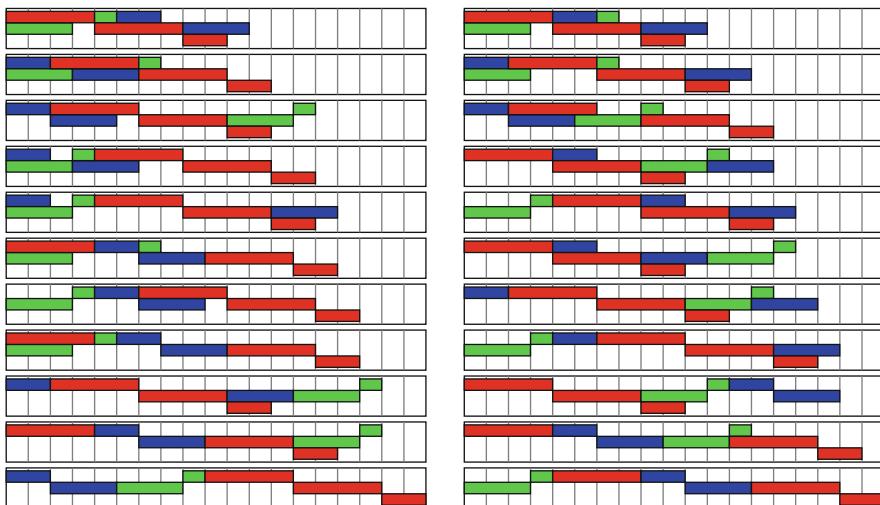
2.5 Interrelationships Between Tools and Applications

In this section we conclude Chap. 2 with a compact summary giving the relationships between the methodological tools previously mentioned on the one side and the application scenarios given in subsequent chapters on the other. The summary provided in Table 2.1 thereby will satisfy several purposes: first, in a forward sense, to indicate where a certain tool will appear; second in a backward sense, to indicate which fundamentals are required for a certain application; and third, to once more underscore the bandwidth and relevance of the spectrum of tools which find use in the field of modeling and simulation—in this book, but also as a matter of principle.

We will use the example of the stochastic traffic simulation described in Chap. 9 to briefly explain how the entries of the table originated and how they are to be interpreted. In Chap. 9, the modeling uses graphs, a central tool of discrete mathematics. Through Little's Law, the conservation law plays a role which is derived through analytical means. Finally, the title already indicates the relevance of stochastics (distributions, stochastic processes) as well as statistics (tests).

Part I

Gaming—Deciding—Planning: A Warm-up for Modeling



This part serves as an introduction to modeling, its examples chosen because a course of action can be formalized in a way that facilitates insight regarding optimal actions. First, we deal with a number of classical problems from game theory. Next, we consider the question regarding the manner in which a fair decision for the group can be reached given the opinions from its individuals. We also discuss the set-up of optimal time schedules before finally concluding with the treatment of random processes.

Besides the similarity existing with the theme, the essential criteria applied for the selection of examples was the desire to best illustrate the process of modeling but balanced to be accomplishable within a manageable effort.

On one side, this holds here from a mathematical perspective since the tools needed remain mostly elementary (with the exception of some topics from

stochastics), in particular, we note that differential equations remain excluded from this part. On the other side, this also holds from the perspective of the required computational effort—the examples can be worked with paper and pencil or at most after writing and executing a few lines of programming code. With our focus on this goal, the examples chosen are as simple as possible and they have been consciously selected; we acknowledge that for the given subject area, there naturally exist questions that require elaborate mathematical methods (for example, in mathematical finance) as well as a large computational effort (for example, in combinatoric optimization used in time-schedule generation) which can only be hinted at in our treatment.

In this spirit, we now begin with modeling by converting parts of the reality—albeit, for the examples, sometimes in a slightly artificial way—into formal models.

Chapter 3

Game Theory

As a point of entry into modeling (simulation, for the most part, will not yet have a role to play), we consider *strategic games*. Before giving a formal definition, we first consider two well-known examples:

- The *prisoners' dilemma*: Two bank robbers A and B are arrested. In absence of a confession, the attorney can only prove the illegal possession of weapons (three years of incarceration for each). So he offers each of them (independently and without any possibility existing that the two can communicate in order to coordinate anything) a guarantee that if a full and truthful confession is made then, as chief witness, only 1 year will be served while the other one—if continuing to plead innocence—suffers the full punishment (let this be 9 years here). To make it interesting, should both confess then both will get 7 years. What should the two do in order to get away as favorably as possible?
- The *battle of the sexes*: Here, partners A and B want to meet, and to be specific, either at the soccer game—which is the preference of A —or at a shopping center which B prefers (the assignment of roles is entirely left to you). Unfortunately, they forgot to agree on the specific location for their meeting. For each we know that: Their personal choice is to be with their partner at their (own) favorite place while their least desirable choice is to be alone (even if this is at their favorite place), and the remaining possibility “with the partner at his or her favorite place” ranges somewhere in the middle between the two extremes. There exists no possibility to communicate with one another so each must therefore decide independently whether they drive to the shopping center or to the stadium. What shall they do?

These two examples illustrate to us what makes game theory so attractive from the perspective of an introduction to modeling: For one, these problems are sufficiently clear to be explained easily, and any statements derived from them can easily be checked. Next, there is promise for an immediate pay-off from the abstraction and formalization of the problems: What actually is the reason behind the fact that in the two problems the decision-making is entirely different? It's definitely not due to the

embedding into the different stories. The stories themselves should be discarded quickly in favor of a formal (mathematical) notation in order to understand the underlying problem which is now reduced to its essential components. For only then there exists the possibility to analyze with reasonable effort those cases that can occur at all and from this the determination of a reasonable response. The respective statements are (as long as they refer to the model) of mathematical exactness and do not leave any room for the doubt which otherwise remains if one tries to puzzle over such problems.

Within the framework of the analysis, there will arise a rather non-intuitive procedure, (the *mixed strategies*) which shows a way out of the decision dilemma for certain problems. This is also a benefit from modeling, without which this possibility would not be obvious at all (and would definitely not be recognized as reasonable).

The mathematical tools employed in this chapter are mostly elementary, the exception involving the framework of mixed strategies in Sect. 3.6 in which some basic concepts from stochastics are used (the expected value of discrete random variables, independence; compare Sect. 2.3.2).

3.1 Games in Strategic Normal Form

We now build a formal framework with which we can describe not only the two examples given above but also the more general situations in which the involved people can choose between different possibilities in attempt to maximize the resulting benefit from their choices.

One remark beforehand to avoid the need to repeatedly point this out in the following: Since we do not study game theory as a subject in its own right, but rather as an example for modeling, we restrict ourselves to a very tight subdomain. Possible generalizations as well as alternative approaches will be covered marginally at best; our goal is to provide a few interesting examples that are nontrivial while yet manageable, and certainly not a comprehensive survey of game theory.

Let us start with the modeling of the involved people which here, independent of their situation (prisoner, partner, ...) are called *player*—their model is very simple since initially they do not have any further properties, thus we only need symbols: Here, the players shall always be denoted as A and B .

The actions that a player can perform (confess, deny, go to the stadium, go shopping, ...) are called *strategies* and, as well, are reduced to names—what defines the action is irrelevant, i.e., what is of singular importance are the consequences of the selected strategy. Let each player $X \in \{A, B\}$ have a (finite or infinite) set S_X of strategies, and we denote the strategy set of the respective other player by S_{-X} , i.e., $S_{-A} := S_B$ and $S_{-B} := S_A$. This notation will be helpful when players have thoughts concerning the other players. In both examples above we have that $S_A = S_B$, but in general, the two players will each have a different set of strategies from which to select their choices.

A game is called *finite* if S_A and S_B are finite. In this case, we let $n_A := |S_A|$ and $n_B := |S_B|$ to denote the number of respective strategies that can be enumerated: $S_A := \{a_1, \dots, a_{n_A}\}$ and $S_B := \{b_1, \dots, b_{n_B}\}$. Although our examples have so far been restricted to finite games, a strategy can be a continuous quantity, e.g., when it corresponds to the amount of invested capital. When considering all of the examples for finite games treated here, the number of strategies involved will be comparatively small in order prevent their respective representation from becoming too overwhelming. However, the strategy sets can certainly be very large as well.

With this, the game can now take place: A selects a strategy from S_A and B selects a strategy from S_B . The game is thus modeled by the selection of an element from the set of all strategy pairs $S := S_A \times S_B$.

For every outcome of the game, i.e., for every $s \in S$, each player X shall assess his or her benefit by means of a *payoff function* $u_X : S \rightarrow \mathbb{R}$. Interpreted in the colloquial sense of a game, it is possible for this to be a monetary gain, but if so, immediate difficulties could present themselves since e.g., a gain of 100€ could imply different value interpretations by the players depending on whether the other player won 100,000€ or nothing at all. The payoff function shall represent the total assessment of the outcome of the game for each player and is therefore usually not an easy quantity to clarify.

Continuing with the case of finite games, one can represent u_X via *payoff matrices* in which $U^X \in \mathbb{R}^{n_A \times n_B}$ and $U_{i,j}^X := u_X(a_i, b_j)$. Frequently, the *bimatrix* U^{AB} is used to give a concise representation in which its elements are given as $U_{i,j}^{AB} := (U_{i,j}^A, U_{i,j}^B)$. An example for a payoff matrix ($n_A = 2, n_B = 3$) is given:

$$U^{AB} = \begin{pmatrix} (30, 0) & (20, 20) & (10, 0) \\ (40, 30) & (0, 0) & (0, 40) \end{pmatrix}. \quad (3.1)$$

Given the case that a payoff matrix is used for a representation, player A thus selects the row (he/she is hence called the *row player*), e.g., row 1, while player B selects the column (thereafter called the *column player*), e.g. column 3. Using the matrix above, the payoff for A is then $u_A(1, 3) = 10$ while the payoff for B is $u_B(1, 3) = 0$.

In the case of the prisoners' dilemma one can interpret the verdict, i.e., the years to be served in prison, as a measure for the negative payoff and subsequently obtain the associated payoff matrix,

$$\begin{pmatrix} (-7, -7) & (-1, -9) \\ (-9, -1) & (-3, -3) \end{pmatrix}. \quad (3.2)$$

In the case of the “battle of the sexes” one can arbitrarily assign the value 20 to the most desired result, the value 10 to the middle result and a value 0 to the worst result; this yields a payoff matrix,

$$\begin{pmatrix} (20, 10) & (0, 0) \\ (0, 0) & (10, 20) \end{pmatrix}. \quad (3.3)$$

Zero-sum games provide an important special case. Here, for all $s \in S$, it holds that $u_A(s) = -u_B(s)$, i.e., in the case of finite games, $U^A = -U^B$: For each outcome of the game, a certain amount of money changes from one player to the other. For this case, it suffices to identify U^A as the payoff matrix in which A is then called the *winner* and B the *loser*—where gains and losses can obviously be negative as well.

The representation of a game using strategy sets and payoff functions is called a *strategic normal form*; this representation is more powerful than one might initially believe since, for example, a strategy could also be a sequence of moves in a game in which each turn alternates between players. However, the strategy sets here quickly become unmanageably large and the strategic normal form frequently ceases to provide a convenient and descriptive form.

In the following, we always assume that both players are knowledgeable of the entire payoff function—here one speaks of games with *complete information*.

Up to this point, the effort in modeling has only caused work—which, in view of the simple example problems, could give the impression of being unduly exaggerated—and without any benefit of having helped. This will change (shortly) as we now get to the essential part of modeling, namely the question pertaining to how the players choose their strategies.

3.2 Games Without Assumptions on the Opponent

First and foremost, the goal of our modeling is to account for considerations such as, in the prisoners' dilemma, players having to integrate the reaction of the other player into their own decisions. However, preliminary to this we first consider the more simplistic case which assumes that B makes his/her choice without consideration of A (e.g., B could represent nature which influences the success given the choice of actions taken by A —afterall, the weather does not really care whether A carries an umbrella); this circumstance is also known to A .

In this case, only the payoff function u_A is of significance and thus, since here we restrict our attention to finite games, the payoff matrix $U := U^A$.

The simplest situation is the *game with certainty*: If A knows b_j , the strategy player B chooses, it is easy for him or her to maximize his or her payoff. Player A must only look for a maximal element in the column j of U which B has chosen:

$$\text{Choose } \hat{i} \in \{1, \dots, n_A\} \text{ with } U_{\hat{i},j} \stackrel{!}{=} \max_{1 \leq i \leq n_A} U_{i,j} .$$

Although this is admittedly a trivial result, it nevertheless is given here in order that we become more familiar with the notation that will be of importance in the following.

Slightly more complicated is the *game with risk*. Assuming now that A has absolutely no information regarding the choice made by B , several options exist and with this the “psychology” of the player becomes relevant.

A player willing to take risks can always choose his or her action so that the maximal payoff remains a possibility. The player simply rates the action a_i by the maximum possible payoff, $\max_{1 \leq j \leq n_B} U_{i,j}$:

$$\text{Choose } \hat{i} \in \{1, \dots, n_A\} \text{ with } \max_{1 \leq j \leq n_B} U_{\hat{i},j} \stackrel{!}{=} \max_{1 \leq i \leq n_A} \max_{1 \leq j \leq n_B} U_{i,j}.$$

Alternatively, a more careful player might attempt to maximize the guaranteed payoff. For this, the focus is placed on the action a_i giving the best of the worst case scenarios, $\min_{1 \leq j \leq n_B} U_{i,j}$, by choosing:

$$\hat{i} \in \{1, \dots, n_A\} \text{ with } \min_{1 \leq j \leq n_B} U_{\hat{i},j} \stackrel{!}{=} \max_{1 \leq i \leq n_A} \min_{1 \leq j \leq n_B} U_{i,j}.$$

Example: For $n_A = n_B = 2$ and

$$U := \begin{pmatrix} 0 & 30 \\ 10 & 10 \end{pmatrix}, \quad (3.4)$$

the following *best* choices are illustrated:

- The maximal payoffs are 30 in the first and 10 in the second row, thus implying that the player willing to take risks will choose a_1 .
- Since the minimal payoffs are 0 in the first and 10 in the second row, the careful player will therefore choose a_2 .

Of course additional methods for the choice of a strategy can be constructed—e.g., one can assume a probability distribution for which B selects the strategies b_j (e.g., one reads the weather forecast before one plans any weekend activities, or one can assume a uniform distribution $P(b_j) = 1/n_B$ when there is no additional information). Thereafter, the action a_i is chosen in which the expected value of the payoff is maximal.

However, a selection guideline for the preferred method is not something provided by the model. In this sense, the results of this section are in contrast to the following in that there are indeed guidelines that are provided in the model that give the resulting actions which are entirely independent of the psychology and/or other characteristics of the players.

3.3 Response Mappings

Henceforth, it is assumed that both players simultaneously attempt to maximize their own payoff and in doing so each must make assumptions pertaining to the action of the other player.

To this end, it is helpful to think about what we would do if we knew that the opponent makes a certain choice, say $y \in S_{-X}$. The *response mapping* describes exactly this since it maps y to the set of all $x \in S_X$, the choices that are optimal given that the opposing player chooses y :

$$r_X : S_{-X} \rightarrow \mathcal{P}(S_X), y \mapsto \left\{ \hat{x} \in S_X : u_X(\hat{x}, y) = \max_{x \in S_X} u_X(x, y) \right\} ,$$

The mapping takes y into the power set of S_X since it is possible for several strategies to be optimal; in the case of infinite strategy sets it is not known a priori whether the maximum is assumed, i.e., whether $r_X(y)$ is never empty—however, this shall always be assumed in the following.

The game having the assumed conditions of certainty from Sect. 3.2 can now be reformulated by means of the response mapping: Player A chooses an element $\hat{i} \in r_A(b_j)$ (j is known).

Given the two mappings r_A and r_B , we compose the overall response mapping r as

$$r : S \rightarrow \mathcal{P}(S), (a, b) \mapsto r_A(b) \times r_B(a) ,$$

which assigns to a strategy pair (a, b) all those strategy pairs in which A chooses an optimal response to b and B chooses an optimal response to a . For a better illustration one can briefly assume that $r_A(b)$ and $r_B(a)$ are singletons, hence $r(a, b)$ is also a singleton, it contains the pair of the two strategies which A and B would gladly have chosen if they had known what the opponent does. It is important to realize here that nothing is said about the payoff $u_X(r(a, b))$ in the case that both change the strategy; it can actually be less than in the original situation (a, b) .

For the representation of the response mapping of a finite game, one can mark those elements $U_{i,j}^A$ in column j of the payoff matrix U^{AB} for which $a_i \in r_A(b_j)$, i.e., a_i is a best answer to b_j , and correspondingly mark those elements $U_{i,j}^B$ in every row i for which $b_j \in r_B(a_i)$, i.e., b_j is a best answer to a_i . We illustrate this as follows:

$$U^{AB} = \begin{pmatrix} (0, \underline{20}) & (30, \underline{20}) \\ (\underline{10}, 0) & (10, \underline{10}) \end{pmatrix} \quad (3.5)$$

in which $r_A(b_1) = \{a_2\}$, $r_A(b_2) = \{a_1\}$, $r_B(a_1) = \{b_1, b_2\}$ and $r_B(a_2) = \{b_2\}$.

As an example illustrating the continuous case, we consider the zero sum game with

$$S_A = S_B = [0, 1] \text{ and } u_A(a, b) = 2ab - a - b , \quad (3.6)$$

i.e., $u_B(a, b) = a + b - 2ab$.

This yields the response mappings

$$r_A(b) = \begin{cases} \{0\} & \text{for } b < \frac{1}{2} \\ [0, 1] & \text{for } b = \frac{1}{2} \\ \{1\} & \text{for } b > \frac{1}{2} \end{cases}, \quad r_B(a) = \begin{cases} \{1\} & \text{for } a < \frac{1}{2} \\ [0, 1] & \text{for } a = \frac{1}{2} \\ \{0\} & \text{for } a > \frac{1}{2} \end{cases}.$$

3.4 Dominant Strategies

We are now ready to consider games in which both players act simultaneously, yet each must take into consideration the presumed action of the other player. Although each player is unaware of the other's strategy, they both know the prerequisites described in the payoff function under which he/she makes his/her choice. Fortunately for us, among these games there are again those for which the analysis is very simple: If player X has a strategy x which is the best response for all $y \in S_{-X}$ ($x \in r_X(y)$ for all y), without hesitation, he/she can simply choose this strategy. Such a strategy is called a *dominant strategy*.

In the prisoners' dilemma, "confessing" is a dominant strategy for both players:

$$U^{AB} = \begin{pmatrix} (-7, -7) & (-1, -9) \\ (-9, -1) & (-3, -3) \end{pmatrix},$$

since independent to the other player's strategy, "denying" is never better than "confessing". Hence, the prisoners' dilemma is not a particularly interesting case; players acting rationally will obviously choose the strategy pair (a_1, b_1) with corresponding payoff -7 for everyone.

Here, the model was not really needed to determine the best strategy since we could also have figured this out with a little forethought—the model itself becomes useful in those cases in which dominant strategies do not exist. Nevertheless, it is indeed useful when supporting an obvious objection against this solution (actually making this problem more interesting): It is not immediately intuitive that rational action leads to a worse result than the payoff $(-3, -3)$ given through the choice (a_2, b_2) . Afterall, shouldn't they both deny and hope that the other has the same idea? Obviously this is not the case here since each player wants to optimize only his/her own benefit: Even if the accomplice denies, the other is still better off with a confession—after all, this is a dominant strategy. But not considered is the possibility that the confessing crook has cause to fear his accomplice's revenge when he is released after 9 years! Should this be the case, the expected damage must somehow be integrated into the payoff matrix—and thus we have reached a singularly important lesson in modeling: The statements that we obtain are just statements concerning the model and not about the underlying reality. If the resultant payoff matrix is like ours, then a confession is the rationally correct

choice—however, we cannot assess from the model the degree to which the payoff matrix is accurate.

Using the concept of dominant strategies, let us briefly consider cases which are to be solved under the assumption that dominant strategies are not available to every player. For games having only one player who has a dominant strategy, one can consider the strategy as being already selected so that the other player is in the situation “game with certainty”; incidentally, these games are also easy to solve. An illustration of this is the zero sum game having the payoff matrix

$$U^A = \begin{pmatrix} 20 & 30 \\ 10 & 0 \end{pmatrix},$$

i.e., a_1 is the dominant strategy for A , but lacking is a dominant strategy for B . Player B can therefore act as if he/she is in a game with certainty (that a rationally acting A chooses strategy a_1) and in response thus chooses strategy b_1 .

In a similar way one can solve, or at least reduce, games in which one or more strategies x_k are *dominated* by another strategy x_l in the sense that x_l is a strategy that, for no strategy of the other player, is worse than x_k , yet for at least one strategy of the other player is truly better than x_k . In this case one can eliminate strategy x_k from the game and proceed with the further analysis of the reduced game.

The more interesting cases, however, are those in which there are no dominant strategies and tend to, at least initially, put the players into a true dilemma.

3.5 Nash Equilibria

We have already considered two examples for games without dominant strategies, namely the “battle of the sexes” and the game in (3.1):

$$U^{AB} = \begin{pmatrix} (30, 10) & (\underline{20}, \underline{20}) & (\underline{10}, 0) \\ (\underline{40}, 30) & (0, 0) & (0, \underline{40}) \end{pmatrix},$$

for which we now introduce and study the principle of an *equilibrium*.

To this end, let us consider the strategy pair $\hat{s} := (a_1, b_2)$ with $U_{1,2}^{AB} = (\underline{20}, \underline{20})$ which distinguishes itself by the fact that a_1 is an optimal answer to b_2 ($a_1 \in r_A(b_2)$) as well as b_2 is an optimal answer to a_1 ($b_2 \in r_B(a_1)$), thus together, $\hat{s} \in r(\hat{s})$).

If the players agree to play \hat{s} , then A has no reason to change the strategy in view of $a_1 \in r_A(b_2)$ —as long as B keeps the agreement he could only deteriorate his situation. Vice versa, B also has no reason to change the strategy in view of $b_2 \in r_B(a_1)$: Rational players will therefore (out of self-interest, i.e., justified only by reasoning based on the game) keep the agreement. A strategy pair $\hat{s} \in S$ with $\hat{s} \in r(\hat{s})$ is called *Nash equilibrium*.

In the example, a previous agreement between players is actually not necessary since (a_1, b_2) is the only equilibrium point. Both players can therefore independently determine their actions using only the payoff matrix.

Another characterization of an equilibrium point (\hat{a}, \hat{b}) is,

$$\forall b \in S_B : u_B(\hat{a}, \hat{b}) \geq u_B(\hat{a}, b) \quad \text{and} \quad \forall a \in S_A : u_A(\hat{a}, \hat{b}) \geq u_A(a, \hat{b})$$

that follows directly from the definition of the response mapping; in the case of a zero sum game ($u_B(a, b) = -u_A(a, b)$) one can reformulate the first description to

$$\forall b \in S_B : u_A(\hat{a}, b) \geq u_A(\hat{a}, \hat{b}).$$

Together, both conditions can then be read as

$$\forall a \in S_A, b \in S_B : u_A(\hat{a}, b) \geq u_A(\hat{a}, \hat{b}) \geq u_A(a, \hat{b}).$$

In this case, the equilibrium point is also called a *saddle point*.

A strategy pair consisting of dominant strategies is obviously always an equilibrium point, and no new equilibrium points can arise by eliminating dominated strategies—therefore the search for equilibria yields all solutions that we could determine using the methods in Sect. 3.4.

If now there existed a statement of the kind, “Every game has exactly one equilibrium point”, then the problem would be solved entirely. Unfortunately, as this is not the case we must still devote some of our thoughts to games for which this is not satisfied.

For the “battle of the sexes” there exist two equilibrium points:

$$U^{AB} = \begin{pmatrix} (\underline{20}, \underline{10}) & (0, 0) \\ (0, 0) & (\underline{10}, \underline{20}) \end{pmatrix}.$$

Whereas an agreement is possible here in which none of the players has reason to violate it, lacking a priori communication there is no reason to prefer one equilibrium over the other. In general, this problem is hard to solve—in some cases one can further classify the equilibrium points and distinguish between more or less attractive ones; for this particular example, however, this is not possible because of the symmetry of the situation. Although the methodology of the next section is initially constructed for games without any equilibrium points, it will still yield a solution for this problem.

3.6 Mixed Strategies

The zero-sum game with the pay-off matrix

$$U^A = \begin{pmatrix} 5 & -5 \\ -5 & 5 \end{pmatrix}$$

has no Nash-equilibrium: One of the two players can always increase their own payoff by changing the chosen strategy (at the expense of the other player).

Since no strategy offers any advantages whatsoever, the players could come up with the idea for each to just randomly choose one of the two—and, as well, this approach will yield us an equilibrium point.

The players now play a modified game: Each player no longer chooses a strategy x_1 or x_2 out of S_X , but there is now a probability $0 \leq p_X \leq 1$ with which he/she chooses strategy x_1 (correspondingly he chooses x_2 with the probability $1 - p_X$). The new strategy sets are therefore $\tilde{S}_X := [0, 1]$, with the chosen probabilities p_X called *mixed strategies*. As payoff \tilde{u}_X , one considers the expected value of u_X .

Since both players make their choices independently, it holds that:

$$\tilde{u}_X(p_A, p_B) = E(u_X) = (p_A, 1 - p_A) \cdot U^X \cdot \begin{pmatrix} p_B \\ 1 - p_B \end{pmatrix}$$

Multiplying this out yields a sum in which the $U_{i,j}^X$ are weighted with the corresponding probability for the strategy being chosen). In the example,

$$\begin{aligned}\tilde{u}_A(p_A, p_B) &= 20p_A p_B - 10p_A - 10p_B + 5, \\ \tilde{u}_B(p_A, p_B) &= -20p_A p_B + 10p_A + 10p_B - 5.\end{aligned}$$

This results in the same response mapping as in (3.6):

$$\tilde{r}_A(p_B) = \begin{cases} \{0\} & \text{for } p_B < \frac{1}{2} \\ [0, 1] & \text{for } p_B = \frac{1}{2} \\ \{1\} & \text{for } p_B > \frac{1}{2} \end{cases}, \quad \tilde{r}_B(p_A) = \begin{cases} \{1\} & \text{for } p_A < \frac{1}{2} \\ [0, 1] & \text{for } p_A = \frac{1}{2} \\ \{0\} & \text{for } p_A > \frac{1}{2} \end{cases}.$$

The graphs of the response mappings intersect in the point $\hat{s} := (\frac{1}{2}, \frac{1}{2})$ which thus is the equilibrium point of the new game.

The agreement could therefore be: “Everyone tosses a coin, and his/her decision depends on its outcome”. Now, none of the players has any motivation to deviate from this strategy. This construction is called the *mixed extension* of a game, and here one can show that an equilibrium point always exists.

For the “battle of the sexes”, in addition to the equilibrium points $(0, 0)$ and $(1, 1)$ which correspond to the two equilibrium points of the initial game (such strategies with probabilities 0 or 1 are called *pure strategies*), the *mixed extension* has yet another one, namely $(2/3, 1/3)$, as one can see through a similar computation. As a result, for the continued case in which one has forgotten to agree upon the meeting place beforehand, the arrangement could read as follows: “Everyone rolls the die. If the die is a three or higher, the player goes to their own favorite place. Otherwise, the player goes to their partner’s favorite.” Here, once again none of the players has any reason to deviate from the agreement, but—in contrast to the two equilibrium points found from pure strategies, i.e., “In this case we will always go to the stadium”

and “In this case we will always go shopping”—now it can be said that no player feels at a disadvantage. This solution also has a serious disadvantage, however: The expected payoff is only $20/3$ for everyone, and as such it falls below the value of 10 which is obtained using pure strategies as long as one accepts the equilibrium point which is less favorable for oneself. Here, a better procedure would be to toss a coin (beforehand, while communication is still possible) and then agree upon one of the two equilibrium points given by the pure strategies. This concept of *correlated strategies*, however, shall not be further pursued.

3.7 Outlook

Instead of pursuing game theory further, we now interpret the results of the previous sections from the perspective of the modeler who has developed a mathematical model from the informally given problem. Two questions arise: First, what do we gain from the modeling, and second, how reliable are the obtained results?

Already made clear from our simple examples are the advantages of a mathematical model: The model is significantly easier to analyze than the original problem—the concept of an equilibrium, for example, can be grasped much more easily from within a mathematical rather than in its verbal description. We had seen that the existence and characteristics of equilibrium points critically determine strategic games, and the process of abstraction in the sense of “leaving out the insignificant” contributes significantly to the following realization: By means of the payoff matrix one can think through the different potential cases much more easily than in the problems posed as stories. However, one should not lose sight of the stories themselves; considerations taken for the model are oftentimes inspired by the original problem and must therefore be checked for their meaningfulness—here, for example, thoughts about equilibrium points and the assessment of results are, in general, easier if one can imagine the players involved.

With respect given to the reliability of the results that are inherently transferred from the model back to reality, one needs to once more emphasize the significance of the correctly chosen payoff function that often in reality is influenced by subjective factors and usually cannot be easily determined. To this end, we put forth another example: Considering the game (3.4) in which player A operates without any assumptions about the course of action of player B , one can initially imagine the payoff as a monetary amount and assume that the payoff is assessed proportionally to the amount gained. However, this is not necessarily the case: A player might see the outcome in relation to the optimal result for the case of b_j , i.e., be upset about the missed gain. This corresponds to a modified payoff matrix U' with entries

$$U'_{i,j} := U_{i,j} - \left(\max_{1 \leq k \leq n_A} U_{k,j} \right),$$

which can be interpreted as risk (missed gain in situation b_j)—here, null stands for “no risk”, the smaller (more negative) the matrix entry, the larger the risk. A careful player could now assess an action with a maximum risk $\min_{1 \leq j \leq n_B} U'_{i,j}$ in order to minimize it:

$$\text{Choose } \hat{i} \in \{1, \dots, n_A\} \text{ with } \min_{1 \leq j \leq n_B} U'_{\hat{i},j} \stackrel{!}{=} \max_{1 \leq i \leq n_A} \min_{1 \leq j \leq n_B} U'_{i,j},$$

which corresponds to the course of action of the careful player in Sect. 3.2 for the modified payoff matrix.

Example (3.4) yields the risk matrix U' in the form

$$U' = \begin{pmatrix} -10 & 0 \\ 0 & -20 \end{pmatrix},$$

such that here the risk minimizer chooses a_1 in order to restrict the risk to -10 . Interestingly, in this example, given the modified payoff matrix, the careful player acts exactly the same as the player willing to take a risk given the original payoff matrix. Since a change of the payoff function can lead to a completely different result, one needs to take special care during its determination—if the payoff function is not correct, then it should be obvious that all considerations downstream that are based on the model become invalid. Another issue to be discussed when assessing our results is the assumption that both players act rationally, i.e., we can assume that they will indeed realize a declared strategy of an equilibrium. As an example, let us consider the game with payoff matrix

$$U^{AB} = \begin{pmatrix} (\underline{10}, \underline{10}) & (10, 0) \\ (0, 10) & (\underline{20}, \underline{20}) \end{pmatrix}.$$

In theory, it is obvious what the players have to do: They agree upon the equilibrium point (a_2, b_2) with the highest payoff of 20 for everyone. However, practical experience shows that many players do not choose this equilibrium but instead choose the strategy pair (a_1, b_1) with lower but riskfree payoff. Even though rational players have no reason to deviate from the agreed upon strategy (a_2, b_2) , one is still not completely sure whether the other player will actually act rationally.

Finally, we emphasize once more that game theory offers much more (and a lot more interesting) than these simple cases whose roles served to provide examples for models. We will reserve any additional discussion on game theory, however, to the more specialized literature (e.g. the textbooks [23, 45]) and instead turn toward another class of problems which we want to model.

Chapter 4

Group Decision Making

This chapter will deal with situations involving several possibilities that can and should be ranked—for example parties at an election, singers at a singing competition or the variations given in the traffic routing in a city.

These possibilities offer different perspectives regarding the individuals considered (e.g., in the case of the election these would be the voters, for the song contest, the audience, and for the traffic routing, possibly all the motorists would be considered, or maybe only the members of the district council). From these views—which will generally be contradictory—a consolidated ranking shall be determined, i.e., there exists an electoral law or another determination of a method which selects one particular ranking out of all that are possible.

In general, one does not assume that everyone will agree with this choice, and those not satisfied could now argue that the procedure was “unfair”. They may even construct examples to demonstrate how the procedure is “blatantly absurd”, and this leads to the question how to assess different decision making methods.

Here we will pursue the so-called *axiomatic* approach in which the properties of decision making methods are established, and thereafter applied to determine which methods actually satisfy them (conversely, one could analyze the probabilities related to undesirable situations and choose to accept them nonetheless if their chance for occurrence has low enough probability.).

To this end, one first models the individuals’ preferences, and then the decision making method itself. In this model, we will consider several examples for decision making methods and in particular situations which lead to undesirable results. It will be seen that the occurrence of these will be almost inevitable—namely, it can be proven that under the requirements that are formulated here, no method satisfies all conditions whenever more than two possibilities become available.

This chapter requires only elementary mathematics—relations and their properties (compare Sect. 2.1) will play an important role, and because the essential definitions are provided in this chapter, there are no special prerequisites necessary.

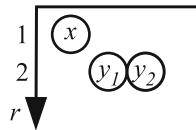


Fig. 4.1 Rank mapping: a voter prefers x with $r(x) = 1$ over y_1 and y_2 with $r(y_1) = r(y_2) = 2$. He has no preference regarding y_1 and y_2

4.1 Individual Preferences and Group Decision Making

We consider a finite set A of candidates (here, it is obviously of no importance whether these are parties, singers, plans, etc.) that are assessed by the individual voters. Eventually, we want to construct a group decision from the entirety of these assessments, but first we must formalize the assessment of the individual voter.

The preferences to be considered here are generated when every voter assigns to each candidate x a natural number denoting its ranking $r(x)$. For two candidates $x, y \in A$, $r(x) < r(y)$ implies that x is preferred over y .

Here, two candidates are permitted to obtain the same ranking (if in a subset of candidates everyone is assessed equally), all rankings from 1 up to the largest assigned number k should occur:

Definition 4.1 (Rank mapping). A rank mapping is a surjective mapping r from the set of candidates A onto a set $\{1, \dots, k\} \subset \mathbb{N}$.

Let us assume that a voter wants to elect candidate $x \in A$ but has no preferences regarding the remaining candidates $A \setminus \{x\}$. Then he can construct r as $r(x) = 1$ and $r(y) = 2$ for all $y \in A \setminus \{x\}$. Figure 4.1 shows this rank mapping for a candidate set of three elements.

In the following, it will prove helpful to describe these preferences in the form of relations. For this purpose we recall: A *relation* R on A is a set of pairs (x, y) of elements in A , i.e., a subset of the Cartesian product $A \times A$, where typically $(x, y) \in R$ is abbreviated by xRy . Several properties are defined for relations; those relevant for this chapter are listed in the following definition.

Definition 4.2 (Properties of relations). A relation R on A is called

- *transitive*, if xRy and yRz imply xRz :

$$\forall x, y, z \in A : (xRy \wedge yRz) \Rightarrow xRz ,$$

- *reflexive*, if xRx holds for all $x \in A$,
- *preorder*, if R is transitive and reflexive,
- *asymmetric*, if xRy and yRx never hold simultaneously:

$$\forall x, y \in A : xRy \Rightarrow \neg(yRx) ,$$

- *connex (linear)*, if any two elements are comparable:

$$\forall x, y \in A : xRy \vee yRx .$$

The rank mapping defines a preference relation $\varrho \in A \times A$ via

$$x\varrho y : \Leftrightarrow r(x) < r(y) . \quad (4.1)$$

This relation ϱ is transitive and asymmetric (since it is also the case for the relation $<$ on \mathbb{N}). The set of all possible relations on A to be found through a rank mapping is called

$$P_A := \{\varrho \subset A \times A : \varrho \text{ satisfies (4.1) for a rank mapping } r\} .$$

One can also add all pairs to the relation in which both candidates share the same ranking and, in doing so, obtains,

$$x\varrho^* y : \Leftrightarrow r(x) \leq r(y) . \quad (4.2)$$

This relation ϱ^* is transitive and reflexive (thus a preorder) and, in addition, connex—this again holds because the corresponding relation \leq on \mathbb{N} has these properties. Obviously, it holds that $\varrho \subset \varrho^* \subset A \times A$.

For the above example ($A = \{x, y_1, y_2\}$, $r(x) = 1$, $r(y_1) = r(y_2) = 2$), the relations ϱ and ϱ^* can be illustrated as follows—the field in row i and column j is marked if $i\varrho j$ (left) or $i\varrho^* j$ (right):

ϱ	x	y_1	y_2	ϱ^*	x	y_1	y_2
		×	×		×	×	×
x						×	×
y_1						×	×
y_2						×	×

Even if knowledge of the rank mapping r is absent, one can determine the corresponding ϱ^* from a given ϱ and vice versa since it holds that,

$$x\varrho y \Leftrightarrow \neg(y\varrho^* x)$$

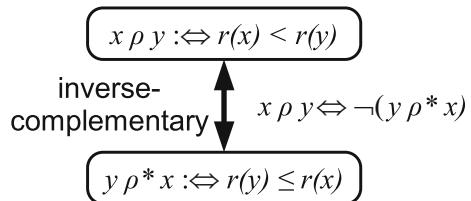
(with respect to each other, the relations ϱ and ϱ^* are *inverse-complementary*, see Fig. 4.2).

Analogous to the set P_A , we now define the set

$$P_A^* := \{\varrho^* \subset A \times A : \varrho^* \text{ satisfies (4.2) for a rank mapping } r\} .$$

Apparently, P_A^* contains only connex preorders; in fact, one can easily reason that it contains all connex preorders on A since we can always sort a given finite set with a connex preorder by this preorder and thus construct the corresponding rank mapping.

Fig. 4.2 The relations ρ and ρ^* defined by the rank mapping r



Furthermore, one sees that this correlation between rank mappings and connex preorders is unique: Different rank mappings generate different relations (this is precisely the reason for requiring the subjectivity of rank mappings). Overall, there exists a one-to-one relationship (bijection) between the set of all rank mappings and the set P_A^* of connex preorders on A .

We now have three different, but equivalent models of the preferences of an individual with respect to the candidate set A : through the rank mapping r , through the asymmetric relation ρ and through the connex preorder ρ^* .

To have different, yet equivalent models of a circumstance can be quite useful—one is then afforded the luxury of choosing the most suitable for the given situation. For example, in Sect. 4.3, we formulate requirements for the decision making method via the relation ρ since this is best suited for the desired goal (to formulate Arrow's theorem). As an introduction and simple visualization, however, the rank mapping is often the one best suited (we frequently work with rank mappings when we study the sport section of the newspaper).

To this point, however, the modeling has not caused any principle difficulties; these only occur when one attempts to form a preference of the entirety from the preferences of the individuals—if hereby some further to be defined principal rules should be abided.

For now, we define the *set of voters* through the simple enumeration of the individuals from 1 to n as a set

$$I := \{1, \dots, n\}.$$

A *decision making method* uses as input data the n preferences of the voters and, using a *collective choice function*,

$$K : P_A^n = P_A \times P_A \times \dots \times P_A \rightarrow P_A,$$

obtains a preference of the entirety.

In the following, through our analysis of examples of decision making methods, we will become acquainted with their desirable and undesirable properties and subsequently impose conditions for admissible decision making methods (which unfortunately turn out to be unrealizable in general).

Two essential (and nontrivial) conditions are already hidden in the definition, but deserve mentioning explicitly here:

- As a mapping, K must be total (each element of the domain must be mapped to an image), i.e., arbitrary combinations of arbitrary preferences of P_A are admissible.
- The result must again be a relation in P_A (this does not at all occur automatically as we will see shortly).

4.2 Examples for Decision Making Methods

The model from the previous section for decision making methods can now be used to select the best possible method from among all possible methods. In order to figure out the leeway that our model gives us, we now consider some choice functions or attempts to construct them.

As a goal we imagine methods that are as “fair” as possible—we do not formally define what is meant by this, but instead rely at least initially on our intuition to tell us that the first two of the following methods are far away from fairness.

External Dictator

The simplest functions are those yielding a constant result, i.e., the function value does not depend on the parameters, and to this point, such functions have not been excluded: For an arbitrary, but fixed $\varrho_E \in P_A$ (which one can associate with the preference of a dictator who is not included in the set of voters), one defines

$$K_{\varrho_E}^E(\varrho_1, \dots, \varrho_n) := \varrho_E ,$$

whose value is determined independently from ϱ_i .

This is obviously a mapping $P_A^n \rightarrow P_A$, i.e., an admissible choice function. It is just as obvious, however, that this method will hardly satisfy any definition for fairness—in Sect. 4.3, we will construct a formal criterion through the use of the *Pareto condition* which, among other things, will exclude an external dictator.

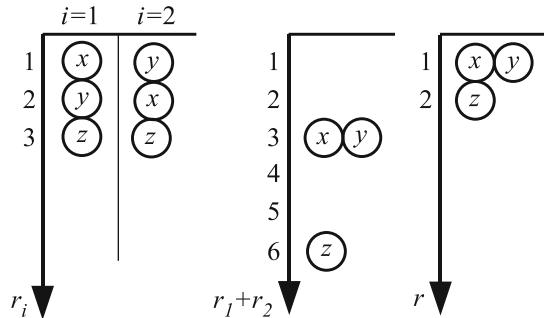
Internal Dictator

Another method that clearly yields an admissible preference for its result can be obtained by appointing a voter $d \in \{1, \dots, n\}$ (which can be seen as an internal dictator) whose preference becomes the result of the choice function:

$$K_d^D(\varrho_1, \dots, \varrho_n) := \varrho_d , \tag{4.3}$$

the result determined independently of the other ϱ_i with $i \neq d$.

Fig. 4.3 In general, the addition of the rank mappings does not result in a rank mapping, however, this is easily repaired by “compression”



This mapping $P_A^n \rightarrow P_A$ also satisfies the previous requirements for the collective choice function but neither will hold out against any definition of fairness. Therefore, we now turn to attempts which will, in a “reasonable” way, consolidate the different individual preferences into a collective preference.

Rank Addition

Since every individual preference ϱ_i is associated with a rank mapping it stands to reason that one must determine the rank of a collective preference from the individual voters’ rankings of the candidates; this can easily be accomplished by adding the ranks: The candidates x and y will be assessed by the sum of their rank numbers. In doing so, $K^A(\varrho_1, \dots, \varrho_n)$ is the relation ϱ , and defined by

$$x \varrho y : \Leftrightarrow \sum_{i=1}^n r_i(x) < \sum_{i=1}^n r_i(y).$$

The sum $\sum r_i$ is, in general, not a rank mapping (it needs not be surjective). However, one easily sees that the generated relation ϱ lies in P_A (the requirement of surjectivity only ensures the uniqueness of the rank mapping, see Fig. 4.3). At first glance, this method has no obvious disadvantages—a detailed analysis, however, reveals that (as it is the case for related methods) certain combinations of individual preferences can lead to unpleasant results. It is because of this that we will consider two additional ways to construct choice functions.

Condorcet Method (Majority Decision)

When comparing two candidates x and y , there exists a set of voters $\{i \in I : x \varrho_i y\}$ who prefer x , a set of voters $\{i \in I : y \varrho_i x\}$ who prefer y , and finally a set of voters who are indifferent with respect to x and y (any of the three sets may, of course, be empty).

The *Condorcet method* takes into account which one of the two candidates wins more comparisons. This yields the collective preference relation ϱ with

$$x \varrho y : \Leftrightarrow |\{i \in I : x \varrho_i y\}| > |\{i \in I : y \varrho_i x\}| . \quad (4.4)$$

This method can be performed for arbitrary ϱ_i , but for the case of more than two candidates the resulting relation ϱ is not always transitive, i.e., it is not always an admissible preference relation of P_A .

An example in which a non-transitive relation occurs is given in the following rank mappings for the case of three candidates $A = \{x, y, z\}$, and three voters $I = \{1, 2, 3\}$:

	$r_i(x)$	$r_i(y)$	$r_i(z)$
$i = 1$	1	2	3
$i = 2$	3	1	2
$i = 3$	2	3	1

Here, the counting that results from the Condorcet method results in $x \varrho y$, $y \varrho z$ and $z \varrho x$. If ϱ is transitive, then $x \varrho x$ would hold from these three relationships which is obviously not the case. Consequently, $\varrho \notin P_A$.

Hence, the Condorcet method does not provide a collective decision function when the candidate set A contains more than two elements (as long as one has to decide between only two candidates, the method is equivalent to the rank addition).

Unanimity

One can place a candidate x ahead of y precisely when all voters have done this. This yields the collective preference relation

$$x \varrho y : \Leftrightarrow \forall i \in \{1, \dots, n\} : x \varrho_i y . \quad (4.5)$$

This principle can also be viewed as the minimal consensus of the entirety, since a single voter who appreciates y at least as much as x leads to the corresponding result in the collective preference:

$$\begin{aligned} \exists i \in \{1, \dots, n\} : y \varrho_i^* x &\Leftrightarrow \exists i \in \{1, \dots, n\} : \neg(x \varrho_i y) \\ &\Leftrightarrow \neg(x \varrho y) \\ &\Leftrightarrow y \varrho^* x . \end{aligned}$$

In practical applications, this method tends to generate no true preferences at all: Since almost always there holds $y \varrho^* x$ (somebody will surely be of this opinion), $x \varrho y$ can almost never hold.

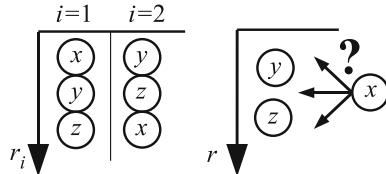


Fig. 4.4 The attempt to use unanimity as a decision making criterion: y must be placed before z , x may neither lie before nor after y and be placed neither before nor after z . Thus, we do not have any preference relation in P_A

Besides this “decision making weakness”, this method has the additional disadvantage that for $|A| > 2$ (similar to the Condorcet method), the result ϱ is not always in P_A , i.e., is does not yield a collective choice function. An example is now given in which the principle of unanimity does not yield a preference relation in P_A : Again, there are three candidates, $A = \{x, y, z\}$, and this time only two voters, $I = \{1, 2\}$, with rank mappings

	$r_i(x)$	$r_i(y)$	$r_i(z)$
$i = 1$	1	2	3
$i = 2$	3	1	2

The relation determined by the principle of unanimity ϱ contains exactly one pair $y\varrho z$, and it holds that $\varrho \notin P_A$. In order to see this, one can try to construct the corresponding rank mapping which turns out to be impossible (see Fig. 4.4). Here, it is also helpful that we have an additional characterization of P_A : the inverse complementary relations to the relations in P_A are exactly the connex preorders. If one considers the ϱ^* corresponding to the ϱ of this example, then one quickly sees that it is not transitive: It holds that $z\varrho^* x$ and $x\varrho^* y$, yet $z\varrho^* y$ is not valid. Thus, $\varrho^* \notin P_A^*$ and therefore $\varrho \notin P_A$.

One could now try to use unanimity in the ϱ_i^* as a decision making method (in (4.5), replace ϱ by ϱ^* and ϱ_i by ϱ_i^*), but this only further aggravates the problem: Even when $|A| = 2$, one in general no longer obtains an admissible preference relation. This can be seen in the example with two candidates x and y and two voters, of which one (truly) prefers x and the other one y : The resulting relation ϱ^* is not connex since neither $x\varrho^* y$ nor $y\varrho^* x$ holds.

4.3 Conditions for Choice Functions, Arrow’s Theorem

In the previous section we have seen that the previous requirements for collective choice functions (to be a mapping $P_A^n \rightarrow P_A$) are also satisfied by methods that are all but desirable. Thus we now formulate two conditions that a “fair” method should satisfy:

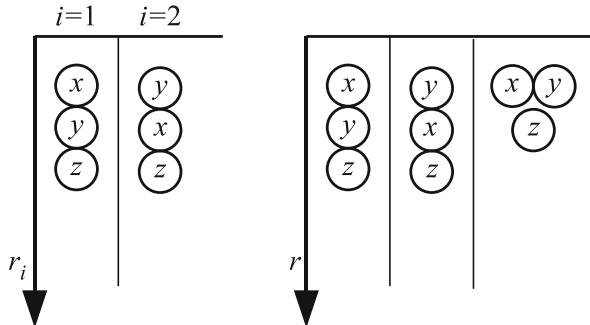


Fig. 4.5 Two voters agree that \$z\$ is to be ranked after \$x\$ and \$y\$. A collective choice function that satisfies the Pareto condition (4.6) must place \$x\$ and \$y\$ before \$z\$ in the group decision, thus only the three results represented to the right come into consideration

- The *Pareto condition* which asserts that the entirety can always force any desired ranking for arbitrary candidates through unanimity,
- and the principle of *independence of irrelevant alternatives* which asserts that the order of two candidates cannot be reversed simply because voters change their preferences with respect to a third candidate.

One could introduce additional wishes for a choice method, but as it turns out, the two assertions above already exclude all methods which can be considered even remotely democratic—this is the statement of *Arrow's theorem*.

Pareto Condition (Unanimity)

Using unanimity as a necessary and sufficient criterion (4.5) turned out to be impractical. However, it is reasonable to expect that unanimity for the preference between two candidates is sufficient for the group decision to have the same order, see Fig. 4.5.

Definition 4.3 (Pareto condition). A collective choice function \$K : P_A^n \rightarrow P_A\$ satisfies the *Pareto condition* if for all \$\varrho_i \in P_A, i = 1, \dots, n\$ with \$\varrho = K(\varrho_1, \dots, \varrho_n)\$ and for all \$x, y \in A\$ it holds that:

$$(\forall i \in \{1, \dots, n\} : x \varrho_i y) \Rightarrow x \varrho y . \quad (4.6)$$

If one requires the Pareto condition to be satisfied, then the external dictator is eliminated; all the other methods introduced in the previous Sect. 4.2 obviously satisfy the condition.

It is reasonable to define this condition in the relation \$\varrho\$ and not in the relation \$\varrho^*\$ since only then the voters can enforce a true preference—if in (4.6) one again replaces \$\varrho\$ by \$\varrho^*\$ and the \$\varrho_i\$ by \$\varrho_i^*\$, then the method which always ranks all candidates

equally ($r(x) = 1$ for all $x \in A$ independent of the ϱ_i) would automatically satisfy the condition. Occasionally one requests a tighter condition than (4.6) in which the relation $x\varrho y$ must already hold if only *one* voter is of this opinion ($\exists i \in \{1, \dots, n\} : x\varrho_i y$) and as long as no other voter requests the opposite ($\forall i \in \{1, \dots, n\} : x\varrho_i^* y$):

$$(\exists i \in \{1, \dots, n\} : x\varrho_i y) \wedge (\forall i \in \{1, \dots, n\} : x\varrho_i^* y) \Rightarrow x\varrho y . \quad (4.7)$$

This condition is called the *strong Pareto condition* (and, in contrast, (4.6) is also called the *weak Pareto condition*); in the following we will only use the weaker condition (4.6).

Independence of Irrelevant Alternatives

The justification of the next condition is slightly less obvious than that for the Pareto condition, therefore we begin with an example which displays an undesired situation.

To this end, we consider three candidates, $A = \{x, y, z\}$, and two voters with two different individual preferences ϱ_i and ϱ'_i , $i = 1, 2$; as a choice method, the rank addition K^A is used. The following tables provide the rank mappings and the sum of the rankings:

ϱ	$r_i(x)$	$r_i(y)$	$r_i(z)$	ϱ'	$r'_i(x)$	$r'_i(y)$	$r'_i(z)$
$i = 1$	1	2	3	$i = 1$	1	2	3
$i = 2$	2	1	3	$i = 2$	3	1	2
$\sum r_i$	3	3	6	$\sum r'_i$	4	3	5

The individual preferences ϱ_i (left) correspond to those in Fig. 4.3; for ϱ'_i (right), the voter $i = 2$ changes his opinion with respect to the candidates x and z .

Let us now consider the group decision with respect to x and y . There holds $y\varrho' x$, but not $y\varrho x$: The result changes even though neither of the two voters have changed their opinions in the direct comparison with respect to the candidates x and y (voter 1 ranks x before y , voter 2 decides vice versa).

In most cases one wants to exclude this situation: The ranking of a third candidate z should have no influence on the result of a direct comparison between x and y . In other words: If no voter changes his opinion concerning the order of x and y ($x\varrho_i y \Leftrightarrow x\varrho'_i y$ for all i), then the order of x and y should also remain unchanged in the result ($x\varrho y \Leftrightarrow x\varrho' y$), see Fig. 4.6. This motivates the following definition.

Definition 4.4 (Independence of irrelevant alternatives). A collective choice function $K : P_A^n \rightarrow P_A$ satisfies the *independence of irrelevant alternatives* if for all $\varrho_i, \varrho'_i \in P_A$, $i = 1, \dots, n$ with $\varrho = K(\varrho_1, \dots, \varrho_n)$, $\varrho' = K(\varrho'_1, \dots, \varrho'_n)$ and for all $x, y \in A$ it holds that:

$$(\forall i \in \{1, \dots, n\} : x\varrho_i y \Leftrightarrow x\varrho'_i y) \Rightarrow (x\varrho y \Leftrightarrow x\varrho' y) . \quad (4.8)$$

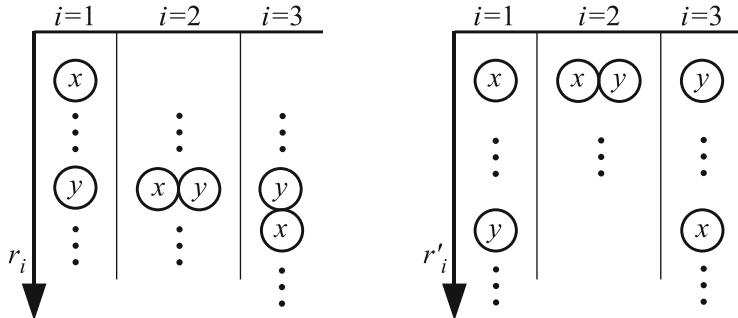


Fig. 4.6 Two “ballots”, none of the three voters change their opinions with respect to candidates x and y . The independence of irrelevant alternatives (4.8) requests that the result of the collective choice function does not do this neither

Although one could now formulate additional conditions for the decision making methods, we have just excluded the method of rank addition from the set of “fair” methods; thus we have only one left which does not appeal to us: Out of all the methods introduced in Sect. 4.2 that satisfy the formal conditions for a collective choice function (the Condorcet method (4.4) and the method of “unanimity” (4.5) already fails here), only the internal dictator K^D satisfies the Pareto condition (4.6) as well as the independence of irrelevant alternatives (4.8).

Arrow's Theorem

If nothing else, the modeling of the group decision making has provided us with arguments why the considered methods are problematic. An obvious reaction would be an attempt to construct methods that satisfy the established conditions but are—in a sense to be specified later, but in any case contrary to the dictatorship K^D —fair.

These attempts would not be successful, and now the mathematical modeling pays off in an unexpected way: It can be proven that such attempts must fail. The formalism previously carried out is therefore no end in itself but facilitates statements that can formally be proven.

A well-known example for such a statement is

Theorem 4.5 (Arrow's theorem). *Let A with $|A| > 2$ be a set of candidates with more than two candidates and $K : P_A^n \rightarrow P_A$ a collective choice function that satisfies the Pareto condition (4.6) and the independence of irrelevant alternatives (4.8). Then there always exists a dictator:*

$$\exists d \in \{1, \dots, n\} : \forall (\varrho_1, \dots, \varrho_n) \in P_A^n : \forall (x, y) \in A \times A : x \varrho_d y \Rightarrow x \varrho y \quad (4.9)$$

with $\varrho := K(\varrho_1, \dots, \varrho_n)$.

(“A sufficiently fair system with more than two candidates always contains a dictator.”)

The theorem is not proven here (proofs can e.g. be found in [26] or [53]), but to clarify its statement we compare the dictator (4.9) with the dictator in (4.3). Here, one sees that the former is slightly more “generous” toward the other group members: If he is indifferent concerning two candidates x and y (neither $x \varrho_d y$ nor $y \varrho_d x$ holds), then in the case of (4.3) this must also be satisfied in the result, whereas (4.9) permits arbitrary orders of x and y .

Still, Arrow’s theorem means the end of our search for the “perfect” decision method (and saves us from possibly lengthy trials): We cannot find a method that satisfies all of the reasonable assertions.

Finally, we mention an alternative formulation of Arrow’s theorem in which two of the conditions of the definition of the collective choice function are listed explicitly again and in which the dictator is explicitly excluded. Hence, one has five “*democratic basic rules*”:

1. The choice function K must be defined for all of P_A^n .
2. The result of K must always lie in P_A .
3. The Pareto condition (4.6) must be satisfied.
4. The independence of irrelevant alternatives (4.8) must be satisfied.
5. There is no dictator:

$$\exists d \in \{1, \dots, n\} : \forall (\varrho_1, \dots, \varrho_n) \in P_A^n : \forall (x, y) \in A \times A : x \varrho_d y \Rightarrow x \varrho y$$

with $\varrho := K(\varrho_1, \dots, \varrho_n)$.

In this formulation, Arrow’s theorem therefore implies that in the case of more than two candidates there cannot exist a collective choice function that satisfies all five listed democratic basic rules.

Finally, let us mention that entirely different criteria than the ones analyzed here are naturally possible for decision making methods. An example to this end is the *susceptibility to manipulation*, which, for example, occurs in the method of rank addition: It could be advantageous for a voter to indicate a different preference from his true one—e.g., if his favorite candidate already has many fans, but his second favorite is not so popular, motivating him to place this one at the top of his (public) preference. Methods that do not offer this possibility of maneuvering would, in principle, be preferable—unfortunately, also for this case there exists the theorem of Gibbard and Satterthwaite that provides an impossibility statement of the kind of Arrow’s theorem.

Chapter 5

Scheduling

The—preferably optimal—assignment of resources (time, personnel, tools, etc.) to tasks that need to be carried out, is a challenging and economically significant field of decision making. Examples include the planning of a project, the production flow in a factory, class schedules in a school, and supplying rental car customers with cars.

This section more specifically addresses the modeling of tasks as they arise in the planning of a project or the production in a factory as well as the dependencies between these tasks: For example, the ordering of tasks that are to be performed may only be partially given, or it may occur that certain tasks cannot be performed simultaneously because the necessary resources are available only once. (The problems that arise in class scheduling or in car rental services are, in general, of a completely different nature and as such different techniques are employed.)

The purpose of such models is, in general, one of optimization: Out of all possible orderings, one should be selected that is optimal with respect to a given quality measure—an optimal *schedule* is to be constructed.

Let us first consider the production planning of a factory in which the products pass through a sequence of processing steps that must be completed at a certain point in time. How does one match a machine to a product in a way that the machine is best utilized? When does the production have to start in order to keep the scheduled delivery date? Where shall the optimization of the production process be applied? In the case of complex production processes with numerous processing steps and a large number of different products, these and similar questions can no longer be answered from experience and “try-outs”, but must be imitated on the computer.

An additional difficulty also becomes apparent when we think of the project management. Even if the number of tasks and their dependencies is manageable, it remains difficult to *a priori* estimate the true time that each individual task actually requires. Thereafter, one would like to know how the variation in the actually required times affects the completion of the entire project, i.e., the scheduled delivery date that we promise the customer.

In both cases, there is a lot of money involved, and significant damage can result from a model which does not sufficiently or accurately reflect reality, thus rendering the plans based on it suboptimal.

Surprisingly, however, in many places rather primitive models remain popular. The reason lies with a typical conflict occurring with the selection of a suitable model, that on one hand must be sufficiently powerful to ensure a realistic description of reality but must, on the other hand, be treatable with passable computational effort.

When considering the preparation of time schedules, there occurs an effect which is typical for combinatorial problems: Here, the separation between very simple and very complex problems is slight. Let us consider an example in which we have a project requiring software to be written for the optimization of a production process. Subsequent to this, we receive a call by the customer who informs us that he had previously forgotten to tell us about a special property involving the production steps. This could easily and completely mess up *our* schedule—in short, such a constraint has the potential to entirely change the difficulty of the task: Here, a careful modeling of the task is particularly important. Consideration concerning cost for the treatment of the model appears for the first time in this chapter—this had not played any role in game theory nor in group decision making.

Given the multitude of possible scenarios for time schedules we will restrict our attention to three for this exposition: The first is the *process scheduling* that serves as an example of a simple model in which an optimal schedule can be easily determined via the *critical path method*. In its first extension we assume that the execution times of the individual tasks will be random variables and study their subsequent effects, while the second extension works to model the constraints of resources in the *job-shop model*. Significant in each of these two extensions is that the very simple problem is transformed into one for which under realistic problem sizes an optimal solution can no longer be determined, i.e., one has to go back to heuristics.

With respect to the tools employed, *graphs* will be used for the modeling of tasks and their dependencies (directed graphs, paths, cycles and acyclicity, depth search; see Sect. 2.1). In Sect. 5.2, contributions taken from stochastics (discrete and continuous random variable, joint distributions, expected values, quantiles, see Sect. 2.3) will play a role for the modeling of uncertainty of the duration of a production step—together, these considerations will become the basis for Chap. 6.

5.1 Process Scheduling (Deterministic)

A *process* initially consists only of a set of n tasks A_1, \dots, A_n that have to be performed. For each task A_i , let us first assume that the required *execution time* $t_i \geq 0$ is known (this assumption accounts for the deterministic feature of this model and stands in contrast to the stochastic consideration in the next section); thus, the description of the process includes a list $(t_i)_{1 \leq i \leq n}$ of execution times.

A *schedule* performs two duties: It identifies the task as well as when it is to be performed. In the following, we will define the required conditions to be followed throughout the processing; a schedule that satisfies all conditions is called *admissible*. Furthermore, it is assumed that there exists a *cost measurement* that we can use to assess the schedules—this could, for example, be the total cost for the processing. However, here we simplify things by only considering the elapsed time for the completion of the process.

Presently, the task of scheduling consists of identifying and selecting from all admissible schedules one with minimal costs (or frequently, in practice, a good approximation to it).

The start times for a respective task are also part of the schedule, i.e., supplied as a list of *start times* $(s_i)_{1 \leq i \leq n}$. We assume that the processing of a task cannot be interrupted, hence the start time s_i and *completion time* $c_i = s_i + t_i$ mutually determine one another.

Since up to now we have not defined any constraints, an admissible schedule that completes the process as quickly as possible is easy to find: One begins immediately with all tasks in parallel, i.e., one has $s_i = 0$ and $c_i = t_i$ for all i . With $t_{\max} := \max(t_i)$, the execution time of the longest task, one can also choose $c_i = t_{\max}$ and therefore $s_i = t_{\max} - t_i$ for all i and also obtains an optimal admissible schedule.

In the following, we will impose the compliance with *precedence conditions* onto the schedule. These contribute the statements regarding the order of the execution. A precedence condition between two tasks A_i and A_j is denoted by $A_i \rightarrow A_j$ and implies that task A_j can only be started after A_i has been completed:

$$A_i \rightarrow A_j : \Leftrightarrow c_i \leq s_j .$$

It follows from this definition that with $A_i \rightarrow A_j$ and $A_j \rightarrow A_k$ also $A_i \rightarrow A_k$ is satisfied. Thus, the problem does not change if the relation of precedences in the set of tasks is replaced by its transitive closure—for reasons of clarity, one usually avoids denoting the precedence $A_i \rightarrow A_k$ explicitly.

Additional constraints will not be considered for the moment—we can, in particular, still process an arbitrary number of tasks simultaneously as long as the precedences are observed.

We now demonstrate how the scheduling problem can be represented as a directed graph $G := (V, E)$. Here, the vertex (node) set V consists of the tasks,

$$V := \{A_1, \dots, A_n\} ,$$

which are labeled with the execution times t_i ; each precedence condition $A_i \rightarrow A_j$ defines an edge (A_i, A_j) :

$$E := \{(A_i, A_j) : A_i \rightarrow A_j\} .$$

Furthermore, it is beneficial to include two additional special nodes (which have no impact on the costs of a schedule):

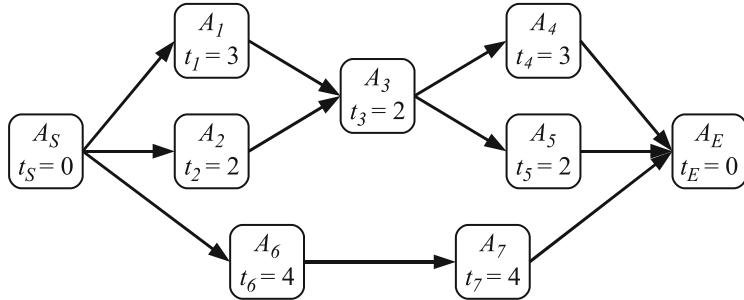


Fig. 5.1 An exemplary graph with seven tasks (plus initial and final vertex)

- An *initial vertex* A_S with number $S := 0$, $t_S := 0$ having precedences $A_S \rightarrow A_i$ for all vertices A_i which until now had no incoming edges $A_j \rightarrow A_i$. In the schedule, this vertex will always be assigned the start time $s_S := 0$.
- A *final vertex* A_E with number $E := n + 1$, $t_E := 0$ and precedences $A_i \rightarrow A_E$ from all vertices A_i which thus far have no outgoing edges $A_i \rightarrow A_j$. In an admissible schedule, we can read off the *total execution time (makespan) c_E* from this vertex which is to be minimized among all admissible schedules.

Figure 5.1 shows the graph for a problem with seven tasks (plus initial vertex A_S and final vertex A_E), the following execution times,

i	1	2	3	4	5	6	7
t_i	3	2	2	3	2	4	4

and the precedence conditions

$$\{A_1 \rightarrow A_3, A_2 \rightarrow A_3, A_3 \rightarrow A_4, A_3 \rightarrow A_5, A_6 \rightarrow A_7\} .$$

Considering the initial vertex, the additional precedences $A_S \rightarrow A_1, A_S \rightarrow A_2, A_S \rightarrow A_6$ are inserted while the final vertex inserts the additional precedences $A_4 \rightarrow A_E, A_5 \rightarrow A_E, A_7 \rightarrow A_E$.

In the following, the *paths* in our graph will play a role, i.e., a list of k vertices (tasks) $A_{i_1}, A_{i_2}, \dots, A_{i_k}$ where for every j , with $1 < j \leq k$, there exists an edge $A_{i_{j-1}} \rightarrow A_{i_j} \in E$, that is:

$$A_{i_1} \rightarrow A_{i_2} \rightarrow \dots \rightarrow A_{i_k} .$$

An important measure is the length of a path $A_{i_1} \rightarrow A_{i_2} \rightarrow \dots \rightarrow A_{i_k}$ which here is defined as the sum of the vertex labels (execution times):

$$l(A_{i_1} \rightarrow A_{i_2} \rightarrow \dots \rightarrow A_{i_k}) := \sum_{j=1}^k t_{i_j} .$$

For our graph, each node lies on at least one path from A_S to A_E ; in every admissible schedule it holds that for every path,

$$c_{i_k} \geq s_{i_1} + \sum_{j=1}^k t_{i_j} = s_{i_1} + l(A_{i_1} \rightarrow A_{i_2} \rightarrow \dots \rightarrow A_{i_k}) .$$

In particular, for every admissible schedule it must hold that c_E is at least as big as the length of the longest path from A_S to A_E —we will see that this bound can be reached, provided there exist admissible schedules at all.

But let us first clarify when there actually exist admissible schedules: If G contains a cycle, i.e., a path $A_{i_1} \rightarrow A_{i_2} \rightarrow \dots \rightarrow A_{i_k} \rightarrow A_{i_1}$, then the condition

$$s_{i_1} \geq c_{i_k} \geq s_{i_1} + \sum_{j=1}^k t_{i_j}$$

is a result which (due to $t_{i_j} \geq 0$) is obviously only satisfiable for the special case $t_{i_j} = 0$, $j = 1, \dots, k$. Therefore, it is hereafter assumed for the set of precedences that G contains no cycles: It is assumed to be a *directed acyclic graph*, abbreviated DAG.

In every DAG one can enumerate the vertices such that $A_i \rightarrow A_j$ occurs only for $i < j$, it is then called *topologically sorted*. In this case, the following method to determine schedules is particularly easy to implement. The topological sorting (in principle, a *depth search* of the graph) is not costly, and as a by-product it simultaneously yields a test of our input data (the precedences) for acyclidicity; the algorithm for this can, e.g., be found in [2] or [13]. The graph in example (Fig. 5.1) is already topologically sorted.

The easiest way to see that there exist admissible schedules under the condition of acyclidicity is to construct one—consequentially, it will turn out to be optimal (with respect to the entire execution time).

The idea for this is simple: Each task is started as early as possible, i.e., at the moment when all prior tasks that need to be performed are completed. The earliest possible time when this can be accomplished is called the *lead time* s'_i . We denote the corresponding completing time by c'_i , and its computation is straight forward:

- $s'_S := c'_S := 0$.
- As long as there remain vertices requiring processing:
 - Find an unprocessed vertex A_i for which each predecessor (i.e., every vertex A_j with $A_j \rightarrow A_i$) has already been processed.
 - In view of the acyclidicity there must exist at least one such vertex. When the vertices are topologically sorted, then one can simply process them in the order $1, 2, \dots, n+1$.
 - Compute

$$s'_i := \max_{j: A_j \rightarrow A_i} c'_j, \quad c'_i := s'_i + t_i .$$

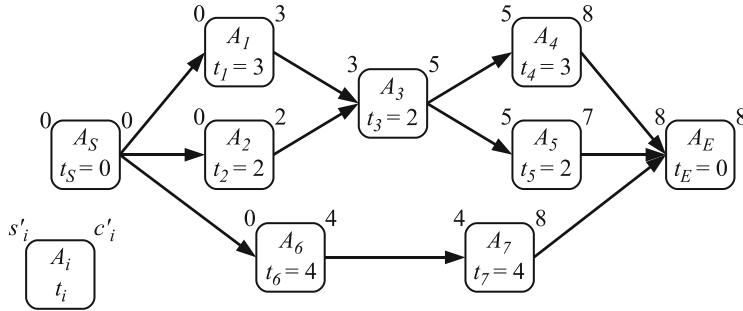


Fig. 5.2 The process of Fig. 5.1 with lead times (earliest possible start times) s'_i and corresponding completions c'_i

Figure 5.2 shows the result of this method for the process shown in Fig. 5.1.

Taking the s'_i as start times, one obtains a schedule which is not only admissible but also minimizes the overall completion time c'_E amongst all admissible schedules: From the computation of the s'_i it follows that on the one hand all precedences $A_j \rightarrow A_i$ are observed, and on the other hand that no task (in particular not A_E) can be started at an earlier time without violating at least one of the precedence conditions.

In particular, it can never result as an improvement in the solution to start a task A_i at a later time than s'_i . Hence, the current optimization task is very simple.

For a better understanding of this scheduling task it is helpful to construct an additional schedule: We now compute the latest possible time c''_i that a task A_i must be completed in order for the optimal entire completion time c'_E to still be observed (meaning that it must then be started at time $s''_i := c''_i - t_i$, and we then call $c'_E - s''_i$ the *remaining time*).

The computation is analogous to the computation of the lead time:

- $c''_E := s''_E := c'_E$.
- As long as there remain vertices to be processed:
 - Find an unprocessed vertex A_i in which every successor (i.e., every vertex A_j with $A_i \rightarrow A_j$) has already been processed. Due to the acyclicity, there must exist at least one such vertex. If the vertices are ordered topologically, one can simply process them in the order $n, n-1, \dots, 2, 1, 0$.
 - Compute

$$c''_i := \min_{j: A_i \rightarrow A_j} s''_j, \quad s''_i := c''_i - t_i .$$

Figure 5.3 shows the result of this method given the process of Fig. 5.1.

Now there exist two possibilities for every vertex (task) A_i :

- $s'_i = s''_i$. In this case the vertex is called *critical* and lies on (at least) one path from A_S to A_E consisting of only critical vertices, and for each edge $A_k \rightarrow A_l$

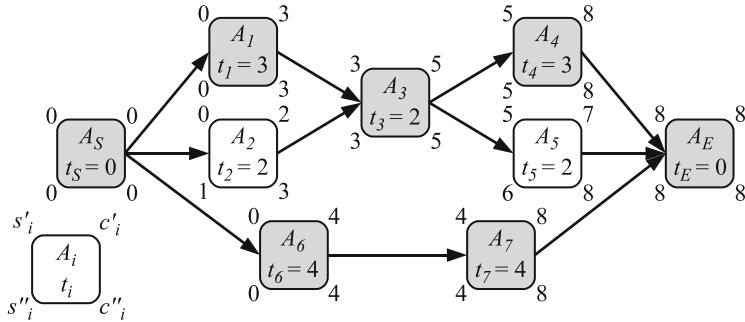


Fig. 5.3 Earliest and latest possible start- and completion times given the process of Fig. 5.1. The critical vertices are marked in gray, here they form two critical paths from A_S to A_E

of this path it holds that $c'_k = s''_l$ (A_l has to directly succeed A_k)—such a path is called a *critical path*. In every optimal schedule, a critical vertex A_i must be assigned the start time $s_i = s'_i = s''_i$. In Fig. 5.3, the critical vertices are marked in gray.

- $s'_i < s''_i$. In this case, the difference $s''_i - s'_i$ is called the *slack*. For an optimal schedule, vertices with slack can have start times within the interval $s'_i \leq s_i \leq s''_i$. (However, this does not imply that one can choose them independently within these intervals since this could violate the precedence conditions between two non-critical vertices.)

One can easily reason that there always exists at least one critical path; if there are several, then they must all have the same path length. The entire completion time results from the length of a critical path in G , i.e., given the example illustrated in Figs. 5.1–5.3, $c_E = 8$. For optimal schedules, the length c_E of a critical path corresponds to the length of a longest path from A_S to A_E (obviously, longer ones cannot exist), this value identified above as a lower bound for the entire completion time is actually reached by these schedules.

This technique (the determination of critical paths for the optimization of the schedule) is called the *critical path method* (CPM). Typical tools that are used here include *Gantt diagrams* in which tasks are represented by bars whose respective lengths are chosen such that, at any given time, one can immediately read off which tasks are scheduled. Another typical tool are *network plans* similar to the one in Fig. 5.3 in which the precedence conditions are stressed more than the processing times. Finally, an alternative description includes graphs modeling the tasks through the edges of the graph that are subsequently labeled with the respective processing times.

A starting point for extensions of the model is the observation that given the current measurement for costs, one cannot distinguish the schedule with the earliest possible start times s'_i from the one with the latest possible start times s''_i —both yield the optimal overall processing time. Yet, arguments for both variants are conceivable:

- Logistical reasons could favor that the tasks start as late as possible, for example, we do not have to store the material quite as long (the “just-in-time production”).
- A more careful person would rather begin the tasks as early as possible—in case that something goes wrong and a task takes a little longer than expected. Our model does not consider this possibility at all; we will analyze the case of processing times that are not known beforehand not until the following section.

The cost measurement for the “overall production time” is therefore questionable since it does not include all aspects for the evaluation of time schedules. A cost measurement that considers variations between the processing times and the actual total cost would be more realistic, however, it is hardly surprising that the determination of optimal schedules then becomes significantly more complex. Although the previous model is quite limited, its optimal solutions are easy to compute which makes the method attractive in practice.

In particular, the method already yields a certain set of instructions in the case of unexpectedly longer processing times: For tasks with slack, one can attempt to use this for a longer processing of the task. However, a delay of a task on a critical path delays the overall completion—if one has the ability to reduce the processing time with increased effort, the vertices on a critical path are then candidates for this additional effort.

We will analyze a completely different extension in Sect. 5.3: There, we consider constraints which result from the fact that certain tasks cannot be simultaneously executed—as is the case, for example, if they require the same machine. Such a model extension will again turn a simple problem into one in which the optimal solutions can only be found under great effort.

5.2 Process-Scheduling (Stochastic)

The assumption that the processing times t_i of the tasks A_i are known beforehand is only realistic for a few applications. The model becomes significantly more flexible if we model the processing times as random variables T_i . During the execution of a process we then consider the processing time that is observed for task A_i as the realization of T_i . The goal will then be to obtain statements from the distributions of the T_i about the distribution of the overall processing time (or about a different cost measurement).

We now give three simple examples of distributions that are sufficient for our purposes (in practical applications one usually uses more complicated distributions, but this does not change anything for the general considerations):

- The previous case of fixed processing times is obtained by a discrete random variable which assumes only one value (T_1 in the diagrams in Fig. 5.4, left).
- If only a finite number of cases with known probabilities can occur at the processing of the task, then we describe the processing time with a distribution as

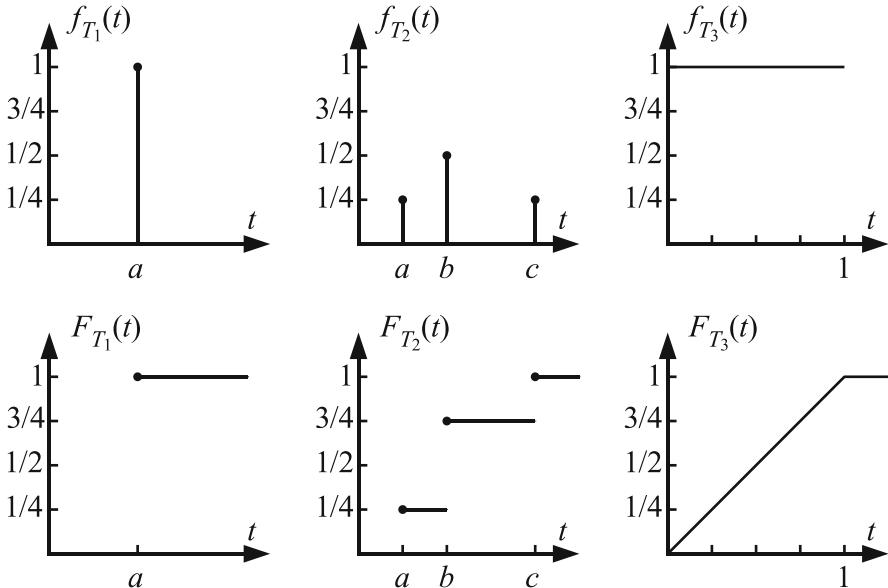


Fig. 5.4 Examples of the distributions of processing times (Densities in the top row, corresponding distribution functions in the bottom row): T_1 corresponds to the deterministic case and always yields the result $T_1 = a$, T_2 yields three different values a, b , and c with probabilities $1/4, 1/2$ and $1/4$, respectively, T_3 is a continuous random variable—it is uniformly distributed on $[0, 1]$, i.e., the probability that the realization T_3 falls within an interval $[a, b]$ ($0 \leq a \leq b \leq 1$) is $P(T_3 \in [a, b]) = b - a$

for T_2 in Fig. 5.4, middle—here, for an example, with three possible processing times a, b and c .

- The processing time can also be a continuous random variable; although the uniform distribution, as illustrated for T_3 in Fig. 5.4, right, for the interval $[0, 1]$, is not very realistic, it is sufficient for our considerations.

Since we have several tasks, in principle we would need the joint distribution of the T_i with the distribution function

$$F_{T_1, T_2, \dots, T_n}(t_1, t_2, \dots, t_n) = P(T_1 \leq t_1, T_2 \leq t_2, \dots, T_n \leq t_n),$$

which also describes the dependencies between the T_i : Two tasks A_i and A_j could, for example, require joint resources so that a delay in A_i will in consequence lead to a high probability for a delay also in A_j .

For simplification, however, we assume in the following that the T_i are independent—although this is not very realistic, the joint distribution is easily stated since for independent random variables it holds that:

$$P(T_1 \leq t_1, T_2 \leq t_2, \dots, T_n \leq t_n) = \prod_{i=1}^n P(T_i \leq t_i).$$

Under this assumption, an optimal strategy (in the sense of minimal overall processing time) is easily stated since for *each* realization (t_1, t_2, \dots, t_n) of the T_i , the previous strategy “to start each task as early as possible” still remains optimal. (In the case of processing times that are not independent, this is not necessarily the case; here, the situation can, for example, arise in which one obtains information from the observed processing time t_i of a task about the processing times of the upcoming tasks.)

For this strategy, we can compute the overall processing time c_E that is dependent on the t_i as before (length of a critical path; the vertices that are critical obviously depend on the realizations t_i). We then can interpret c_E as a realization of a random variable $C_E = f(T_1, T_2, \dots, T_n)$ whose distribution, in principle, can be computed from the distributions of the T_i .

For example, this would answer questions such as “How long does the processing take on average?” (i.e., for the expected value $E(C_E)$) or “At what time will the processing be completed with 95 % probability?” (i.e., for the quantiles of the distribution of C_E).

Unfortunately, in practice the determination of the distribution of C_E is much too expensive. For example, let us consider the processing times having distributions T_2 as in Fig. 5.4 each with three possible values: For n such distributed T_i it would indeed be possible to compute, as the length of a critical path, the overall processing time c_E for each of the 3^n possible combinations of the realizations t_i ; the corresponding probability for the occurrence of (t_1, t_2, \dots, t_n) is a result from the distributions of the T_i under the assumption of independence. For problems with a few hundred tasks that are of practical relevance, this becomes much too costly simply because of the subsequent large number of combinations.

A simplification of the problem that is often used is to assume the expected values (average processing time) $E(T_i)$ as processing times t_i and to use the resulting c_E as an estimate for the average overall processing time $E(C_E)$.

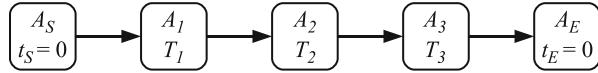
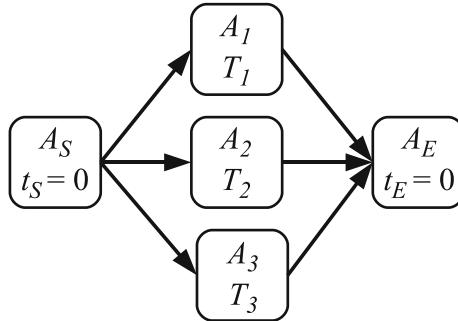
We will next analyse the suitability of this procedure with the help of two configurations: strictly serial processing on the one hand and strictly parallel processing on the other.

Strictly Serial Processing

We speak of strictly serial processing if the precedence conditions already uniquely determine the order—without loss of generality, let the tasks be numbered accordingly so that the precedences $A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_n$ are satisfied (see Fig. 5.5).

The optimal schedule places the tasks directly one after the other. This yields the overall processing time,

$$C_E = \sum_{i=1}^n T_i .$$

**Fig. 5.5** Strictly serial processing**Fig. 5.6** Strictly parallel processing

In view of the linearity of the expected value,

$$E(C_E) = E \left(\sum_{i=1}^n T_i \right) = \sum_{i=1}^n E(T_i) ,$$

it holds that the replacement of t_i by $E(T_i)$ is correct in this situation. Similar applications will be visited and further analyzed in Chap. 6.

Strictly Parallel Processing

The opposite of strictly serial processing is strictly parallel processing: Here, there exist no precedences among the tasks A_i except those at the initial vertex $A_S \rightarrow A_i$ and those at the final vertex $A_i \rightarrow A_E$ (Fig. 5.6).

The optimal schedule sets $s_i := 0$ for all $i = 1, \dots, n$; the overall processing time depends only on the longest running task:

$$C_E = \max_{1 \leq i \leq n} T_i .$$

In contrast to the direct summation we applied to the strictly serial processing case, finding the maximum here, in general, is not interchangeable with the expected value. We have in this case only *Jensen's inequality* holding,

$$E(C_E) \geq \max_{1 \leq i \leq n} E(T_i) .$$

If one now replaces the realizations t_i of the processing times by $E(T_i)$ and uses

$$\tilde{c}_E := \max_{1 \leq i \leq n} E(T_i)$$

as an estimate for the average overall processing time, the inequality $E(C_E) \geq \tilde{c}_E$ then guarantees that one never computes an estimate that is too pessimistic, i.e., one that anticipates the completion later than the actual expected value. Unfortunately, this estimate is, in general, clearly too optimistic (\tilde{c}_E is much smaller than $E(C_E)$): The overall expected processing time is systematically underestimated. We all know this phenomenon from major public projects—from relocating the German Federal Parliament from Bonn to Berlin and the introduction of the highway toll system to the construction of a portion of the highway. Naturally, these tasks as a whole, do not satisfy the “strictly parallel processing” structure, but they do contain subprocesses having a similar structure in which the underestimation strikes (i.e., the amount in which the estimated completion time is exceeded).

An example: Let all T_i be independent and uniformly distributed on $[0, 1]$. Then, for the distribution function of C_E it follows that

$$\begin{aligned} F_{C_E}(t) &= P(C_E \leq t) = P(T_i \leq t, 1 \leq i \leq n) \\ &= \prod_{i=1}^n P(T_i \leq t) = t^n , \end{aligned}$$

with density function,

$$f_{C_E}(t) = F'_{C_E}(t) = nt^{n-1}$$

and its expected value is

$$E(C_E) = \int_0^1 t \cdot f_{C_E}(t) dt = \frac{n}{n+1} .$$

For increasing n , $E(C_E)$ converges toward 1 (Fig. 5.7). Conversely, it holds that each $E(T_i) = 1/2$. Therefore, $\tilde{c}_E = 1/2$ and it is clear to see that it indeed underestimates $E(C_E)$ as soon as more than one task is to be processed: Thus, the expected processing times $E(T_i)$ typically do not provide a good estimation for the expected overall processing time $E(C_E)$.

Now, one could attempt to make use of the expected processing times $E(T_i)$ to, at least, determine critical paths for the process having these processing times. Under the assumption that with high probability these will also be critical for a realization of the T_i , these would, for example, be suitable starting points for an optimization of the process. In general, however, this assumption is wrong as one can see for the following simple process having two parallel tasks A_1 and A_2 . For each of the two tasks, the processing time T_i can only assume one of two values each. For A_1 , these

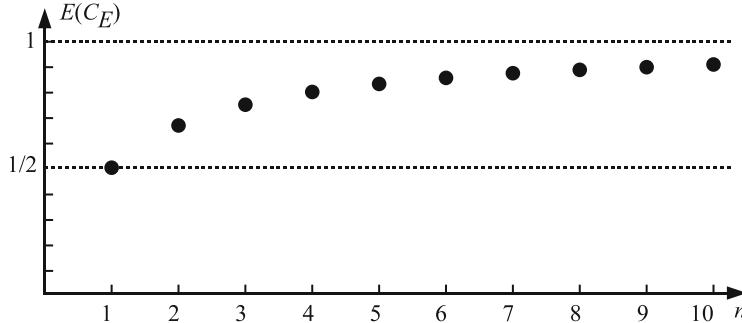


Fig. 5.7 Average overall processing time $E(C_E) = n/(n + 1)$ of n strictly parallel tasks; processing times T_i independently $[0, 1]$ -uniformly distributed

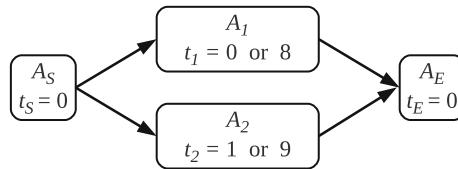


Fig. 5.8 Strictly parallel processing, T_1 assumes the values 0 and 8 each having probability $1/2$, while T_2 assumes the value 1 with probability $3/4$ and the value 9 with probability $1/4$

are the values 0 and 8 which each occur with a respective probability of $1/2$,

$$P(T_1 = 0) = P(T_1 = 8) = \frac{1}{2},$$

i.e., $E(T_1) = 4$, and for A_2 , the values 1 and 9 occur with probabilities $3/4$ and $1/4$, respectively,

$$P(T_2 = 1) = \frac{3}{4}, \quad P(T_2 = 9) = \frac{1}{4},$$

i.e., $E(T_2) = 3$ (Fig. 5.8). Again, let T_1 and T_2 be independent.

In view of $E(T_1) > E(T_2)$, one would expect that if one only considers the expected values, there is a high probability that $A_S \rightarrow A_1 \rightarrow A_E$ is a critical path. However, this is only the case if the realization of T_1 yields the value 8 and the realization of T_2 yields the value 1, i.e., with a probability of $3/8 < 1/2$ —for the other three combinations, it holds that $T_2 > T_1$. Therefore, the path $A_S \rightarrow A_2 \rightarrow A_E$ has probability $5/8 > 1/2$ for being a critical path.

Additional material concerning problems and solution approaches for stochastic process scheduling—in particular about the frequently used technique PERT (Program Evaluation and Review Technique)—can, for example, be found in Fulkerson [24], Adlakha and Kulkarni [1] and Möhring [41, 42].

5.3 Job-Shop Problems

The next extension of the model concerns itself with the treatment of constraints originating from limited resources: One has reached a point in which it is no longer possible to process arbitrarily many jobs in parallel.

The particular model we will analyze here is the *job-shop model without recirculation*: A job A_i breaks down into n_i subjobs $A_{i,j}$, $j = 1, \dots, n_i$ which have to be processed in this particular order and it is given that subjob $A_{i,j}$ has duration $t_{i,j}$ and requires a machine $m_{i,j}$ to be processed ($1 \leq m_{i,j} \leq m$ with m the number of available machines). For each machine, only one (sub-)job can be processed at any given time so that subjobs requiring the same machine must be sequenced into an order. Jobs of this type in which, specifically, the order of the subjobs identified within the processing of a job is prescribed, are called *job-shop problems*. Additionally, in the following it is assumed that every job requires every machine at most once: There is no *recirculation*.

A job can then be denoted as

$$A_i = \begin{pmatrix} m_{i,1} & m_{i,2} & \dots & m_{i,j} & \dots & m_{i,n_i} \\ t_{i,1} & t_{i,2} & \dots & t_{i,j} & \dots & t_{i,n_i} \end{pmatrix},$$

the requirement “no recirculation” reads as $m_{i,j} \neq m_{i,j'}$ for $j \neq j'$.

As an example, we consider a problem with three tasks and three machines:

$$A_1 := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 4 & 2 \end{pmatrix}, \quad A_2 := \begin{pmatrix} 2 & 1 \\ 3 & 1 \end{pmatrix}, \quad A_3 := \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}. \quad (5.1)$$

From the perspectives of the machines, this example problem presents itself as follows: Machines 1 and 2 are required for all three jobs, whereas the jobs A_1 and A_3 require that machine 1 is to be used first before machine 2, while for job A_2 it is vice versa. Machine 3 is used only once, namely at the end of A_1 .

Our task is once again the construction of an admissible schedule that minimizes the overall processing time. Here, a schedule is called admissible if

- no subjob $A_{i,j}$ is started before its predecessor $A_{i,j-1}$ is finished (and their respective first subjobs $A_{i,1}$ are started no earlier than at $t = 0$),
- and at no time can two or more subjobs requiring the same machine be scheduled.

Using the precedence graph as given for the process scheduling (Sect. 5.1), the first condition can be modeled directly; however, for the second condition, it is necessary to further extend the model.

Firstly, we concern ourselves with the order within a job: To this end, we construct a precedence graph in which the nodes no longer stand for the jobs A_i but for the subjobs $A_{i,j}$. There are edges $A_{i,j-1} \rightarrow A_{i,j}$ for all $i = 1, \dots, n$, $j = 2, \dots, n_i$, an initial vertex A_S having edges $A_S \rightarrow A_{i,1}$, $i = 1, \dots, n$, and a final vertex A_E having edges $A_{i,n_i} \rightarrow A_E$, $i = 1, \dots, n$. Process scheduling on this graph

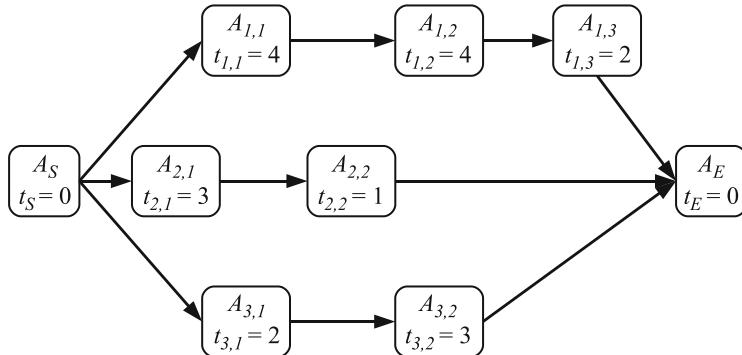


Fig. 5.9 Here, given the example problem (5.1)—conjunctive edges model the successive execution of the subjobs occurring within a job

would guarantee that for each job A_i the subjobs $A_{i,j}$ are processed one after the other and in the correct order $j = 1, 2, \dots, n$; the constraint considering machine availability has not yet been accounted for at all. In order to distinguish them from the edges which will be added next, here we call the previously introduced edges *conjunctive edges*, refer to Fig. 5.9.

Next, we turn to the condition requiring that each machine k must process all of its respective subjobs,

$$M(k) := \{A_{i,j} : m_{i,j} = k\}$$

that are assigned to it sequentially. This condition is satisfied if and only if for every pair of subjobs $A_{i,j}, A_{i',j'} \in M(k)$, the processing times do not overlap. This, in turn, is satisfied if and only if an additional precedence edge $A_{i,j} \rightarrow A_{i',j'}$ or $A_{i',j'} \rightarrow A_{i,j}$ can be added to the graph and permits the schedule to remain admissible for its augmented graph—there are only the two possibilities “ $A_{i,j}$ before $A_{i',j'}$ ” and “ $A_{i',j'}$ before $A_{i,j}$ ”.

Hence, the machine capacity can be integrated into the precedence graph: For every machine k and all $A_{i,j}, A_{i',j'} \in M(k)$, we must add one of the two edges $A_{i,j} \rightarrow A_{i',j'}$ or $A_{i',j'} \rightarrow A_{i,j}$; it is an automatic consequence that an admissible schedule designated for the augmented graph also satisfies the constraint. The only problem is: For each case, which of the two edges should we take?

In order to analyze this, at first we include *all* such edges: For all machines $k = 1, \dots, m$ and all $A_{i,j}, A_{i',j'} \in M(k)$, a *disjunctive edge* is added which is the pair of directed edges $(A_{i,j} \rightarrow A_{i',j'}, A_{i',j'} \rightarrow A_{i,j})$, see Fig. 5.10. For each k , the vertices in $M(k)$ now form a clique (a subset of the vertex set which forms a complete graph).

As long as there exists at least one disjunctive edge, there obviously exist cycles and therefore no longer optimal time schedules. For each edge pair, we now have

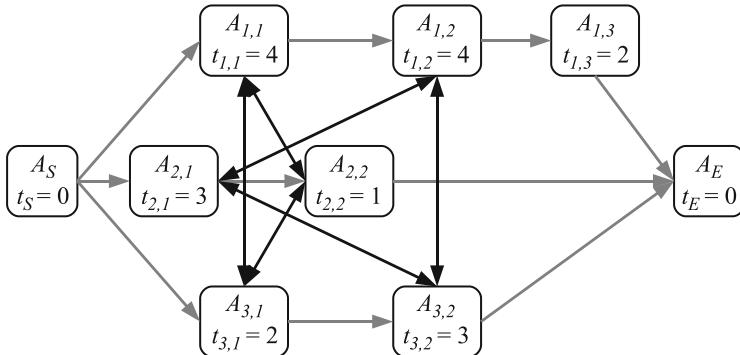


Fig. 5.10 Disjunctive edges—here, for the example problem (5.1), each edge pair is represented by a *black double arrow*—model the necessity to process the subjobs associated to a machine successively

to choose only one of them: An edge set that contains exactly one edge of every disjunctive edge is called *disjunctive edge assignment*.

A disjunctive edge assignment is called admissible if the resulting precedence graph is acyclic. This is not, however, satisfied automatically—if, for example, three or more subjobs require the same machine, there are obviously disjunctive edge assignments which are already cyclic on a single machine. However, one can easily construct examples for which a disjunctive edge assignment is acyclic for every single machine but yet, together with the conjunctive edges, cycles occur.

We obtain a disjunctive edge assignment by starting a subjob $A_{i,j}$ when its predecessor $A_{i,j-1}$ (assuming there is one) is finished and the required machine is available; if, at a given time, several subjobs are possible, we then choose an arbitrary subjob for each of the machines in question.

Therefore, given our problem setting there always exist admissible disjunctive edge assignments—but which of these ones are optimal with respect to the overall processing time (the one we just constructed will typically not be optimal)? Since, for a given admissible disjunctive edge assignment, an optimal admissible schedule is easily determined by means of CPM, one can again reduce the problem to an optimization problem over all admissible disjunctive edges, i.e., to a discrete optimization problem.

In principle, for k disjunctive edges one could try out all 2^k disjunctive edge assignments—although this is possible in finite time, it is very costly for practical problem sizes.

In view of the exponentially increasing number of possibilities, it would be quite helpful in the sense of *divide et impera* if one could extract a subproblem that can be treated separately. This is indeed possible for process scheduling: If the (sub-)jobs are ordered topologically, one can assign their start times using the following ordering scheme; order so that jobs that are further down the row do not have any influence on the leading ones.

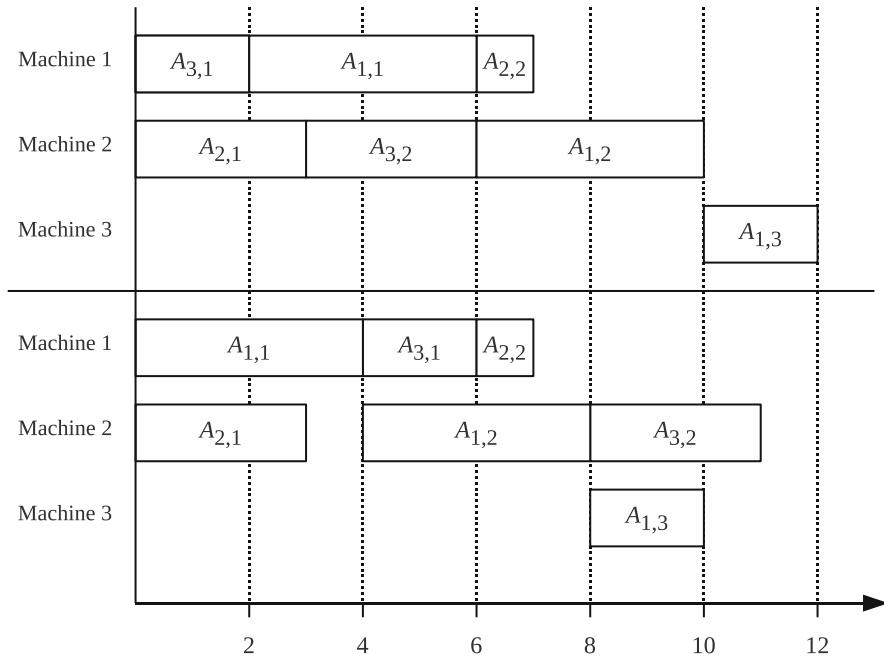


Fig. 5.11 Process flow for two disjunctive edge assignments in the example problem (5.1). The choice $A_{3,1} \rightarrow A_{1,1} \rightarrow A_{2,2}$ and $A_{2,1} \rightarrow A_{3,2} \rightarrow A_{1,2}$ minimizes the overall processing time for the entire problem without $A_{1,3}$, however, the choice $A_{1,1} \rightarrow A_{3,1} \rightarrow A_{2,2}$ and $A_{2,1} \rightarrow A_{1,2} \rightarrow A_{3,2}$ is more favorable

Since in example problem (5.1) no subjob (except A_E) has to be placed after $A_{1,3}$, one could try to at first search for an optimal schedule while disregarding $A_{1,3}$. Another motivation for this approach is that $A_{1,3}$ is the only subjob that requires machine 3, i.e., it plays no role for the side constraints of the machine assignments and therefore appears to be decoupled from the other subjobs.

For the reduced problem (5.1) without $A_{1,3}$

$$A_1 := \begin{pmatrix} 1 & 2 \\ 4 & 4 \end{pmatrix}, \quad A_2 := \begin{pmatrix} 2 & 1 \\ 3 & 1 \end{pmatrix}, \quad A_3 := \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \quad (5.2)$$

one is easily convinced by trial and error that one needs to begin with $A_{2,1}$ on machine 2 and with $A_{3,1}$ on machine 1; this yields the disjunctive edge assignment

$$\{A_{3,1} \rightarrow A_{1,1}, A_{1,1} \rightarrow A_{2,2}, A_{3,1} \rightarrow A_{2,2}, \\ A_{2,1} \rightarrow A_{3,2}, A_{3,2} \rightarrow A_{1,2}, A_{2,1} \rightarrow A_{1,2}\}$$

and the production flow in Fig 5.11 (upper part) yield an overall production time (without $A_{1,3}$) of 10.

If one now once more considers $A_{1,3}$, one must no longer reuse this assignment of machines 1 and 2—the lower part of the figure shows an optimal schedule which appears nonintuitive and unreasonable from the local perspective of machines 1 and 2 (in particular since machine 2 is left idle for a time unit), however, this sequencing does indeed reduce the overall processing time.

Here, the disjunctive edge assignment is

$$\{A_{1,1} \rightarrow A_{3,1}, A_{3,1} \rightarrow A_{2,2}, A_{1,1} \rightarrow A_{2,2}, \\ A_{2,1} \rightarrow A_{1,2}, A_{1,2} \rightarrow A_{3,2}, A_{2,1} \rightarrow A_{3,2}\}.$$

With respect to discrete optimization problems, it is typical that strategies are rarely found in which problems can be separated and an optimal solution of the entire problem can be constructed from optimal solutions of the subproblems with reasonable expectations for the cost. Therefore, it is often the case that the solution to such problems is very expensive. (Naturally, this excludes our small example problem in which all $2^6 = 64$ possibilities can be easily tried out; it turns out that there are 22 admissible disjunctive edge assignments which are listed in the illustration given on page 85: the subjobs corresponding to A_1 are red, those corresponding to A_2 green and those corresponding to A_3 blue.)

With strategies such as *branch and bound* one can oftentimes restrict the search space significantly by showing that certain parts of the search space cannot contain optimal solutions. However, even in this case the cost typically grows exponentially with respect to the problem size.

Fortunately, in practical applications it is usually not necessary to find an optimal solution. Thus, in most cases heuristic methods can be used (e.g. the *shifting-bottleneck method*) whose solutions are determined in significantly less time and, as well, are usually not much worse than an optimal schedule.

5.4 Further Scheduling Problems

Two minor extensions can be integrated into our job-shop model in a straightforward way:

- Interdependencies between jobs (“start job j only if job i is completed”) have so far not been considered but are easy to model through the addition of conjunctive edges (here, $A_{i,n_i} \rightarrow A_{j,1}$); with further conjunctive edges, such additional dependencies between the subjobs of two different jobs can be represented.
- So far, we used disjunctive edges only to connect a set of subjobs (vertices) to a clique. Such a restriction to cliques is rather artificial—and is usually violated by the method that executes the disjunctive edge assignment when disjunctive edges are successively replaced by conjunctive edges. Without further consideration, one can place disjunctive edges wherever two subjobs cannot be processed simultaneously.

In a real world application, however, there will occur additional conditions which will render this model useless. In particular, it could be the case that machines allow the parallel processing of a certain number of subjobs (this can simply be an intermediate storage of limited capacity), or there could be temporal requirements for the processing (“there must pass at least 2 h but at most 4 h between the lacquering and the post-processing”), or many more scenarios that should or must be considered. Therefore, even though the job-shop model applies only in special cases as a complete description to a real life process, it nonetheless has its justification as a starting point for modeling in which additional elements are to be added.

Corresponding issues also hold for an important specialization of the job-shop model: A *flow-shop* is a job shop in which all subjobs pass through the control units (machines) in the same order. At once, this specialization obviously simplifies the schedule construction significantly while also allowing us to employ more complicated models for the individual control units.

For large systems, one often uses models of which parts can be modeled as job shops or flow shops and these are used as building blocks for the entire system. In this case, there does not exist a global optimization which finds a schedule that is optimal for the entire system, so that in general one only finds solutions that are (hopefully only a little) worse than the absolute optimum. However, the computational costs are significantly reduced. In Chap. 9 we consider queuing systems and queuing networks, and we will again encounter models of this type in which a part of the system appears on the exterior as a control unit which is characterized by a few parameters.

A detailed treatment of scheduling problems as well as additional problems on optimal resource usage can, for example, be found in [47].

Chapter 6

Wiener Processes

With time schedule optimization, we have already considered problems in which the processing times were no longer known beforehand and were therefore modeled by random variables: Instead of a fixed value, we assumed a—in principle arbitrary—distribution. As a result, the quantities that are observed in the model (such as the overall completion time) are also random variables, and we are interested in statements regarding their distributions. This model can be extended for situations in which the observed quantities are obtained as the sum of a large number of independent random variables so that a transition from a discrete to a continuous model is advisable (here: a *Wiener process*).

This opens up interesting new application areas, such as mathematical finance. In this chapter, we will derive a simple model for stock prices as an example, the so-called *Black–Scholes model*. These considerations shall complete the part of this text, “Games—decisions—scheduling”, and illustrate how mathematical models can suddenly reappear in a completely different context. On the other hand, it is often possible to approach similar problem settings from entirely different perspectives—for example, the model for the development of a capital investment which is introduced in this chapter also has a close connection to the models for population dynamics discussed in Chap. 10 but will make use of entirely different tools. Thus, this chapter is considered to be interconnected with the others in the sense that it builds upon Chap. 5, and, in particular, the topics of Sect. 5.2 will be continued. Naturally, the tools used in stochastics (see Sect. 2.3) play a significant role in this chapter—we will have to deal with both discrete distributions (Bernoulli distribution, binomial distribution) as well as continuous distributions (normal distribution and its quantiles) and the transition between these two distribution worlds (key word asymptotics, Sect. 2.3.4).

6.1 From the Bernoulli Experiment to the Normal Distribution

We begin by considering the serial processing of n subjobs as given in Sect. 5.2, Fig. 5.5. In the following, we will determine the distribution governing the overall processing time—since in Sect. 5.2, we were only interested in the expected value (and had seen that in the case of serial processing, one can always compute it as the sum of the individual processing times), therefore, we will now require additional calculations.

We again assume that the processing times T_i are *independent*, and in addition, we assume that they are also *identically distributed* (*iid*: independent, identically distributed). One can therefore think of a random generator that yields a realization of T_i for every task A_i ; its result is independent of the previous values.

Initially, let the distribution of the T_i be very simple: T_i can assume two values $\mu - \sigma$ and $\mu + \sigma$ ($\mu, \sigma \in \mathbb{R}, \sigma > 0$), each having probability $1/2$, respectively. Thus, the values lie with distance σ symmetric about μ , and the random generator which determines whether the bigger or smaller value is chosen could simply be realized by tossing a coin. For the expected value and the variance of T_i , it holds that $E(T_i) = \mu$ and $\text{Var}(T_i) = \sigma^2$ (which also motivated the naming conventions μ and σ).

As long as one only considers processing times that cannot be negative, one requires that $\mu \geq \sigma$. For our considerations, however, this is not necessary and will therefore not be assumed in the following.

We now determine the distribution of the overall processing time; shifting away from the example application “Scheduling”, we now focus our attention on the sum of all T_i as our target quantity. In doing so, it is no longer denoted as C_E as in Sect. 5.2 but simply as S :

$$S = \sum_{i=1}^n T_i .$$

In order to determine its distribution, we consider the random variables

$$\tilde{T}_i := \frac{T_i - (\mu - \sigma)}{2\sigma} ,$$

which result from the T_i through the linear transformation

$$t \mapsto \frac{t - (\mu - \sigma)}{2\sigma} .$$

They can assume the values 0 and 1 (each with respective probability $1/2$), i.e., they are *Bernoulli*-distributed with parameter $p = 1/2$.

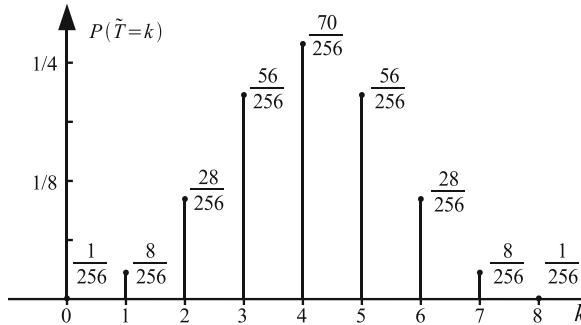


Fig. 6.1 Density of the binomial distribution (parameter $n = 8, p = 1/2$)

The sum of n Bernoulli-distributed random variables with parameter p results in a binomial distribution with parameters n and p :

$$\tilde{S} := \sum_{i=1}^n \tilde{T}_i \sim \mathcal{B}(n, \frac{1}{2}).$$

Hence, it holds that

$$P(\tilde{S} = k) = \binom{n}{k} p^k (1-p)^{n-k} = 2^{-n} \binom{n}{k}.$$

Figure 6.1 shows the density of the binomial distribution for $n = 8$ and $p = 1/2$.

Thus, we know the distribution of \tilde{S} , and in view that

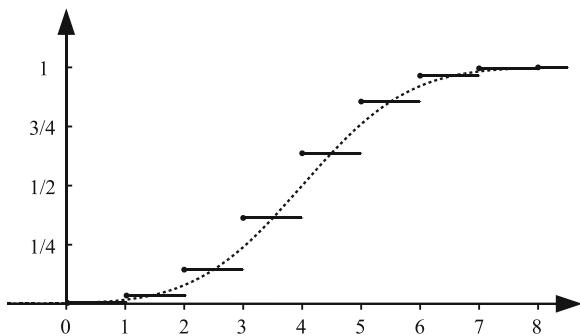
$$S = 2\sigma \tilde{S} + n \cdot (\mu - \sigma), \quad (6.1)$$

the task to determine the distribution of S is in principle solved. However, the binomial distribution is not practical for large n , one rather approximates it by a normal distribution.

For $\tilde{S} \sim \mathcal{B}(n, p)$, i.e., with expected value $E(\tilde{S}) = np$ and variance $\text{Var}(\tilde{S}) = np(1-p)$, for a sufficiently large n , one can replace the (discrete) binomial distribution by a (continuous) normal distribution $\mathcal{N}(np, np(1-p))$ with the same expected value and the same variance (De Moivre–Laplace theorem). Figure 6.2 illustrates an example for $n = 8$ and $p = 1/2$. (The figure suggests that the approximation can be improved by choosing a normal distribution with expected value $np - 1/2$, i.e., shifting the curve to the left by $1/2$. However, for large n , this is not significant.) For $\hat{S} \sim \mathcal{N}(n/2, n/4)$ it follows that $P(\tilde{S} \leq t) \approx P(\hat{S} \leq t)$.

In view of (6.1) and the transformation rule for normal distributions under linear transformations (for $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$ it holds that $Y := aX + b \sim \mathcal{N}(a\mu_X + b, a^2\sigma_X^2)$), we obtain as a consequence that the distribution of S satisfies

Fig. 6.2 Distribution functions of the binomial distribution (parameter $n = 8, p = 1/2$) and the approximating normal distribution $\mathcal{N}(4, 2)$: The continuous distribution function for $\mathcal{N}(4, 2)$ approximates the jump function for $\mathcal{B}(8, 1/2)$



For large n , S is approximated well by a normal distribution $\mathcal{N}(n\mu, n\sigma^2)$.

Using the normal distribution as an approximation to the binomial distribution can be exploited in both directions:

- If one does not have normally distributed random numbers at hand, one can simply (but relatively expensively) approximate them as the sum of Bernoulli-distributed random numbers (coin toss).
- On the other hand, if normally distributed random numbers are available, one can use them to approximate a binomial distribution without having to perform many Bernoulli experiments.

6.2 Normally Distributed Input Parameters

Since normal distributions also turn out to be good approximations under significantly more general conditions—this results e.g. from the central limit theorem—we now change the scenario such that the input parameters (in the previous example, these were the processing times of the subprocesses) are assumed to be normally distributed random variables

$$T_i \sim \mathcal{N}(\mu, \sigma)$$

($\mu, \sigma \in \mathbb{R}, \sigma > 0$).

This model is useful even in cases such as those described in the previous section in which the individual input parameters T_i were definitely not normally distributed; one simply merges sufficiently many individual steps so that the resulting sum can be assumed to be normally distributed.

If now we assume the T_i to be (μ, σ) -normally distributed, then the distribution of its sum simply results from the addition theorem for normal distributions:

$$S := \sum_{i=1}^n T_i \sim \mathcal{N}(n\mu, n\sigma^2).$$

From a practical point of view, the second parameter (the variance) of S deserves special attention: At first glance, it is not surprising that it grows proportionally to n —as does the expected value—but this is a frequent source for errors from computations when the variance $n\sigma^2$ is confounded with the standard deviation $\sqrt{n}\sigma$: This only grows proportionally to \sqrt{n} .

For the realization of this distribution through a computer program it may appear as an obstacle that normally distributed random variables are harder to generate as was the case in Sect. 6.1 with the coin toss. If computing times are not of consequence, then approximation using the binomial distribution as described above is quite practical as a quick-and-dirty solution, but there exist significantly more efficient methods to generate normally distributed random numbers. For example, in Maple there exist taylorored functions, and the command

```
stats[random,normald[mu,sigma]]()
```

produces realizations of a (μ, σ^2) -normally distributed random variable.

In comparison, an essential difference to the model in Sect. 6.1 having the discrete distribution of the T_i results from the fact that in the normal distribution both parameters are real numbers—this is in direct contrast to the integer parameter n of the binomial distribution.

By doing this, we can again think of each of the random variables $T_i \sim \mathcal{N}(\mu, \sigma^2)$ as the sum of normally distributed random variables, e.g., of k independent $\mathcal{N}(\frac{\mu}{k}, \frac{\sigma^2}{k})$ -distributed random variables. But if one again considers the processing times, this then means that every task would need to be decomposed into k equivalent subtasks. For our new model, this is possible up to an arbitrary fineness which motivates the next step to transition from a model that is discrete in its course (n individual tasks with their processing times given sequentially) to a continuous model—there will be a real axis on which an arbitrary interval can be interpreted as a task. The assignment of tasks (e.g. the fineness of the decomposition into subtasks), however, does not play any role for the observed quantity S .

6.3 Wiener Processes

We begin by once more reconsidering the discrete process as in the previous sections, only that now we assume an infinite family $\{T_i, i \in \mathbb{N}\}$ of random variables instead of requiring them to be finite in number. Here, the T_i shall again be assumed to be independent and identically distributed (e.g. normally distributed).

Instead of the overall sum, we now consider the partial sums

$$S_k := \sum_{i=1}^k T_i ,$$

which serve to provide the contribution from steps 1 to k , $k \in \mathbb{N}$, where we set $S_0 := 0$.

The difference $S_l - S_k$ ($l > k \in \mathbb{N}_0$) thus corresponds to the contribution of steps $k+1, \dots, l$. In the example ‘‘processing times’’, this would present the time needed by these subjobs, and given that a negative difference would be unrealistic, we proceed to our next example in which we think of S_k as the value of a monetary investment on day k —in this case, both a gain ($S_l - S_k > 0$) but (unfortunately) also a loss ($S_l - S_k < 0$) between the days k and l are realistic.

Although the S_k are not independent (the previously accrued processing time/the stock price today has a large influence on the probabilities after the next step), the differences $S_l - S_k$ and $S_{l'} - S_{k'}$ for non-overlapping intervals $0 \leq k < l \leq k' < l' \in \mathbb{N}_0$, however, are independent (since the T_i are independent and the two partial sums employ disjoint subsets of the T_i).

If we assume the T_i to be normally distributed, we can think of a step to be divided into arbitrarily fine substeps whose contribution is also normally distributed. This permits a direct transition to a continuous process, only that then there no longer are individual T_i but only the individual terms W_t , $t \in \mathbb{R}_+$ of the partial sums S_k , $k \in \mathbb{N}_0$:

Definition 6.1 (Wiener process). A (standard-)Wiener process (after Norbert Wiener, 1894–1964) is a family of random variables $\{W_t, t \in \mathbb{R}_+\}$ with

- 1) $W_0 = 0$.
- 2) For all $0 \leq s < t$ it holds that $W_t - W_s \sim \mathcal{N}(0, t - s)$.
- 3) $W_t - W_s$ and $W_{t'} - W_{s'}$ are independent for all nonoverlapping pairs of time intervals $0 \leq s < t \leq s' < t'$.

While in the discrete case, the difference $S_l - S_k$, $l > k \in \mathbb{N}_0$ results from the input parameters T_{l+1}, \dots, T_k , we now see the difference $W_t - W_s$, $t > s \in \mathbb{R}_0$ continuously within the interval $[s, t]$. In particular, since the distribution of these differences $W_t - W_s$ are known, the (normal-)distribution of the W_t is thus also determined. Considering the family $\{W_t\}$, if one now only considers the W_t with integer $t \in \mathbb{N}_0$, one then obtains the discrete family $\{S_k\}$ in which the T_i are $(0, 1)$ -normally distributed—property (2) states that $W_t - W_{t-1} \sim \mathcal{N}(0, 1)$.

A generalization of the standard Wiener process required in the following is the so-called *Wiener process with drift* $\mu \in \mathbb{R}$ and *volatility* $\sigma \in \mathbb{R}$, $\sigma > 0$. It arises from a standard Wiener process for a time interval of length $t - s = 1$ in which the standard deviation is now σ and, as a whole, it is overlayed with a deterministic growth (growth rate μ , i.e., linear increase/decrease of $\mu \cdot t$). In the definition, requirement 2) is therefore replaced by

- 2') For all $0 \leq s < t$ it holds that $W_t - W_s \sim \mathcal{N}(\mu \cdot (t - s), \sigma^2 \cdot (t - s))$.

In the following, we will consider Wiener processes with drift and volatility; the restriction to those W_t with integer t corresponds here to the discrete process $\{S_k\}$ with (μ, σ^2) -normally distributed random variables T_i .

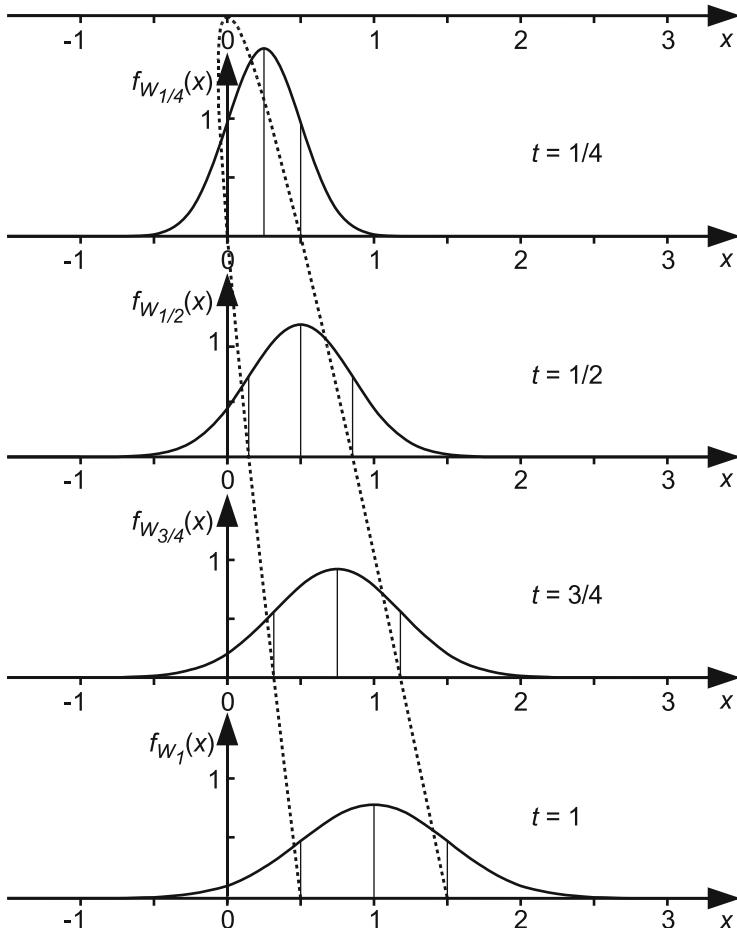


Fig. 6.3 Development of the density of W_t for a Wiener process with drift $\mu = 1$ and volatility $\sigma = 1/2$ for (top to bottom) $t = 1/4$, $t = 1/2$, $t = 3/4$ and $t = 1$. The dotted line marks the temporal procession of $\mu \cdot t \pm \sigma \cdot \sqrt{t}$

Figure 6.3 shows the densities of $W_{1/4}$, $W_{1/2}$, $W_{3/4}$ and W_1 (top to bottom) for a Wiener process with drift $\mu = 1$ and volatility $\sigma = 1/2$. The “width” of the curve (i.e., the standard deviation of the W_t) increases as a function of \sqrt{t} ; the dotted line intersects the abscissae of the coordinate systems at the points $\mu \cdot t \pm \sigma \cdot \sqrt{t}$, respectively.

When compared to the discrete process $\{S_k\}$, we have now become more flexible in the sense that we can simply assign an $\mathcal{N}(\mu \cdot \delta t, \sigma^2 \cdot \delta t)$ -distributed amount to an interval $[t, t + \delta t]$, where δt is no longer restricted to integer values. Now the question arises whether this generalization is still acceptable for any practical applications: With the discrete process we could simply compute the S_k through

k realizations of the random variables T_i and a summation. Now it appears that one has to consider the entire continuum $[0, t]$ for a realization of W_t —which probably requires a discretization and thus causes a corresponding discretization error. However, in reality, this is not the case: We actually know the distribution of W_t , *without* having to look at the intermediate values of s , $0 < s < t$.

This allows for the simulation, i.e., the computation of observed values for each discrete distance δt :

- $W_0 := 0$.
- For $i = 1, 2, 3, \dots$: compute $W_{i \cdot \delta t} := W_{(i-1) \cdot \delta t} + R$, where R is an $\mathcal{N}(\mu \delta t, \sigma^2 \delta t)$ -distributed random variable.

$$(E.g., R = \sigma \sqrt{\delta t} \tilde{R} + \mu \delta t \text{ with standard normally distributed } \tilde{R})$$

Figure 6.4 illustrates some results using this method.

Concerning the choice of the time-step δt , one should point out that the time-step has no influence on the distribution of the observed values $W_{i \cdot \delta t}$ —there is no discretization error that would force one to choose a smaller grid size than that which is desired for the representation of the results. Instead, the computed values are distributed exactly as if we had observed a real (continuous) Wiener process at times separated by the uniform distance δt . (An exception occurs if a normally distributed random variable is not available but instead is simulated via a different, suitably distributed random variable on a fixed grid such that one obtains normal distributions in the limiting case $\delta t \rightarrow 0$ —in this case, the finer grid width naturally has an influence on the observed distributions.)

Therefore, in principle it is no problem to “look into the future” arbitrarily far (as long as μ and σ can be considered constant); however, the standard deviation of the results grows in proportion to \sqrt{t} so that the picture of the distant future is rather blurry.

Wiener processes are useful models not only for truly random processes but also for those in which many independent parameters sum up to an overall movement that can subsequently be modeled as a stochastic quantity—such as stock prices which will be considered closer in the next section.

In consideration of the placement of methods within their mathematical apparatus, it is worthy of mention that the Wiener process is a standard example for a *Markov process* in continuous time and can thus serve as an additional example for the Markov chains that appear in Chap. 9.

6.4 Application: Development of Money Investments

Wiener processes can also be used to model the development of a risky money investment (e.g. of stocks); this will be considered in this section.

To this end, we first consider a risk-free (i.e., a fixed interest rate) money investment, such as a savings account, whose initial deposit is increased by a fixed amount, the interest rate P , per unit of time. Here, the time unit is typically a year;

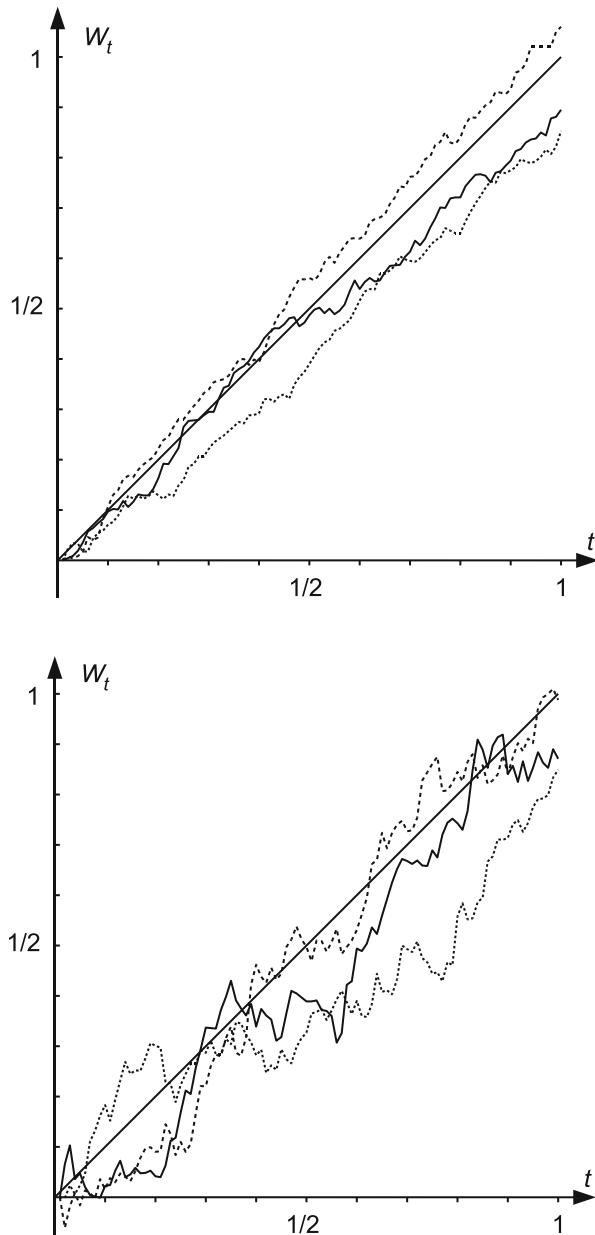


Fig. 6.4 Three respective runs of a Wiener process for $\mu = 1$ and $\sigma = 1/100$ (top) and $\sigma = 1/10$ (bottom), observations in distance $\delta t = 1/100$

in our model, which is also continuous along the time axis, the choice of time unit influences only the labeling of the axis but not the model itself; the interest rate could, for example, be $0.03 = 3\%$.

This leads to a growth process; the deposit grows per time unit by the factor $1 + P$ and, more generally, by the factor $(1 + P)^t$ for a time interval of length t (compare this to the model for population dynamics in Sect. 10.1). Since the growth is exponential, it is easier to compute using logarithmic quantities:

Let X_t present the (natural) logarithm of the value of the deposit at time t .

Let $R := X_t - X_s$ denote the *yield* that was earned during a time period $[s, t]$ of length $\delta t := t - s$. For the capital e^{X_t} , and for R not too large in modulus, this corresponds to

$$e^{X_t} = e^R \cdot e^{X_s} \approx (1 + R)e^{X_s},$$

so that the yield roughly corresponds to the interest rate for the time period δt .

Here, it is very helpful to work with logarithmic quantities: The yields are additive (i.e., if for some time period we earn a yield R_1 and then in the subsequent time period we earn a yield R_2 , the entire yield is $R_1 + R_2$). In addition, the interest rate, which up to now has been considered a constant, simply corresponds to linear growth—different yields per unit time can be read off the $t - X_t$ -diagram directly as different slopes. In contrast, it is very difficult to visually compare exponential functions of different growth rates!

Let us now turn to money investments with risk, e.g., stocks with their price fluctuations. The determination of the price depends on the influence of a large number of factors that can cause the price to go up or down. The essential point of our model is not to model these factors individually (which would lead to a much too complicated model), but to assume that the sum of all influencing factors over a certain period of time can be assumed normally distributed. Since one knows nothing, or perhaps only a little, about the distributions of the individual influencing factors, this assumption cannot be mathematically justified (e.g., by applying the central limit theorem). However, empirically it turns out to be suitable after all.

More precisely, in the so-called *Black–Scholes model*, it is assumed that the yield

$$R \sim \mathcal{N}(\mu \delta t, \sigma^2 \delta t)$$

is independent and normally distributed for non-overlapping intervals. Therefore, we have a Wiener process with drift μ and volatility σ . (The initial value X_0 is, in general, not zero, but can be set through a suitable choice of the monetary unit; however, one can also let the Wiener process begin with an arbitrary initial value.) Let the initial parameters μ and σ be given (in reality we have to estimate them).

Obviously, our model will not yield a prediction of the future stock price but only of its distribution. Nonetheless, this information is useful, e.g., for computing the underlying risks involved with the investment. A typical question is the minimum threshold amount the price will not fall below with a probability of, e.g., 99 % at

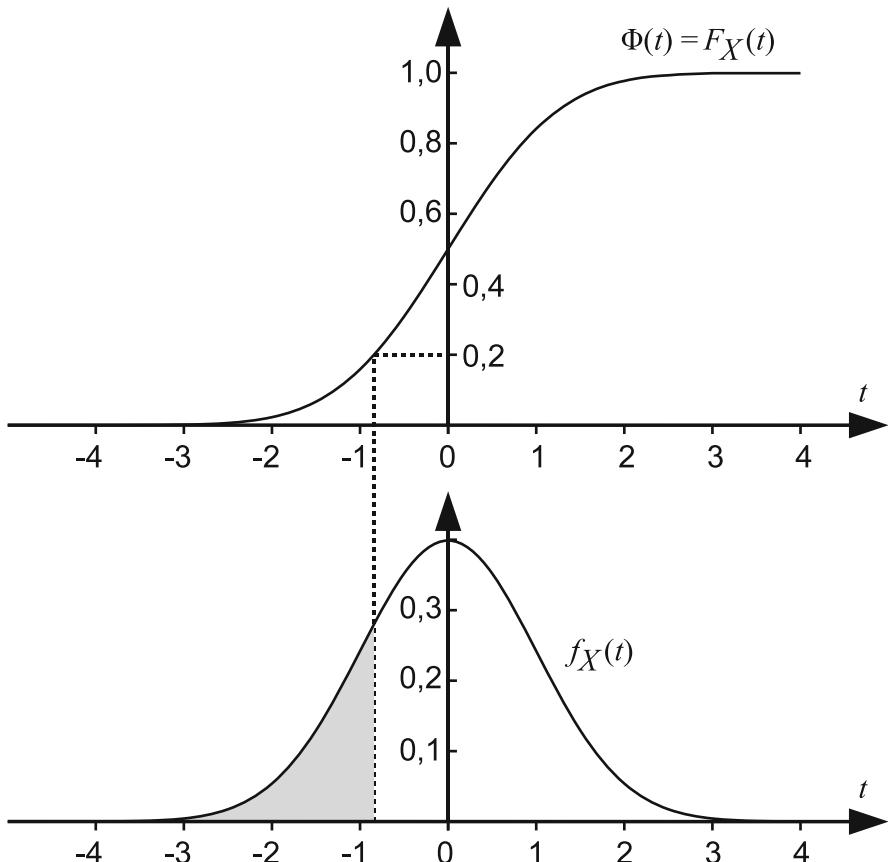


Fig. 6.5 20 %-quantile of the standard normal distribution: $z_{0.2} \approx -0.842$

time t (“This much value is still present at time t with a certainty of 99 %”—here, a high value is interpreted as a safe money investment; the smaller the value, the more speculative the investment).

For our model, this leads to the determination of a quantile of the normal distribution—in practical applications, this is mostly solved through a transformation to the standard normal distribution. The (one-sided) λ -quantile z_λ of the standard normal distribution is defined as the value which a standard normally distributed random variable $Z \sim \mathcal{N}(0, 1)$ does not exceed with a probability of λ :

$$P(Z \leq z_\lambda) = \lambda .$$

Figure 6.5 shows the graphical determination of $z_{0.2}$ using the density and distribution function given from the standard normal distribution

The values of z_λ are tabulated and available as a library function (e.g. in Maple using the function `statevalf[icdf, normald[0, 1]](lambda)` in the package `stats`, in Excel and OpenOffice using `NORMINV(lambda; 0; 1)`). One can obtain quantiles for an arbitrary normal distribution, i.e., for a $Y \sim \mathcal{N}(\mu, \sigma^2)$ by a linear transformation onto

$$Z := \frac{Y - \mu}{\sigma} \sim \mathcal{N}(0, 1) ;$$

it holds then, that

$$\lambda = P(Z \leq z_\lambda) = P(Y \leq z_\lambda \sigma + \mu) ,$$

and thus, the one-sided λ -quantile of Y can be computed as $z_\lambda \sigma + \mu$ from z_λ (when using library functions instead of tables, this conversion is mostly included).

Let us now come back to the risk assessment: The value that X_t falls below with a probability λ is the one-sided λ -quantile of an $\mathcal{N}(X_0 + \mu t, \sigma^2 t)$ distributed random variable, i.e.,

$$X_0 + \mu t + z_\lambda \sigma \sqrt{t} .$$

The corresponding amount of money is

$$e^{X_t} = e^{X_0} \cdot e^{\mu t + z_\lambda \sigma \sqrt{t}} ,$$

i.e., the initial capital e^{X_0} has constant interest yield $\approx \mu t$ and the “risk portion” $\approx z_\lambda \sigma \sqrt{t}$. Normally, λ will be (significantly) smaller than 50 %, so that z_λ is (strongly) negative, e.g. $\lambda = 1\%$ in the above example (“How much money is still there with a certainty of 99 %?”) with $z_{0.01} \approx -2.33$.

A high volatility obviously makes the money investment riskier—with our model, we can now quantify this: The “risk factor” grows (in the logarithmic quantity X_t) linearly with the volatility and with the square root of the considered time period t (while the constant yield grows linearly with t).

Two important questions remain open: How good is the model, and how do I obtain the parameters μ and σ for the money investment I am interested in?

For the assessment of the quality we look at mathematical finance in practice: The Black–Scholes model is happily used there since it is easy to work with while still yielding satisfactory results for most cases. The detractors are, on the one hand, the reasonableness in assuming a normal distribution (in practice, large amplitudes occur more frequently than the normal distribution predicts—key word “fat tail”-distributions) and, on the other hand, the assumption that μ and σ are assumed to be constant which is obviously not realistic for arbitrarily long time periods.

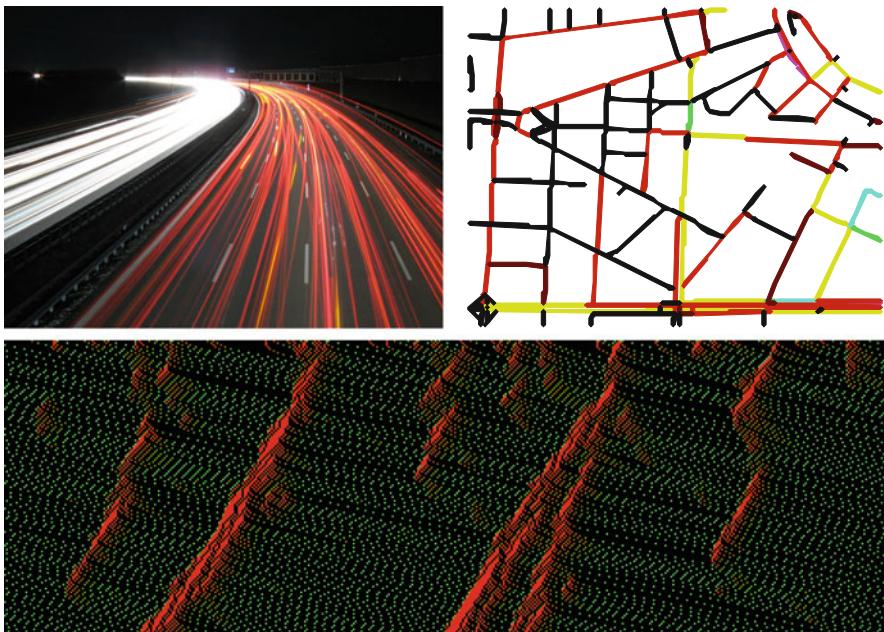
This leads to the second problem, namely the determination of the parameters μ and σ . These must be estimated from an observation of the market: One could make use of the previous development of the stock price which one can then project to

the future (for the drift this is often sufficient) or, with additional complexity (and typically necessary to include the volatility) by using the prices which are obtained from the market for certain volatility-dependent products in which one assumes that the market converts all known circumstances correctly into prices. The results of these computations can, for example, be looked up in the indices. For Germany, as an example, one could use the two indices VDAX and VDAX-NEW.

Since our excursion into the financial world has now come to an end here, we again provide an additional reference, for example, the book by Capinski [11].

Part II

Traffic on Highways and Data Highways: Once Through the Simulation Pipeline



In this part, we will consider the simulation of traffic three times. Although the underlying problem is the same, however, the perspectives taken change and we incorporate different methodical approaches.

Traffic plays an important role in many areas: examples include pedestrians, car drivers or bicycle riders in common street traffic, or with respect to data packets for data traffic in the internet or even for data packets in local networks that are in route to the printer; or the sequencing (or traffic) of orders for events in process

scheduling, telephone connections in mobile networks, repairs in a machine hall or customers at the post office.

As different as the particular applications appear, they are nonetheless linked by common requirements. Networks have to be designed that give maximum benefits for minimal costs (seemingly contradictory requirements!) while using available resources in the best possible way. Of course, bottlenecks should be avoided when planning networks or, for existing networks, be identified and then eliminated. Extensions and restructurings should also be performed to give the best possible result while suffering the smallest amount of possible side effects. If possible, one should have a positive controlling or governing impact on the traffic, as it is (hopefully) the case for auto traffic management systems.

Some common characteristics are: Simply implementing the changes is typically too time and/or cost intensive (here one may just think of the tentative construction of an interstate highway); therefore, it makes sense to make use of modeling and computer simulation beforehand. For more complicated problems, provable statements are difficult to obtain or are only attainable under a major simplification of model assumptions—a numerical simulation can hardly be avoided. Furthermore, the highway users (real or digital) oftentimes depend on more parameters than can be reasonably expected to be included in the simulation. This suggests a consideration of the traffic via average quantities or by stochastic means.

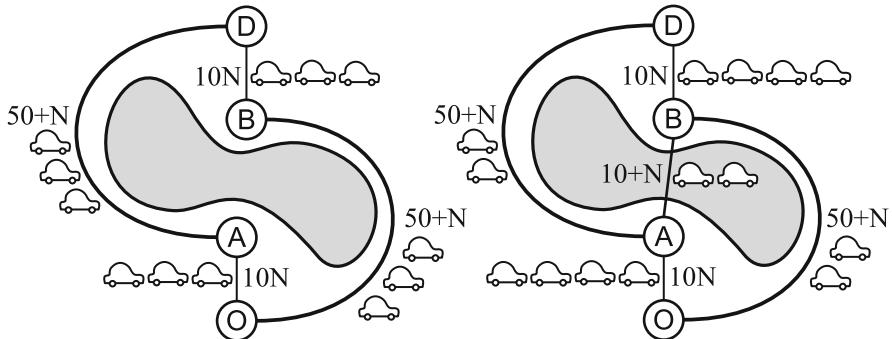
Three different scenarios for traffic with different modeling approaches have been selected exemplarily. *Macroscopically*, we consider road traffic, i.e., a continuum such as a liquid in street canals. Alternatively, road traffic resolved *microscopically* leads us to a discrete model with cellular automata. We will consider traffic in computing systems (or customers in post offices, resp.) *stochastically* and event-driven.

Here, the choice of modeling depends on the perspective taken with respect to the problem, the selection of interesting quantities, the scaling and the size of the network that is to be simulated.

The models introduced are basic and will be kept simple. We are far from grasping or covering all aspects of traffic; however, we want to use these exemplary sections to introduce and consider the path through the *simulation pipeline* from modeling up to the validation of the results gained from the analysis or simulation—and beyond this, to provide one with an additional outlook.

As a small appetizer as well as to show that traffic can be more complicated than one would at first glance assume, a little example: *Braess's paradox*. Car drivers that travel around a lake from O (origin) to D (destination) could use a route along A or B as is shown in the figure to the right.

The roads OA and BD are short, narrow, and with high danger of congestion. If N vehicles travel along one of these two routes, then they all require $10N$ min each due to mutual impediment. In contrast, the roads OB and AD are long, wide, and well developed. Here, the respective travel time is $50 + N$ min. Six cars that start simultaneously from O with destination D will sensibly split up between the two routes (left picture) in a uniform manner since each driver would like to reach the destination as soon as possible. The driving time then is calculated as $30 + 53 = 83$ min for everyone.



Next, suppose a bridge is built (with a driving time of $10 + N$ min) from A to B that is short, new, and well developed (right picture). A driver on the route via A who would like to save 2 min driving time uses the new bridge and arrives at destination D after $30 + 11 + 40 = 81$ min. However, this also increases the driving time for the three drivers via B to 93 min, and surely one of these three will want to reduce his driving time to 92 min and thus also decides to use the bridge.

As a consequence, not only do all traffic participants require 92 min, i.e., a lot longer than before, but the traffic situation is even in a stable equilibrium since (without agreement) none of the drivers can reduce the driving time by switching the route!

A modeling and simulation of the system before the decision regarding the construction of a bridge could have spared the financial burden of the developer, the environment and last but not least the drivers. We can do it better...

Chapter 7

Macroscopic Simulation of Road Traffic

Road traffic concerns all of us since everyone is affected by it. However, our wishes and ideas with respect to road traffic are typically somewhat contradictory: On the one hand, we want as much of it as possible. We want to be mobile and get as quickly and comfortably from the apartment to the university or to work, from the hometown to the vacation place or from shopping to the leisure activity. For this, we desire well-developed roads and parking places and convenient and immediately available public transportation. On the other hand, we also want as little of it as possible. Road traffic should not bother us. We do not want to be in commuter traffic nor in the kilometer long traffic jams at the beginning and ending of holidays. We wish to be without the noise and exhaust emissions and prefer green corridors over asphalt streets.

Especially the desire for a free ride is often enough more of a utopia than reality. As an example, we pick Germany, with about 81 million inhabitants one of the more densely populated countries in Europe and among the 30 most densely populated countries in the world (counting those with more than 1 million inhabitants). On the one hand, there are approximately 230,782 km of rural roads alone in Germany (only highways, interstates, state and country roads) and therefore an argument that there is a lot of room for everyone. On the other hand, however, these roads are populated by more than 51,735,000 motor vehicles licensed in Germany driving an overall mileage of about 705 billion km per year (as of 2011, [9, 50]). And this while visitors and transients from other countries have not even been considered yet. In addition, these large numbers of vehicles just happen to concentrate at central places and do not distribute uniformly across the road networks. This overload of the traffic routes leads therefore to traffic jams and ultimately to the collapse of the traffic.

Not only are these problems nerve-racking but expensive as well and should be avoided if only for financial reasons: In the EU region, the costs incurred by traffic jams are estimated to be approximately 1 % of the GDP [18], i.e., 126 billion€ in 2011. And this is not much different in other parts of the world, just consider the traffic in metropolitan areas such as New York, London, Mexico City, Paris,

Beijing, Mumbai, Los Angeles, Moscow and Tokyo. Naively building many new streets fails in view of the costs (construction and up-keep), space and time effort and is only possible up to a point. Presently, general infrastructure covers about 5 % of the German landscape, and of this infrastructure about 90 % are roads, paths and squares [49]. Above all, therefore, the available resources must be better utilized. An optimal control of the traffic via active traffic management is an example. But how does traffic work? How can we learn how to intervene and to improve the traffic situation proactively or reactively?

At first, we must understand how, for example, traffic congestion or *stop-and-go waves* are formed, and then find out what relationships hold between traffic quantities and its impact should we intervene in the road traffic at a certain place. Only then can we use this knowledge to predict traffic jams, plan new constructions, diagnose the effects of road closures and construction sites and contribute to improvements.

Hence, in this chapter we want to model road traffic. We will derive a basic model based on physical considerations and provide illustrations how refinements of an initially very simple model world can lead to a realistic image of reality. The dynamics of individual traffic participants are rather uninteresting for us. More important is the collective overall dynamic of the traffic. We thus speak of *macroscopic traffic simulation* and consider, in particular, average quantities for the *traffic density* ϱ , the *flow* f and the *velocity* v . Here, the traffic density is given in units of veh/km and describes how “closely” vehicles are positioned on the road, and the flow is given in units of veh/h and describes how many vehicle pass a check point per time unit.

If we want to have the capability to model large traffic networks such as the European highway system, we should pay particular attention that we avoid computing any unnecessary quantities. In particular, in the early days of traffic simulation, the modeling and consideration of individual participants were excluded simply from the perspective of computational effort; nowadays, road traffic can also be resolved and simulated microscopically. This will be the topic in Chap. 8. In the following, we will take a closer look at the macroscopic traffic simulation which will enable us to theoretically analyze the traffic model obtained and to obtain analytical results.

As required tools from Chap. 2, besides the basic notions of analysis (Sect. 2.2.2), in particular we need some parts from Sect. 2.4 on numerics (stability, discretization, finite differences, partial differential equations).

7.1 Model Approach

If one observes traffic from a bird’s eye view, one notices that disruptions in traffic flow expand in wave form. Here, a disruption can be anything that differs from the surroundings, for example, the end of a traffic jam or the congestion at a traffic light. Should anyone photograph road traffic at night from above using a long exposure,

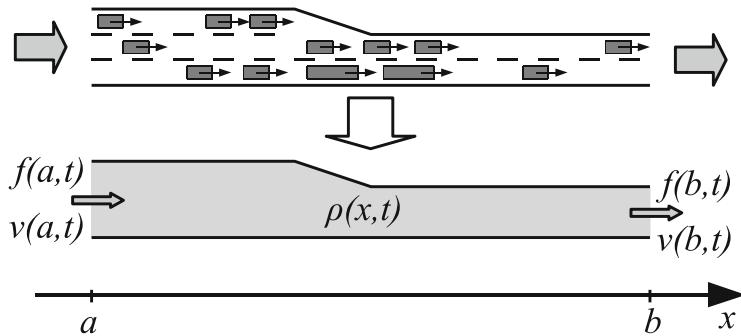


Fig. 7.1 Liquidization of vehicles

the result is the impression that vehicles flow like a viscous liquid through a canal system of streets. If a traffic light changes to green, the flow of traffic pours into the neighbouring streets as if a lock opens in a canal system.

From the perspective of the traffic, those situations are of particular interest in that the disruption dissolves or even amplifies very slowly. These are situations that involve high traffic density. Here, the expansion of disruptions (particularly on long roads) is similar to the expansion of kinematic waves in long rivers.

Based on such or similar considerations, a simple, basic *traffic model* was introduced in 1955 by M. J. Lighthill and G. B. Whitham as well as independently in 1956 by P. I. Richards, which can describe and explain the dynamic characteristics of traffic on a homogeneous, one-way road. It serves as a foundation for many improved models. Since it remains simple enough to be derived as well as simulated using basic techniques, it will be the topic of this chapter.

In addition, the macroscopic consideration of the traffic can be motivated from a microscopic perspective: If one views vehicles as individual particles in a gas or liquid, and if these, however, are not of direct interest but rather the behavior and properties of the vehicle collective, one can justify the “liquidization” of the vehicles and transition from discrete particles to continuous quantities and consider the traffic as a flowing liquid, see Fig. 7.1.

Hence, for the macroscopic modeling of traffic, for example highway traffic, the individual particles are not of interest. The model assumes that the three quantities

- *traffic*, or *flow velocity* $v(x, t)$,
- *vehicle density* $\varrho(x, t)$, and
- *traffic flow* $f(x, t)$

are sufficient for the description of the system. Above all, this assumption holds for high volume traffic when local deviations do not appear as intensely as they would on an almost empty road. Here, the following additional model assumptions hold naturally:

- $0 \leq \varrho(x, t) \leq \varrho_{\max}$ —the vehicle density is positive and limited by the maximal possible density ϱ_{\max} for which vehicles (of average length) stand bumper to bumper;
- $0 \leq v(x, t) \leq v_{\max}$ —vehicles can only move forward, and the maximal legally permitted velocity v_{\max} of the road section is not exceeded.

7.2 Homogeneous Traffic Flow

At first, we consider the most simple traffic situation, the so-called *homogeneous equilibrium flow*: All vehicles travel at the same speed and have the same spacing. There is a uniform flow of vehicles that is independent of the two variables x and t , where x represents the location x and t the time. In other words, for a section of the given route, we consider average values for velocity \bar{v} , traffic density $\bar{\varrho}$ and flow \bar{f} .

7.2.1 An Initial Result

Past measurements support the premise that the state of the considered system depends only on the density, i.e., velocity and flow are functions that exclusively depend on the density. Initially, however, we want to take a look at the general relationship between the quantities \bar{v} , $\bar{\varrho}$ and \bar{f} .

For constant velocity, a vehicle covers a distance of $\bar{v}\tau$ in time τ , see Fig. 7.2. All vehicles traveling in section $[0, \bar{v}\tau]$ have passed check point $X = \bar{v}\tau$ by the time $t = \tau$. The question addressing the number of vehicles passing this point per unit time provides us with information regarding the traffic density: In section $[0, 1]$, there are $\bar{\varrho}$ vehicles and thus $\bar{\varrho}\bar{v}\tau$ pieces in section $[0, \bar{v}\tau]$. The flow at check point X is hence $\bar{\varrho}\bar{v}\tau/\tau$. Thus, we have the *state equation*,

$$\bar{f} = \bar{\varrho}\bar{v}. \quad (7.1)$$

This state equation for traffic flow can be viewed analogously to *Little's law* (9.1), p. 216). There, it holds that

$$\text{number of customers } \bar{f}_L = \text{throughput } d_L \cdot \text{time spent in the system } \bar{y}_L.$$

Here, the *number of customers* \bar{f}_L in the system corresponds to the density $\bar{\varrho}$, the *throughput* d_L to the flow \bar{f} and the total *time spent in the system* \bar{y}_L to the reciprocal of the velocity, \bar{v}^{-1} .

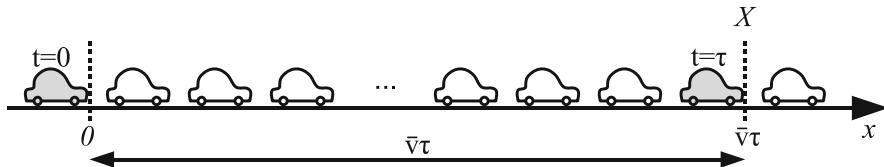


Fig. 7.2 Illustration of the state equation: All vehicles travel at the same velocity as well as the same spacing. During time τ , the gray marked vehicle moves forward by distance $\bar{v}\tau$, and all $\bar{\varrho}\bar{v}\tau$ vehicles in front of it pass the check point X

7.2.2 Velocity, Flow and Density

Next, the velocity v shall be modeled as being dependent upon the density ϱ . The study of actual traffic reveals that both quantities have an obvious causal relationship: From the point of view of the car driver, the vehicle speed should (at least for higher traffic volume) be chosen depending on the density. For the distance from the car in front, the rule of thumb, “distance of half the speedometer” is a commonly used illustration for this.

But how can $v(\varrho)$ be modeled? A look at real traffic events leads to three immediate conditions for our model:

Condition 1) If the road is (almost) empty, then drivers usually hit the gas pedal.

In the case of an empty lane, for simplification we assume for our model in the following that all drivers accelerate to the maximally allowed speed v_{\max} which is prescribed by the speed limit for the road:

$$\varrho \rightarrow 0 \Rightarrow v \rightarrow v_{\max}.$$

Condition 2) In the case of a filled road as, for example, in a traffic jam or idling before a red light, the vehicles stand bumper to bumper, i.e., with maximal density ϱ_{\max} . The traffic speed comes to a standstill:

$$\varrho \rightarrow \varrho_{\max} \Rightarrow v \rightarrow 0.$$

Condition 3) Thus, the two cornerstones are given. Since it is reasonable in view of traffic safety that a vehicle does not increase its velocity when the density increases, we can request as a third condition for the intermediate situation that the velocity decreases monotonically for increasing density.

The easiest possibility allowing the modeling of v in a way that satisfies these three conditions is through a linear representation that is given uniquely through the two cornerstones. Then, it holds that,

$$v(\varrho) := v_{\max} \left(1 - \frac{\varrho}{\varrho_{\max}} \right). \quad (7.2)$$

Next, the flow must be modeled. Once more, we use a simplified modeling assumption in that the flow f depends only on the density ϱ . From this, we obtain an equation for f if we substitute the general relationship $v(\varrho)$ into the state equation (7.1):

$$f(\varrho) = v(\varrho) \cdot \varrho. \quad (7.3)$$

Let us reflect for a moment to establish whether the model assumption is realistic. It holds that

$$f \rightarrow 0, \text{ for } \varrho \rightarrow 0 \text{ and } \varrho \rightarrow \varrho_{\max}.$$

For very small densities, the vehicles travel with maximal speed. However, vehicles pass a check point only very infrequently—the distance between them is generally very large. For very large densities, the distance between vehicles is small; however, the velocity goes towards zero. The vehicles congest and hardly move forward any more, and once more a check point is passed very infrequently by (new) vehicles. Somewhere between these two extremes the flow will be maximal. The above assumption can therefore be employed.

7.2.3 Fundamental Diagram

The graphical representation of the relationship $f(\varrho)$ is so fundamental and important for the determination of the parameters of traffic models that it is called the *fundamental diagram of traffic flow*. In the case of the linear relationship between the velocity and density we obtain

$$f(\varrho) = v_{\max} \left(1 - \frac{\varrho}{\varrho_{\max}} \right) \varrho, \quad (7.4)$$

giving a quadratic model for the relationship between flow and density, see Fig. 7.3. For the density ϱ_{opt} , the maximal flow f_{\max} is reached.

In particular, from the point of view of traffic planners, the density ϱ_{opt} is an important optimization goal since the road section is utilized the best possible way and the maximal possible number of vehicles per time interval travels on it. For all other densities the flow is reduced.

Under these model assumptions it is of interest for the traffic planner that a (given) flow $f_{lr} < f_{\max}$ can be realized for two densities (ϱ_l and ϱ_r) and thus also for two velocities (v_l and v_r). In particular, from the perspective of the road users, the better alternative is the one having the higher velocity since their destinations will be reached more quickly. Thus, in general, the state with the smaller density and higher velocity would be desirable. A look at reality, however, shows that this state does not occur exactly in this way: car drivers tend to drive fast until the optimal

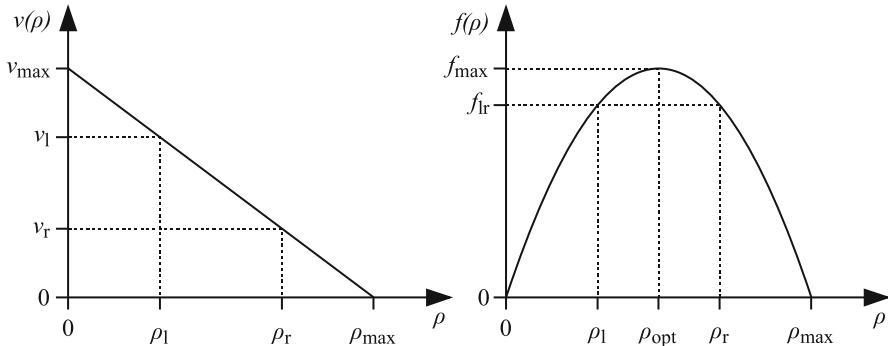


Fig. 7.3 Fundamental diagram (right) for the linear velocity–density relationship (left)

flow is exceeded ($v(\varrho)$ is just not linear), and then end up with the—also from their point of view—worse alternative. The task of the traffic planner is to intervene in a regulative manner so that the better alternative is enforced. Possibilities to this end are offered by, for example, variable speed limits signs, traffic management systems, or traffic lights.

7.2.4 Model Refinements

Through the modeling of $v(\varrho)$ it can be shown here in an exemplary way how the model can be further refined and adapted to reality.

A weakness of the previous linear model is that the maximal flow occurs at exactly half of the maximal velocity, which is unrealistic. For a given road type, one can empirically determine the density ϱ_{opt} for which the maximal flow, f_{\max} , can be measured. In order to be able to integrate this additional information into our model, we have to refine the model. To this end, we have to introduce additional degrees of freedom from which the velocity can be set for which the flow is maximal.

One possibility is to assume a quadratic relationship between velocity and density rather than the linear one:

$$v(\varrho) := v_{\max} (1 - \alpha \varrho + \beta \varrho^2). \quad (7.5)$$

For the flow–density relationship

$$f(\varrho) = v_{\max} (1 - \alpha \varrho + \beta \varrho^2) \varrho$$

this gives a cubic model.

Condition 1 is still satisfied. However, we must ensure condition 2, i.e., $v(\varrho_{\max}) = 0$ must hold. Hence, we obtain a first equation,

$$0 = v_{\max}(1 - \alpha\varrho_{\max} + \beta\varrho_{\max}^2).$$

Now we can bring into the game our knowledge on ϱ_{opt} . For this density, the flow shall be maximal, i.e., for the model it should hold that, $\frac{d}{d\varrho} f(\varrho_{\text{opt}}) = 0$. From this we obtain the second equation,

$$0 = v_{\max}(1 - 2\alpha\varrho_{\text{opt}} + 3\beta\varrho_{\text{opt}}^2),$$

allowing for the determination of our two unknowns α and β . However, it remains that we have no influence on the exact value of $f(\varrho_{\text{opt}})$.

Alternatively, we could also set up a second equation by prescribing the measured flow in the place of ϱ_{opt} . Then, with $f(\varrho_{\text{opt}}) = f_{\max}$, the second equation becomes

$$f_{\max} = v_{\max}(1 - \alpha\varrho_{\text{opt}} + \beta\varrho_{\text{opt}}^2)\varrho_{\text{opt}}.$$

However, in this case we would lose control over the fact that the flow assumes a maximum here.

Figure 7.4 shows each of these variants for a value pair $(\varrho_{\text{opt}}, f_{\max})$. In the second case, with $f(\varrho_{\text{opt}}) = f_{\max}$, we compute the desired flow at ϱ_{opt} ; however, the maximum of the flow falls on the completely wrong position and does not occur at $\varrho = \varrho_{\text{opt}}$, as desired.

While for $\frac{d}{d\varrho} f(\varrho_{\text{opt}}) = 0$ we observe the maximal flow at the right place, we violate condition 3: Negative average velocities, and therefore negative flows, are forbidden in our model. With some analysis we can figure out that we only satisfy condition 3 for $\varrho_{\text{opt}} \in [1/3 \varrho_{\max}, 2/3 \varrho_{\max}]$.

If one desires to simulate a road section and thus collects the data for a specified road type, one then obtains a similar picture as the measured data shown in Fig. 7.4. A good model should correspond as closely as possible to the observations for the fundamental quantities v and f , thus the name fundamental diagram. At least for the lower densities, the basic course of the flow can be adjusted somewhat to the behavior of the empirical observations. However, significant differences become apparent for the velocity with low density: in reality and above all else, car drivers act selfishly and want to reach their destinations as fast as possible. Because of this, they usually do not pay any attention to the effects of their own driving behavior in regard to the overall traffic: Their own velocity is reduced only if there is no other option and the density of traffic does not permit anything else.

From the fundamental diagram one can therefore distinguish between a *free flow phase* and a *congestion or traffic jam phase*. While inside the region of free flow, the traffic participants hardly obstruct each other at all. They can drive almost unhindered with the maximal allowed velocity v_{\max} , and the flow increases almost linearly. In the congestion phase, however, obstructions that are interacting with

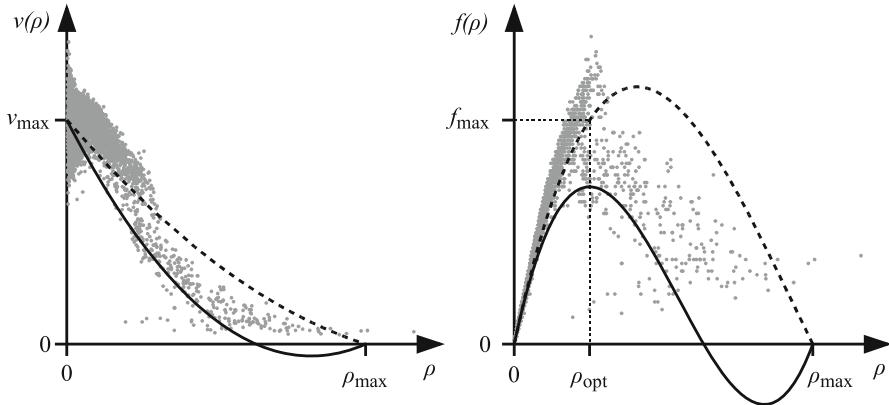


Fig. 7.4 Fundamental diagram (right) for the quadratic velocity–density relationship (left) as well as measured data (PTV AG, Karlsruhe) of a two-lane federal motorway in gray. The parameters $\frac{d}{d\varrho} f(\varrho_{\text{opt}}) = 0$ (solid), and $f(\varrho_{\text{opt}}) = f_{\text{max}}$ (dotted), resp., have been used

one another take the upper hand and the flow declines. Additionally, the velocity does not decrease proportionally to the increase in density since, in general, the traffic participants tend to overreact with their braking and thus aggravate the traffic situation.

In order to model this very basic behavior the following equation for the velocity has been suggested,

$$v(\varrho) := v_{\text{max}} \left(1 - \left(\frac{\varrho}{\varrho_{\text{max}}} \right)^{\alpha} \right)^{\beta}, \quad (7.6)$$

as another alternative [38]. In this model approach, conditions 1 and 2 are always satisfied. If we only allow positive values for the unknowns α and β , then this also applies to condition 3. The determination of the two degrees of freedom, however, becomes significantly more problematic than in the cases for the linear or quadratic models since both α as well as β appear in the exponent. For the data collected empirically of a two-lane interstate we have determined the parameters using the *least squares method* (Fig. 7.5): The starting point here is the given set M of empirical data, $\{(\varrho_i, f_i)\}_{i=1}^M$. The goal is to determine the unknowns and thus the function $f(\varrho)$ such that the sum of the squared errors on the measured points, $\sum_{i=1}^M (f(\varrho_i) - f_i)^2$, becomes minimal. Our model, however, is not linear with respect to the velocity and flow. We therefore require a numerical, iterative method such as the Gauß–Newton method which we are not prepared to further define here.

Figure 7.5 shows that the courses for $v(\varrho)$ and $f(\varrho)$ which were computed in this way are much better adapted to reality than in the previous cases. However, for

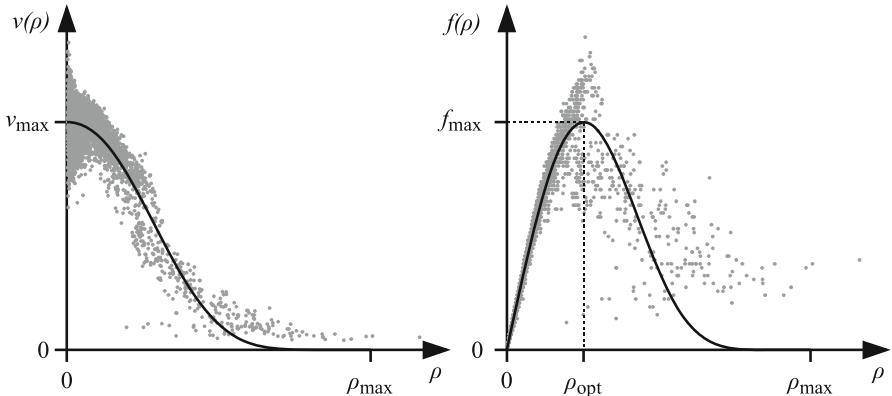


Fig. 7.5 Fundamental diagram (right) and velocity–density relationship (left) for the model (7.6). Measured data (PTV AG, Karlsruhe) of a two-lane interstate in gray

simplicity reasons, in the following we will use the linear model for the upcoming simulation and interpretation of the simulation results.

7.3 Inhomogeneous Traffic Flow

The inhomogenous situation, as, for example, it applies when considering a section of interstate, is of course significantly more interesting. In reality, homogeneous traffic flows only appear in very rare cases. For example, vehicles tend to follow a pack mentality as can be nicely observed on rural roads, and the density is subject to large fluctuations depending on the traffic situation.

The current traffic situation (f , v and ρ) can be measured by means of a variety of sensors (cameras, induction loops, ...). This provides a source for answers to the question of traffic participants with respect to whether there are currently traffic congestions or stagnant traffic. What is not captured, however, is how the traffic disturbance will develop in the near future. The later is also of interest for traffic planners who want, in addition, to know how they must interfere with the traffic in order to dissolve or avoid congestions. Both parties are interested in how the flow develops. In the following, we therefore require the flow to be interpreted as a continuous quantity that is dependent upon the time t and the position x of the road section under consideration, i.e., $f(x, t)$. Here, we assume that a continuous description of our traffic system is possible.

The foundation of the traffic model by Lighthill, Whitham and Richards is a *conservation theorem* which appears in numerous physical models and one we will encounter many more times in part IV. Here, it applies to the requirement for the

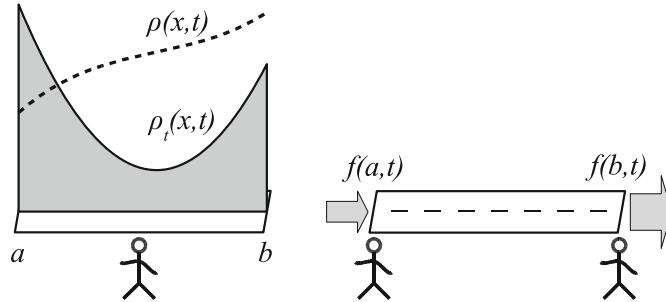


Fig. 7.6 Visualisation of the two computations of $\frac{\partial}{\partial t}n(t)$

conservation of the number of vehicles: On a road section $[a, b]$ without on- or off-ramps, no vehicles shall be lost. This leads to the so-called *continuity equation* which we will derive shortly.

At first, we consider the number of vehicles $n(t)$ at time t on the road section between $x = a$ and $x = b$. If the density is known and—as we assume—continuous in x , then we can compute $n(t)$ as

$$n(t) = \int_a^b \varrho(x, t) \, dx.$$

The change in the number of cars at time t is then,

$$\frac{\partial}{\partial t}n(t) = \int_a^b \frac{\partial}{\partial t} \varrho(x, t) \, dx. \quad (7.7)$$

For the unidirectional road section being considered, there are no on- or off-ramps between a and b . The number of vehicles $n(t)$ can hence only change if, at $x = a$, vehicles enter or, at $x = b$, vehicles leave this section. Since the flow describes the number of vehicles passing by per time at time instant t and at location x , it holds that

$$\frac{\partial}{\partial t}n(t) = f(a, t) - f(b, t) = - \int_a^b \frac{\partial}{\partial x} f(x, t) \, dx. \quad (7.8)$$

Figure 7.6 graphically illustrates the two ways to compute the time derivative.

Together with (7.7), it follows that

$$\int_a^b \frac{\partial}{\partial t} \varrho(x, t) + \frac{\partial}{\partial x} f(x, t) \, dx = 0.$$

This holds for every time instant t and every test section $[a, b]$. If we assume sufficient differentiability, we can then obtain the *continuity equation*,

$$\frac{\partial}{\partial t} \varrho(x, t) + \frac{\partial}{\partial x} f(x, t) = 0 \quad \forall x, t. \quad (7.9)$$

Let us assume in the following, that the considerations for homogenous equilibrium flows remain locally valid and, in particular, the flow thus depends on the density (7.3). This assumption does not hold in more complicated situations since it implies that cars adjust to the traffic situation without delays. Thus, situations that are caused by a change at a traffic light or from accidents cannot be simulated. The *quasisteady continuity equation*

$$\frac{\partial}{\partial t} \varrho(x, t) + \frac{\partial}{\partial x} f(\varrho(x, t)) = 0 \quad \forall x, t \quad (7.10)$$

is still sufficiently meaningful to describe the most important phenomena of traffic flows.

This *traffic equation* is a simple example of a nonlinear hyperbolic shallow water equation (transport equation) which describes general wave propagation phenomena.

7.4 Simulation of a Simple Circular Road

Let us now consider a given road section $[a, b]$ for the linear velocity model (7.4). Our goal will be to determine the traffic depending on its current state at some later time instant $t = \hat{t}$ or, in the case of a realistic traffic situation, to predict the future traffic flow. For this, we need the density $\varrho(x, \hat{t})$ of the vehicles in the road section at this time instant. If we know the density, then velocity and flow can easily be calculated.

In order to simulate this road section, we must discretize the traffic equation (7.10). For simple models such as the linear one without additional refinements, one can still find direct solutions for problem settings, for example through the analysis of *characteristics*. Here, however, we will numerically simulate in a simple way a road section and then analyse and interpret the results. If the model is extended and the traffic equation is, for example, no longer of only first order, then in general it can only be solved numerically anyways.

Once we have discretized the partial differential equation in space and time, we then need to determine suitable *initial and boundary conditions*. Thereafter, we can uniquely compute the density and thus the flow and velocity, even if only as a numerical approximation of the exact solution.

7.4.1 A First Attempt

We first discretize the path x . Instead of a real-valued position $x \in [a, b]$ we are only interested in the $n + 1$ discrete points $x_i = i \cdot h$, $i = 0, \dots, n$ which are equidistant with mesh width $h = (b - a)/n$ for the considered section. We proceed in a like manner with the time t . Starting at time instant $t = t_0$, a time step width δt leads us to the discrete time instants $t_j = t_0 + j \cdot \delta t$, $j = 0, 1, 2, \dots$

To discretize the partial derivatives we use the method of *finite differences* which have been previously introduced in the Sect. 2.4.5 for ordinary and Sect. 2.4.6 for partial differential equations. The forward difference quotients yield,

$$\frac{\partial}{\partial x} f(x, t) \doteq \frac{f(x + h, t) - f(x, t)}{h}$$

as well as,

$$\frac{\partial}{\partial t} \varrho(x, t) \doteq \frac{\varrho(x, t + \delta t) - \varrho(x, t)}{\delta t}.$$

Since—with the exception of the *initial* and *boundary values* (IV and BV)—we deal with numerical approximations as opposed to exact function values, in the following we use the notation

$$f_{i,j} := f(x_i, t_j) \quad \text{as well as} \quad \varrho_{i,j} := \varrho(x_i, t_j).$$

Given this, along with the difference quotients, we obtain from (7.9) the discretized continuity equation

$$\frac{\varrho_{i,j+1} - \varrho_{i,j}}{\delta t} + \frac{f_{i+1,j} - f_{i,j}}{h} = 0. \quad (7.11)$$

If we insert the explicit flow–density relation (the assumption for a locally predominantly homogeneous equilibrium flow) of the linear model (7.4),

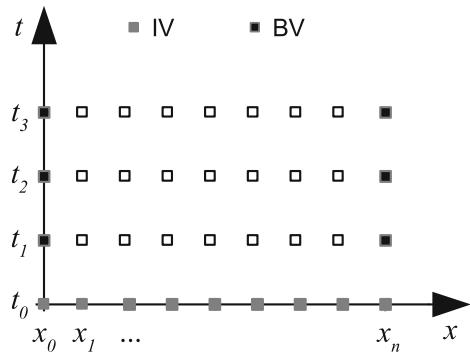
$$f_{i,j} = v_{\max} \left(1 - \frac{\varrho_{i,j}}{\varrho_{\max}} \right) \varrho_{i,j},$$

we then can compute the density at location x_i and at time t_{i+1} as

$$\varrho_{i,j+1} = \varrho_{i,j} - \frac{\delta t}{h} v_{\max} \left(\left(1 - \frac{\varrho_{i+1,j}}{\varrho_{\max}} \right) \varrho_{i+1,j} - \left(1 - \frac{\varrho_{i,j}}{\varrho_{\max}} \right) \varrho_{i,j} \right). \quad (7.12)$$

This corresponds to an explicit *Euler step* in the time direction. With the linear model, flow and velocity can easily be computed from the density. For a simulation, we need only some additional initial and boundary values (see Fig. 7.7). For initial

Fig. 7.7 Initial and boundary values for the one-dimensional road section



values, we need to know the density at the starting time t_0 of the simulation, $\varrho_{i,0} = \varrho(x_i, t_0)$, $i = 0, \dots, n$, and for the boundary values, we need the density at the end points of the road section, $\varrho_{0,j}$ and $\varrho_{n,j}$, $j \in \mathbb{N}$. In order to determine the initial values $\varrho_{i,0}$, in reality one would measure the number of cars passing by at various measuring points, for example, and therefore measure the number of cars within a road section until the next measuring point and then determine the respective density and assign it to the measuring point. Values at intermediate locations which are required for a finer resolution in the model can be determined through piecewise linear interpolation.

The neighboring road sections of the modeled street network are important for the determination of the boundary values. Road sections are defined between critical points. In the simulation of highway traffic, these would be the on- and off-ramps and motorway junctions, but also places in which the speed limit or the number of lanes change, as well as construction sites and other invasive points one finds in traffic. If two highway sections lead exclusively into each other, then the end point of one section serves as the start point of the other, and vice versa. At the on-ramps, the density has to be described at the boundary as long as the surrounding traffic network lies outside of the modeled partial section. For example, one can use characteristic diagrams for the density for this which are obtained through measurements taken on average days. It becomes even more interesting if several streets join into each other, as, for example, at an intersection in city traffic. Here, the intersection needs to be modeled separately and the densities have to be “distributed” adequately at the boundary points by means of empirical observations and thus the inlets of the street canals have to be widened or made more narrow.

In the following, we are interested in some basic observations. For this, it is sufficient to consider an individual road section. For simplicity, we therefore use *periodic boundary conditions*, simulate a circular road and set $\varrho_{0,j} = \varrho_{n,j} \forall j$.

But we still need the underlying data for the simulation. Let the length of the circular road be 10 km, the mesh width of the spatial discretization $h = 100$ m and the time step width $\delta t = 0.001$ h = 3.6 s. We consider a highway with a maximal allowed velocity v_{\max} of 120 km/h (about 75 mph). The maximal possible

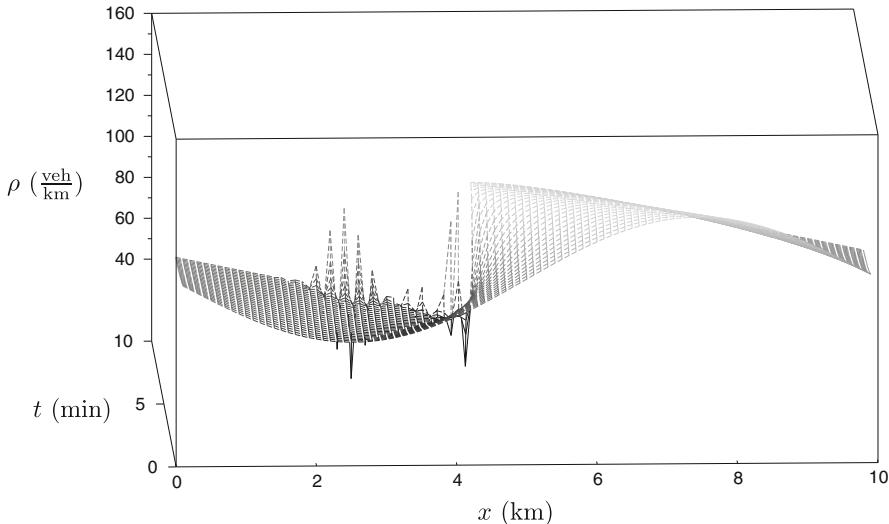


Fig. 7.8 Simulation of the density for a sinusoidal density profile (IV), $h = 0.1 \text{ km}$, $\delta t = 0.001 \text{ h}$. When using the explicit Euler method in t -direction, instabilities occur

density ρ_{\max} is set to 160 veh/km (257.5 veh/mile) in the following. This value is typically determined empirically for different types of roads and in different scenarios.

By means of equation (7.12) it is easy to see that nothing changes as time proceeds if the density at the starting time is held constant along the entire road. Phenomena such as *traffic jams out of nowhere* which dissolve again after some time cannot be explained through this simple model. The typical behavior of drivers that can cause these have not been modeled and include overreaction when braking or fluctuating in the velocity through lack of concentration or through a change in the road conditions.

In order for something to be observed, we use a simple sinusoidal density profile for the initial conditions $\rho_{i,0}$. Unfortunately, the above numerical method is not *stable*, see Fig. 7.8: Regardless of how small the time step width is chosen, the density values will explode and oscillate towards $\pm\infty$. This can also be shown very nicely by means of a constant initial density that has a small perturbation. At the place of the perturbation, the small deviation quickly builds up and the usage of symmetric difference quotients in the spatial direction does not change anything with respect to this fundamental problem.

7.4.2 An Improved Simulation

In order to avoid the numerical instability described above, an implicit Euler method could be used for the approximation of the time derivative. This, however, hinders the treatment of boundary values at on- and off-ramps as well as at the junctions.

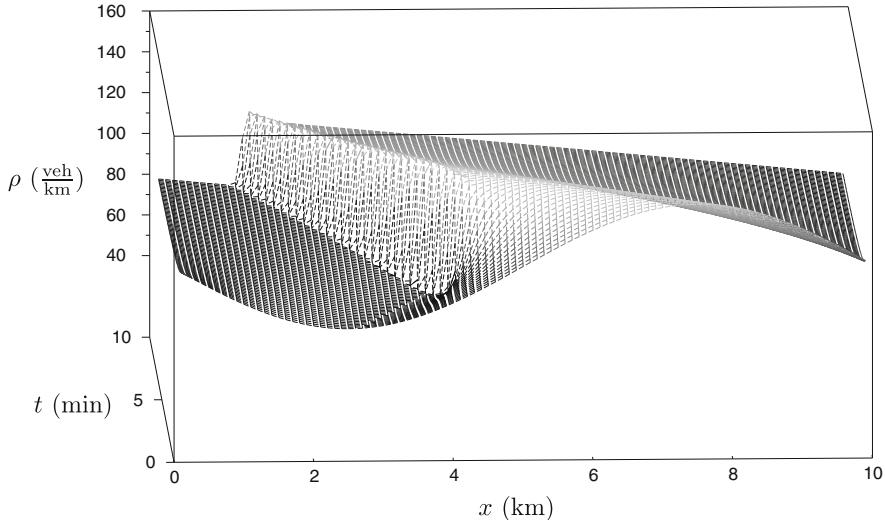


Fig. 7.9 Simulation of the density for a sinusoidal density profile (IV) with the method of MacCormack, $h = 0.1 \text{ km}$, $\delta t = 0.001 \text{ h}$. Average density 95 veh/km

Instead, we can use the *method of MacCormack* that is a *predictor-corrector method*. As the predictor step,

$$\tilde{\varrho}_{i,j+1} = \varrho_{i,j} - \delta t \frac{f_{i,j} - f_{i-1,j}}{h}, \quad (7.13)$$

which corresponds to an Euler step, a first approximation $\tilde{\varrho}$ is determined for the densities at the next time instant t_{j+1} , from which the values $\tilde{f}_{i,j+1}$ for the flow can be calculated. For the corrector step,

$$\varrho_{i,j+1} = \varrho_{i,j} - \frac{\delta t}{2} \left(\frac{f_{i,j} - f_{i-1,j}}{h} + \frac{\tilde{f}_{i+1,j+1} - \tilde{f}_{i,j+1}}{h} \right) \quad (7.14)$$

in which the difference quotient for the flow is replaced by the average taken from the backward difference quotient at time t_j (from the current values f) and the forward difference quotient at time t_{j+1} (using the new calculated approximations \tilde{f} derived from the predictor step).

This method is *stable* as long as the quotient $\frac{\delta t}{h}$ is sufficiently small, i.e., the time step has been chosen sufficiently small in relation to the spatial discretization. With this, we can simulate the previous example. Figure 7.9 shows the simulation for the first ten minutes.

Noticeable is that the initial smooth density profile changes quickly at the increasing flank. A discontinuous density jump, or shock wave, forms. This moves opposite to the driving direction. A look at highways shows that this behavior is

characteristic: By maintaining the high speed suitable for low density traffic volume, most drivers wait until they are forced to decelerate at the tail end of a traffic jam even when the traffic jam is already in sight. Thus, typically an abrupt density jump occurs at the end of a congestion. Here, an anticipatory and early deceleration would avoid many rear end collisions.

The traffic jam beginning in the model, however, differs from that which occurs in reality: Here, the traffic jam dissolves uniformly as the density decreases linearly. Whoever is traveling in the car, however, discovers that the road is suddenly empty when departing the traffic jam, the vehicle density decreases stepwise and it is now possible to accelerate considerably. The reason for this is that the ideal driver is modeled to react immediately, while an actual driver reacts only after some delay.

This behaviour was not modeled. But more complicated models do take this into account and include, for example, a term in the equation which models relaxation or reaction time.

7.5 Signal and Traffic Velocity

The direction and speed in which the “signal” end-of-traffic-congestion moves is not only of particular interest to the traffic planner. It is also of interest to the traffic participants to know when and where they are likely to find a traffic jam so that they can drive around it. Additionally, in the following we want to answer the question as to why in our model there even appears a density jump rather than, as one might possibly assume, just a shift of the increase of the flank.

In order to answer these two questions we must introduce the *signal velocity*

$$f_\varrho(\varrho) := \frac{\partial}{\partial \varrho} f(\varrho) \quad [\text{veh/h}] / [\text{veh/km}] = [\text{km/h}]. \quad (7.15)$$

This describes the effect of a density change onto the flow and indicates the speed in which the information of the change propagates with respect to the location on the street. Since a density change represents a deviation from the desired uniform traffic, one frequently refers to the speed in which a disruption propagates throughout the traffic. A look at the unit shows that $f_\varrho(\varrho)$ is in fact a velocity.

For a given density ϱ , the signal velocity can be descriptively read from the fundamental diagram, see Fig. 7.10: It corresponds to the slope of the tangent in point $(\varrho, f(\varrho))$. It is not to be mixed up with the traffic velocity $v(\varrho)$, that is the velocity at which the vehicles travel on the road. This is always positive, since our model assumes that driving backwards is forbidden (see also Fig. 7.3). Likewise, it can be read off the fundamental diagram as the slope of the secant through the point $(\varrho, f(\varrho))$ and the origin. Thanks to the convexity of the fundamental diagram, the signal velocity is always less than or equal to the traffic velocity.

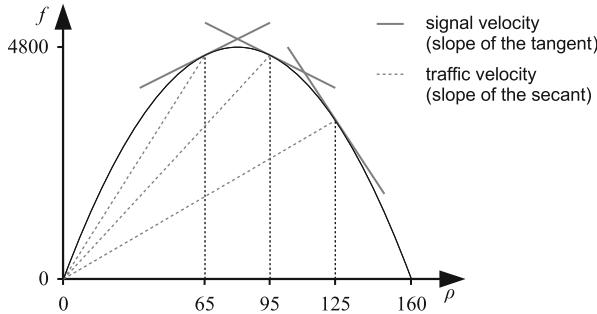


Fig. 7.10 Signal- and traffic velocities for the simulation in Fig. 7.9 for the densities 65 veh/km, 95 veh/km and 125 veh/km

Before we will take up this thought again in our simulation example, we want to take a somewhat closer look at the significance of the signal velocity: In the case of an (almost) empty road, it holds that

$$\varrho(x, t) \rightarrow 0, \quad v(\varrho) \rightarrow v_{\max}, \quad f(\varrho) \rightarrow 0, \quad f_\varrho(\varrho) \rightarrow v_{\max},$$

i.e., the signal velocity is maximal and corresponds to the traffic velocity. As an illustrative example, one can think of a vehicle that has to brake abruptly because of a deer pass. If the density is close to zero and the next driver is far enough away, he will never notice this disruption: The vehicle that slowed will already have accelerated and driven on by the time he reaches the same location. The disruption (or the information about it) disappears from the system along with the originator.

For a situation with higher density, the succeeding vehicles are hindered and therefore must also slow down, but only at a location further down the section of the road. The signal velocity in this case is still positive albeit a little smaller. The disruption or congestion moves out of the system with the corresponding signal velocity.

We can observe this exact behavior on the circular road if we start our simulation at time $t = 0$ with a sinusoidal density profile having significantly less density than in the previous case (Fig. 7.9). Figure 7.11 shows that once again a shock wave forms at the end of the traffic jam. This, however, moves with positive signal velocity and therefore forward in the driving direction.

At $\varrho(x, t) = \varrho_{\text{opt}}$, for example, for our simulation $\varrho_{\text{opt}} = 80 \text{ veh/km}$, the maximal flow has been reached. From the perspective of the traffic planner, this is the optimal traffic situation. Overall, however, it holds that,

$$\varrho(x, t) = \varrho_{\text{opt}}, \quad f(\varrho) = f_{\max}, \quad f_\varrho(\varrho) = 0.$$

Each disturbance, regardless whether it has been caused by a passing truck or the proverbial butterfly, remains persistently in the system since the signal velocity is zero and the disturbance remains at the same location. If now a vehicle has to

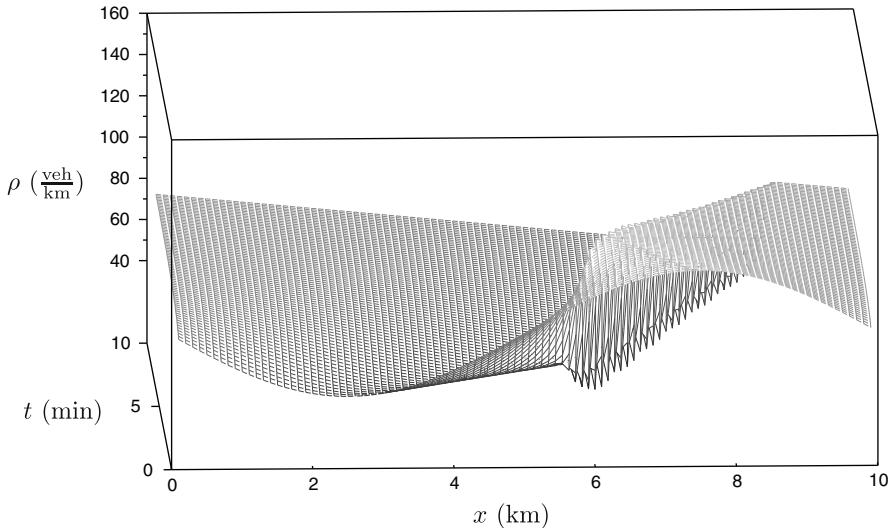


Fig. 7.11 Simulation of the density for a sinusoidal density profile (IV) using the method of MacCormack, $h = 0.1$ km, $\delta t = 0.001$ h. Average density 65 veh/km

slow down at the end of a traffic jam, this then occurs at the same place as for his predecessor since this vehicle has moved forward just enough. Without external influences the traffic jam will never dissolve. We can observe this phenomenon on the circular road as well, see Fig. 7.12. It occurs in the exact case if at the starting time the sinusoidal density profile oscillates around ϱ_{opt} . We can also observe the so-called *wiggles*, or short oscillations, which we see in the simulation at the ends of the flank, just as in reality, where a slight overreaction of the drivers when decelerating subsequently leads to a short term stop-and-go behavior.

If the traffic density increases, then for the entire road it holds that:

$$\varrho \rightarrow \varrho_{\max}, \quad v \rightarrow 0, \quad f \rightarrow 0, \quad f_\varrho \rightarrow -v_{\max}.$$

The signal velocity is highly negative while the velocity of the vehicles is very small. Now small disruptions such as a careless deceleration can lead to a total collapse of the traffic: Traffic jams form that quickly propagate to congestions over large distances. A high, negative signal velocity can be experienced on a highway in a very nice and illustrative way: If one encounters the end of a traffic jam in a high traffic density, one then can observe a “wave of braking lights” which quickly advances toward their own vehicle—until this wave reaches them, they must themselves slow down and thus the signal “end of the traffic jam” is passed on to the vehicles behind.

This is the situation given in the first simulation (Fig. 7.9). If we correspond the density information that is propagated to its associated signal velocity, then we can draw curves of constant signal velocity in the x - t plane. These so-called *characteristics* are lines. Since the density is constant along these lines, we can

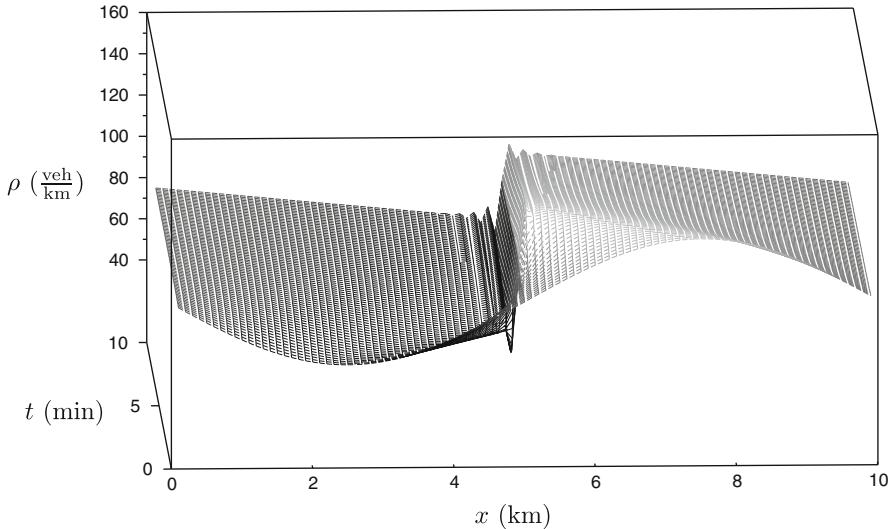


Fig. 7.12 Simulation of the density for a sinusoidal density profile (IV) using the method of MacCormack, $h = 0.1$ km, $\delta t = 0.001$ h. Average density 80 veh/km

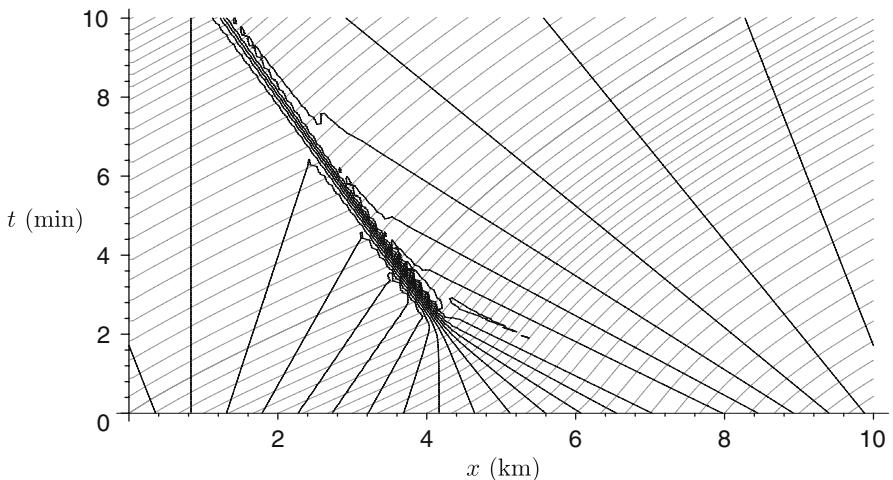
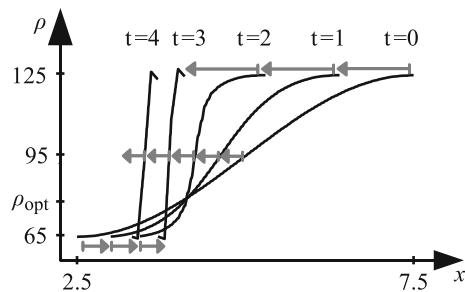


Fig. 7.13 Trajectories (curves of vehicle movements, gray) and characteristics (level curves of constant density, black) for the simulation in Fig. 7.9

draw level curves of the density projected onto the x - t -plane, starting from positions (x_i, t_0) . For the simulation this is shown in Fig. 7.13.

Thus, it becomes apparent why there is a density jump at the end of the traffic jam, but not at its beginning. At locations $x_1 = \frac{5}{6}$ and $x_2 = 4\frac{1}{6}$ it holds from the starting time that $\varrho(x_1, t) = \varrho(x_2, t) = \varrho_{\text{opt}}$. The signal velocity $f_\varrho(\varrho_{\text{opt}})$ is 0.

Fig. 7.14 Density profiles for the simulation in Fig. 7.9 at the times $t = 0, 1, 2, 3$ and 4 min. Road section between kilometer 2.5 and 7.5



In the region before the end of the traffic jam the signal velocity is positive while the region behind it is negative. This works out until the characteristics intersect each other: A density jump develops. This can therefore only occur in regions in which the density increases. If the density decreases, then the lines move apart and the transition region between high and low density becomes increasingly wider.

The density profiles of the increasing region at times $t = 0, 1, 2, 3$ and 4 min in Fig. 7.14 explain the behavior at the end of the traffic jam even more clearly: The density information at the peak value of 125 veh/km is propagated through with a highly negative signal velocity of -67.5 km/h while the one at the minimal density expands in the positive direction at only 22.5 km/h . A shock wave develops which then moves with average velocity, -19 km/h , and therefore in the negative direction. Since $f_\varrho(\varrho)$ is linear in ϱ , this occurs for the average density of 90 veh/km.

In Fig. 7.13, the trajectories of vehicle movements are given as well, i.e., the trails of some exemplarily selected vehicles. Here the difference between signal and traffic velocity becomes particularly obvious. The vehicle that starts at $x = 0$ can drive faster and faster for decreasing density (the trajectory becomes more level) until, after three minutes, it encounters the congestion front which has moved opposite to the driving direction and toward the vehicle. At this point, the trajectory has a kink and becomes steeper. The vehicle has to slow down considerably but can accelerate again little by little until it finally hits the traffic jam again thanks to the circular road.

7.6 Summary and Outlook

In this chapter, we have derived a simple, macroscopic model for the simulation of road traffic. Based on considerations for homogeneous equilibrium flow, the flow and velocity have been modeled as functions dependent on the density. Extended to more realistic, inhomogeneous traffic flow, the requirement for the conservation of the number of vehicles led to the traffic equation. This was realized with a suitable numerical method and simulated for a simple circular road for a given initial density profile.

The simulation results show that a shock wave forms at the beginning of the area of higher vehicle density which also agrees with observations from reality. This led to deliberations concerning the velocity and direction in which information is propagated in the traffic flow.

The findings from the analysis of the simulation results allow conclusions to be drawn for traffic planners: For example, disruptions that occur at the density of optimal flow can only be dissolved through external influence since here the signal velocity is close to zero. With the considerations discussed, additional traffic situations can be explained even though the current numerical simulation might lead to instabilities or the model might be too simple. An example for this is the behavior of traffic for a jump-like decrease in density as in the observed case when a traffic light changes from red to green. The obtained findings can also be transferred to other problems which also have wave characteristics. As an example here, let us mention the spread of large-scale fires.

The simple traffic model, however, is not sophisticated enough to allow for an interpretation or simulation of the many prevalent problems and phenomena occurring in road traffic. For example, the spontaneous formation of stop-and-go waves cannot be explained any better than the increase of the density wave amplitude which can be observed in reality and due to overreactions by drivers for small density fluctuations. Furthermore, inconsistent traffic states can occur: In the course of time, the model neither prevents negative densities nor densities which exceed the maximal admissible density.

However, extensions of the model which lead to Burgers equation or the vehicle sequence model by Payne and include the introduction of a diffusion component as well as additional terms exceed the scope of this book. A discussion of many additional extensions is found, for example, in [55]. By means of the modeling of traffic velocity and flow it was shown how the model can be further refined and adapted to the traffic situation that one observes in reality. In any case, as the springboard it is necessary to start with empirically captured traffic behavior in order to obtain constants as, for example, in the case for the maximal possible density for a given road type. The model can then be extended and adapted in order to correspond to the observations as closely as possible.

Chapter 8

Microscopic Simulation of Road Traffic

As in Chap. 7 we again desire to model and simulate road traffic. Naturally, once again we have as one goal the better understanding of road traffic. We want our model to explain traffic phenomena as well as possible and simulate real traffic. The successful model shall allow us to optimize requirements for traffic (to take controlling action) and to plan changes (for example building projects)—without actually having to try out all the conceivable variations that occur in reality. It is almost self-evident that these different requirements partially compete with each other. For example, a “free ride on empty roads” taken from the perspective of the traffic participant does not coincide with the goal of the traffic planner to accomodate as many vehicles per time as possible on a given section of highway without a traffic jam.

In contrast to macroscopic simulation, we are no longer interested entirely in the average value of important traffic parameters for a roadway such as the *velocity* v in km/h, the *flow* f in veh/h or the *density* ϱ in veh/km. To determine the flow, we measure the number of vehicles passing by a check point per unit time. For the density, we count the number of vehicles per leg on a control section. At the *microscopic simulation*, we want to resolve the traffic down to the individual traffic participant to “microscopic” accuracy in order to consider the individual behavior. From the point of view of the individual driver, this is important in the case for route planning, for example. This should be as dynamic as possible and be dependent on the current traffic situation. It should also account for individual properties such as the maximum speed of each driver’s vehicle.

A person traveling a lot can easily comprehend that, in part, the properties of traffic can strongly depend on the individual or on individual classes of vehicle types. Driving in the country with little view of the road ahead, a single tractor or truck can cause a long vehicle line since the other traffic participants are offered little opportunity for passing—for our macroscopic model that was considered, there always existed the implicit possibility to pass. From the perspective of the traffic planners, on a highway it is of interest to know the effect that a no-passing rule has on the overall traffic if it is applied only to a portion of the traffic participants, for

example, only for trucks. In addition, the behavior of trucks and cars on the road is significantly different. These are all reasons to consider the traffic microscopically.

If we also include the goal to precisely resolve and represent a road network (for example of a larger city), then a macroscopic simulation based on wave propagation models, as done in Chap. 7, becomes computationally very intense. In order to predict traffic congestions, it must be possible to simulate at a faster rate than real time since otherwise the results would effectively become obsolete during their computation. At least from a historical point of view the macroscopic simulation has reached its limits. As such, a simpler way of modeling had to be found.

In this chapter we want to introduce a model that is based on stochastic cellular automata. In its basic form, this model is very simple. However, it models and explains different traffic phenomena and, in improved versions, is successful in applications such as, for example, the prediction of a traffic jams in entire countries such as Germany or for the simulation of the entire individual traffic in smaller countries such as Switzerland. The required tools are elementary (except for the notion of the graph from Sect. 2.1), and no previous knowledge is necessary.

8.1 Model Approach

The first approaches making use of microscopic modeling techniques for road traffic were the so-called *car-following models*. Here, individual traffic participants are modeled via partial differential equations which has the significant advantage that for simple variants the model is analytically solvable. However, the car-following model becomes computationally too intensive when attempting to simulate a reasonable number of vehicles in a realistic scenario. Therefore, for decades traffic has mostly been simulated only macroscopically.

In the early 1990s, Kai Nagel and Michael Schreckenberg had the idea [43] to develop a traffic model (*NaSch-model*) based on *cellular automata* (CA) that allows traffic participants to be represented individually, yet is simple enough in order to allow for the simulation of larger nets as well as many traffic participants.

8.1.1 Cellular Automata

The theory of cellular automata goes back to Stanislav Ulam who, back in the 1940s in Los Alamos, analyzed the growth of crystals. John von Neumann further improved the model through his work on self-replicating systems. The basic characteristics of a CA are:

Cell space: A discrete, in most cases one- or two-dimensional cell space. All cells have the same geometry such that in the two-dimensional case mostly rectangular (cartesian), hexagonal or triangular grids arise, see Fig. 8.1.

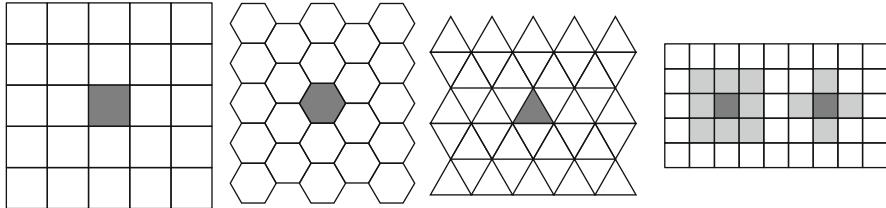


Fig. 8.1 Two-dimensional CA; Moore and von-Neumann neighborhoods

Set of states: Each cell of an automaton can assume only one (typically discrete) state from a set of states.

Neighborhood relation: Each cell can only “sense” the state of the cells in its neighborhood. In two-dimensional cartesian cell grids, typically the *Moore neighborhood* (all eight neighboring cells) or the *von-Neumann neighborhood* (only the four neighbors with joint boundary) is used (Fig. 8.1, right).

Discrete time: The state of the CA changes in discrete time steps δt . Typically the new state is computed in parallel for all cells.

Local transition function: The transition function describes how the state of a cell develops from time t to time $t + \delta t$. Since the new state depends only on the previous state of the cell and those of its neighbors, one speaks of a local transition function.

Cellular automata received publicity extensively through the *Game of Life* developed by John Conway in 1970, a simple two-dimensional cartesian CA. It is a *binary automaton*, i.e., each cell either has the state 1 (“living”) or 0 (“dead”). Whether a cell lives in the next time step depends on the current state of the cell itself as well as its eight direct neighbors (Moore neighborhood). Viewed as biological cell growth, a cell dies due to a lack of food if too many neighboring cells are occupied, for example. From a theoretical perspective, it is interesting to see that different patterns can be observed, for example, static ones or self-reproducing ones. For this, a few rules suffice for a Moore neighborhood:

1. A living cell with fewer than two, or more than three, living neighbors dies (due to solitude or lack of food, resp.).
2. A living cell with two or three living neighbors happily continues to live.
3. A dead cell with exactly three living neighbors becomes alive.

It is important that all state transitions take place in parallel and that all births and deaths occur simultaneously. From a randomly initialized “primordial soup”, patterns can arise and then become extinct, remain constant, oscillate or generate further patterns as well.

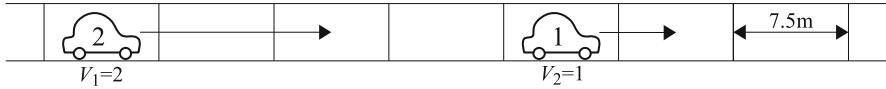


Fig. 8.2 Schematic representation of a roadway section in the NaSch model

8.1.2 Road Traffic

We now transfer the concept of cellular automata to the traffic simulation. For the modeling of road traffic we at first want to consider a simple, closed, single lane roadway. We will later extend the model to more sophisticated and realistic cases which, in principal, are developed from such simple building blocks. In addition, we allow only standard cars as traffic participants.

How does our cell space look, and what actually is a cell? We only consider a single lane which can be driven exclusively in one direction. Several vehicles cannot drive next to each other, and therefore we deal with a one-dimensional CA. In order to obtain the cell space we must decompose the roadway into discrete units.

Now, the state of a cell is not living or dead, but a cell may or may not contain a car with a certain velocity. Therefore, the cell state is either a car with a velocity value (0, if it currently stands still) or there is “no car”. With respect to the set of states, it is conceivable to allow for an arbitrary velocity for occupied cells. By doing this, however, it could then happen that a vehicle that is moving comes to a standstill somewhere between two cells. Thus, in our model we measure the velocity in cells/time step and allow only for discrete velocity levels. For the following, the maximum admissible velocity of the model will be $v_{\max} = 5 \text{ cells/timestep}$.

How big is a cell? A cell should correspond to the minimum space that a car occupies on the road, including its safety distance. In order to realistically determine this parameter we turn to the real world for help: Observations on highways have revealed that in traffic congestion situations, i.e., for maximum density, one has to assume around 7.5 m per car. This corresponds to a maximum density ϱ_{\max} of about 133.3 cars/km (214.5 veh/mile). In extreme situations and, in particular, for city traffic, the maximum density can also be higher which we neglect and exclude from our model. In summary, Fig. 8.2 shows the current state of the model.

Even before we think about neighborhood and transition functions, we can already observe a first effect of the present modeling: Since cars can only move forward in complete cells, there is a connection between the maximum velocity v_{\max} , the maximum number of cells that a car can move forward in a time step, and the time step δt . An example: Given city traffic, our modeling goal is to discretize the maximum velocity of 50 km/h (about 31 mph) in steps of 10 km/h (6.2 mph). The motion of a car with $v_{\max} = 5 \text{ cells/timestep}$ then corresponds to the velocity of 50 km/h. This implies that the length of a time step δt is determined through

$$5 \cdot 7.5 \text{ m}/\delta t = 50 \text{ km/h}$$

to be $\delta t = 2.7 \text{ s}$.

From a realistic perspective this time step is much too long—simply too much happens in the road traffic in 2.7 s. Our model is much too inaccurate and cannot represent the road traffic phenomena well enough, e.g., at intersections and junctions. As an alternative, we fix the time step and work backwards to find what this implies for the value of the maximum velocity. A realistic time step length is $\delta t = 1$ s which roughly corresponds to the average duration of one's reaction time. The velocity of the motion with one cell per time step subsequently corresponds to a velocity of 27 km/h (16.8 mph); and a maximum of five cells per time step corresponds to 135 km/h (83.9 mph) as the maximum velocity.

Is a subdivision into velocity increments of 27 km/h realistic? It is obvious that for the modeled road traffic there then only exist the velocities 0 km/h, 27 km/h and 54 km/h, which is a very coarse resolution. But overall, however, the five most important speed limits needed for the simulation of road traffic (30, 50, 80, 100, 120 km/h or, respectively, 15, 30, 50, 70, 80 mph) are roughly represented. Furthermore, the simulation already shows that in the city traffic, realistic traffic behavior can also be observed. Thus, in the following we keep the time step $\delta t = 1$ s and the maximum velocity $v_{\max} = 5$ cells/timestep, which, of course, exempts the German autobahns without any speed limits, for example.

Two central model assumptions that our model shall satisfy are the requirements for the

- *absence of collisions* and
- *conservation of vehicles*.

Absence of collisions means that two vehicles may never drive within the same cell within a time step or between two time steps. This enforces, in particular, that a vehicle must be able to decelerate without any time lag. If a car travels at $v = v_{\max} = 5$ and, at time step t , arrives in the cell directly behind a non-moving car, then it must be assured that in the next time step it reduces its velocity to $v = 0$ and also stops moving. Hence it holds for the neighborhood relationship: For this simple model it is sufficient for a vehicle to look ahead the maximum step width in the driving direction, i.e., five cells.

In view of the requirement for a conservation of vehicles, no vehicles can be lost. There cannot be any transition rule that lets vehicles disappear as, for example, in the case of the Game of Life. In particular, no cars can be lost at intersections and junctions or as the result of traffic rules. However, this will interest us at a later time. But we now turn our attention to the transition function. Let the vehicles be numbered with respect to the driving direction, i.e., vehicle i drives with velocity v_i behind vehicle $i + 1$ with velocity v_{i+1} . Let $d(i, j)$ be the distance of vehicle i to vehicle j in driving direction, i.e., the number of cells between i and j . Two cars that stand directly one after the other have distance zero. The rules of the transition function hold in parallel for all vehicles:

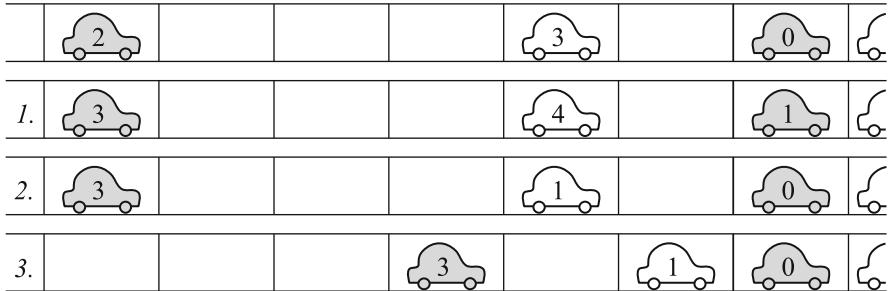


Fig. 8.3 Vehicles in the model labeled with their respective velocity as they are reaching the end of a traffic jam of a simulation step: Original state, 1. Accelerate, 2. Decelerate, 3. Move

Algorithm 8.1 (The rules of the CA model).

Update for vehicle i :

1. **Accelerate:** $v_i := \min\{v_i + 1, v_{\max}\}$
2. **Decelerate:** $v_i := d(i, i + 1)$, if $v_i > d(i, i + 1)$
3. **Move:** vehicle i moves v_i cells forward

Figure 8.3 illustrates a time step for our example. In the first step, all drivers attempt to accelerate in order to reach the maximum allowed velocity v_{\max} . Here we assume idealized drivers and vehicles in that all want to, and can, reach the allowed maximum velocity. In the next step, the absence of collisions comes to bear: The vehicle must be decelerated in the case that the preceding car is too close and thus driving at v_i is not allowed. Finally, in the third step all vehicles move. It is noteworthy that vehicles only need to know their own velocity and not that of the other vehicles. In reality, one typically tries to come up with a rough estimate of the velocity of the other traffic participants in order to modify one's own behaviour.

8.2 A First Simulation

Of interest is that this simple model can actually be simulated. We simulate a one-lane unidirectional highway section. For this model, the vehicles may drive at most with a velocity $v_{\max} = 5$, i.e., with 135 km/h. Our road has a length of 2.25 km, which corresponds to 300 cells.

As usual we need the *initial* as well as the *boundary conditions* for the simulation. On the boundary the question arises which addresses when new vehicles enter the roadway as well as what happens to vehicles that drive beyond the last cell at the end. For some simulations it is advantageous to prescribe the flow at the beginning and to let vehicles disappear when they leave the simulation domain. In the following, we use the easiest possibility and choose periodic boundary conditions: Every vehicle

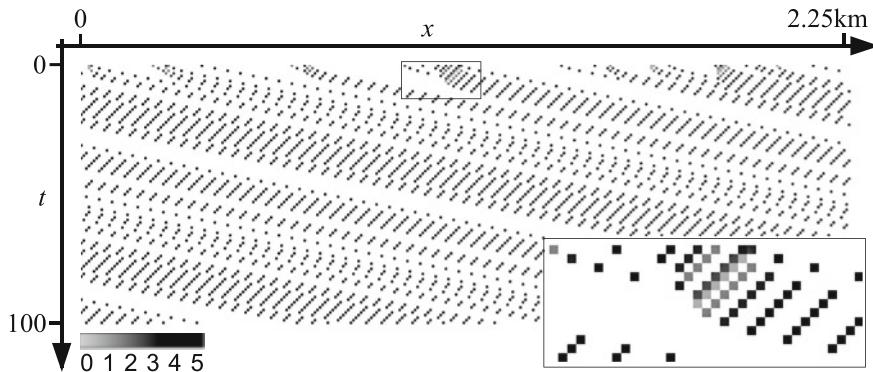


Fig. 8.4 Simulation of a circular road of length 2.25 km over the time period of 100 s, 10 % occupancy. The initial occupancy at time $t = 0$ was initialized randomly, hence traffic jams arise. These dissolve already after a few steps (see enlarged section), and a steady traffic situation is the result

that leaves the roadway at the end reappears at the beginning. In particular, this permits one to simulate a constant number of vehicles over a long time period.

As an initial condition we can randomly generate cars, i.e., we select cells randomly and initialize them with vehicles having an initial velocity between 0 and 5. In general, the initial state is thereby inconsistent and as such could never be reached in the course of a simulation. Since the velocities are adjusted before the subsequent movement of the vehicles and since it is possible to decelerate arbitrarily fast from the maximum velocity down to 0, this does not impose a problem. For improved models in which one decelerates through foresight it is advantageous to let all vehicles start with $v = 0$.

On our circular road we can prescribe an arbitrary density at the initialization. For our simulation, we choose to occupy 10 % of all cells which corresponds to a density of approximately 13.3 veh/km. Figure 8.4 shows a first simulation. The roadway is applied horizontally while time is progressing in the vertical direction. Hence, a row in the diagram corresponds to the state of the CA at time t . The shade of gray presents a scale for the velocity; a light gray scale implies that a vehicle stands still. The first row is the CA after the initialization.

In the enlarged section one sees that the initial occupancy could never be reached since a slower vehicle tailgates a vehicle ahead that is too fast. Those vehicles coming at high speed from the left all must decelerate to the model velocity $v = 0$ and stop because of a slower vehicle in front. As soon as they are no longer blocked they enter the acceleration phase and after five time steps they reach $v_{\max} = 5$.

The random initial distribution results in initial traffic jams that, however, all dissolve quickly. On average, every vehicle has 10 cells. Since only 6 cells are required to reach and maintain the maximum velocity (one for the vehicle and five empty ones ahead of it), it can reach a speed of v_{\max} after a few time steps and maintain this velocity.

Only when at least one vehicle no longer has 6 cells available will it then necessarily come to at least one traffic jam which moves forward uniformly. This is the case when more than one sixth of all cells are occupied or if the traffic density is larger than the *critical density* of 22.2 veh/km . In either case, the simulation results are rather boring since a *steady traffic situation* occurs after an initial adjustment phase. Unfortunately, this does not correspond at all to what we observe on actual streets. We therefore need an extension of the model in order to obtain more realistic traffic behavior.

8.3 Stochastic Extension: Randomization

We do not want to make any changes with respect to the basic model assumptions such as the absence of collisions and the resulting possibility of delay-free deceleration. If one observes actual road traffic, one learns that drivers often overreact when they decelerate, and on an open highway one indeed notices who drives with cruise control and keeps his speed constant and who doesn't. For this, the crucial idea in the NaSch-model was to introduce a *randomization parameter* ("dally factor") p and to introduce a further step in the transition function. This leads to an extension of the cellular automaton to a *stochastic cellular automaton* (SCA):

Algorithm 8.2 (The rules of the NaSch-model, SCA).

Update for vehicle i :

1. **Accelerate:** $v_i := \min\{v_i + 1, v_{\max}\}$
2. **Decelerate:** $v_i := d(i, i + 1)$, if $v_i > d(i, i + 1)$
3. **Randomize:** $v_i := \max\{v_i - 1, 0\}$ with probability $p < 1$
4. **Move:** vehicle i moves forward v_i cells

Novel is that after the act of decelerating, which ensures the absence of collisions, there is an additional, randomized "dallying". In particular, the randomization step incorporates into the model three fundamental phenomena of road traffic:

1. Delay when accelerating: A vehicle that does not drive with maximum velocity v_{\max} and has an open road, i.e., one that would theoretically accelerate, does not do so as soon as possible but only with some time delay to a later step. One notes that the velocity is not reduced in the acceleration phase. The acceleration just drags on a little longer. The vehicle will reach v_{\max} eventually despite dallying. For example, $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ can become $0 \rightarrow 1 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 4 \rightarrow 5$.
2. Dallying on an open road: Drivers who drive v_{\max} for a longer period of time on an open road tend to not keep their velocity constant. Again the model excludes that a vehicle suddenly decelerates completely in several steps.
3. Overreaction when decelerating: A vehicle is hindered by a slower vehicle in front and must adjust its velocity accordingly. For example, drivers tend to hit

the brakes too hard since they misjudge the distance or speed or because they drive too carefully with respect to what is deemed an optimal traffic behavior. The condition describing an underreaction that leads to rear end collisions is excluded in the model.

There are a myriad of reasons for a driver to “dally”, whether it is a chat with the passenger or the hands-free speaking system, the handling of the GPS, a gorgeous scenery, misjudgment of the traffic situation or simply carelessness. As a consequence, our model now includes, via the stochastic parameter p , the possibility to represent a set of entirely different influencing factors that include mixtures of the above three phenomena.

Surprisingly, the introduction of the randomization step is sufficient for the model to demonstrate realistic traffic behavior. Moreover, our new stochastic model is minimal, i.e., we must not omit any rule. For $p = 0$, we get back to the deterministic first model.

8.3.1 Free Traffic Flow

If we once again simulate our circular road with a randomization factor of $p = 20\%$ and use the same boundary and initial conditions, we should then observe a similar picture as produced previously even for a longer simulation time period. Only a few, very local disruptions affect the traffic flow. Shouldn’t a dally probability of 20 % affect the traffic much stronger?

At a traffic density ϱ of about 13.3 veh/km or a respective occupancy of 10 % we are (still) in the region of free traffic flow. One speaks of a *free flow phase* if the traffic density is so low that vehicles are hardly affected by other traffic participants and can drive at approximately the maximum speed. As already described in the introduction of the randomization factor, the occurrence of dallying during free flow does not lead to a drastic decrease in the speed of a vehicle but at most to it slowing to a speed of $v_{\max} - 1$. On average, each vehicle has 10 cells available which implies 9 free cells between itself and its predecessor. If the predecessor dallies, he then has another 4 simulation steps to accelerate back to the maximum speed so that the rear vehicle does not notice anything. If in the meantime the successor dallies himself, then the distance increases again anyway. Only if a vehicle dallies long enough at a speed $v_{\max} - 1$ so that the next driver is offered a chance to overreact when decelerating, does it become possible that the velocity falls below $v_{\max} - 1$ and there arises a local danger for a traffic jam.

In the case of ideal free traffic flow we can easily determine the expected average velocity of the traffic participants with respect to its dependence on v_{\max} . Since there is presumed to be no interaction between vehicles, a vehicle travels with probability p at speed $v_{\max} - 1$, otherwise at v_{\max} . The average velocity is then,

$$v_{\text{avg}} = p(v_{\max} - 1) + (1 - p)v_{\max} = v_{\max} - p,$$

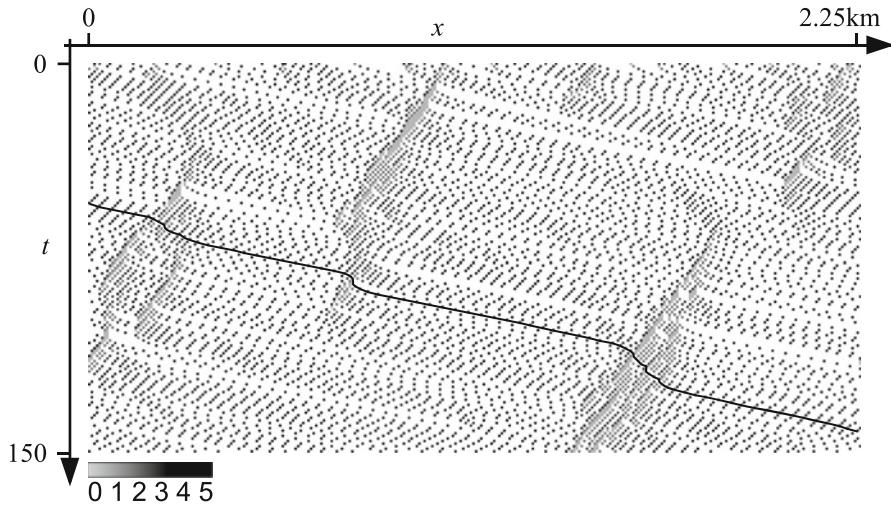


Fig. 8.5 Simulation of the circular road of length 2.25 km over 150 s, $p = 0.2$. 16 % occupancy, initialized randomly. The dallying of the traffic participants leads to traffic jams out of nowhere that dissolve again. Shown is the trajectory of a vehicle for a single pass through the circular road

i.e., for $v_{\max} = 5$ and $p = 0.2$, our vehicles travel at the converted speed $129.6 \text{ km/h} < 135 \text{ km/h}$. As with reality, traffic participants on average obtain a lower speed than legally permitted, even on entirely open roads.

8.3.2 Higher Densities, Traffic Jams out of Nowhere

In order to better assess the effects of the randomization factor, we increase the traffic density on the circular road to $\varrho = 21.3 \text{ veh/km}$. There is therefore a vehicle in 16 % of the cells. It is noteworthy to recall that this density is still noncritical, and for $p = 0$ all traffic participants after an initial phase will be on their way at v_{\max} and no traffic jams accumulate. We once again simulate our circular road. Starting at the point $x = 0 \text{ km}$, Fig. 8.5 illustrates the trajectory of a vehicle for a single pass through the circular road over 150 s.

We can observe that traffic jams form which dissolve again after a short time. Especially nice is the fact that the model can explain so-called *traffic jams out of nowhere*, traffic jams that appear spontaneously and, from an outsider perspective, without any reason. With other models these are usually not easy to observe without additional effort and must be modeled explicitly. With respect to the macroscopic modeling of road traffic which has been introduced in Chap. 7, the basic model would also have to be extended by additional terms in order to represent this phenomenon in the model.

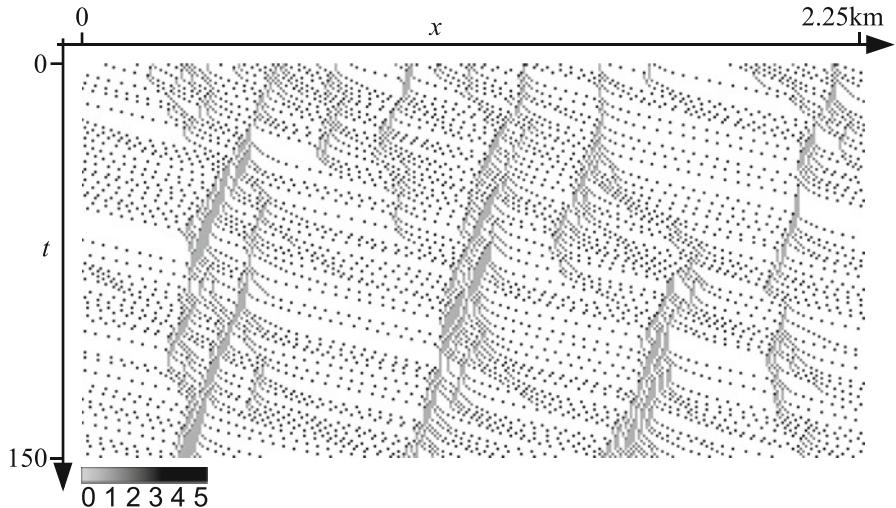


Fig. 8.6 Simulation of a circular road of length 2.25 km, $p = 0.5$; 16 % occupancy, initialized randomly. Traffic jams occur frequently due to the high randomization factor

At the bottom right in Fig. 8.5 we see how such a traffic jam can develop. One driver first dallies in a region having essentially free traffic flow reducing his speed to $v_{\max} - 1$ —and does so long enough to force the vehicle behind to drive more slowly. Due to the high density only a few time steps are sufficient for this to occur. However, this second one overreacts when decelerating, and in turn a third driver must decelerate even more abruptly. This game repeats itself until a vehicle comes to a standstill: A traffic jam is born.

What happens if we increase the probability to dally some more? Figure 8.6 shows the result from a simulation with $p = 0.5$ with the same traffic density. In comparison to the previous example one could get the impression that fewer vehicles are on the road. The reason is that the vehicles are divided into traffic jam regions with very high traffic density and regions with very low density. A consequence of this more irregular driving pattern is that traffic jams occur more frequently, therefore there are fewer vehicles in between so that it rarely comes to vehicles impeding one another. The trajectories of the vehicles are divided into very flat regions (high velocity) or very steep ones (a traffic jam).

If now we further increase the density and, e.g., use $\varrho = 33.3 \text{ veh/km}$ to occupy a quarter of all the cells but with the same randomization factor $p = 0.2$, we would then expect more and possibly even longer traffic jams. This is exactly what happens as illustrated with the simulation given in Fig. 8.7 in comparison to the one in Fig. 8.5. The traffic jams that occur are significantly more persistent and do not readily dissolve after a few minutes. The region in which the traffic comes to a complete standstill also becomes greater in distance.

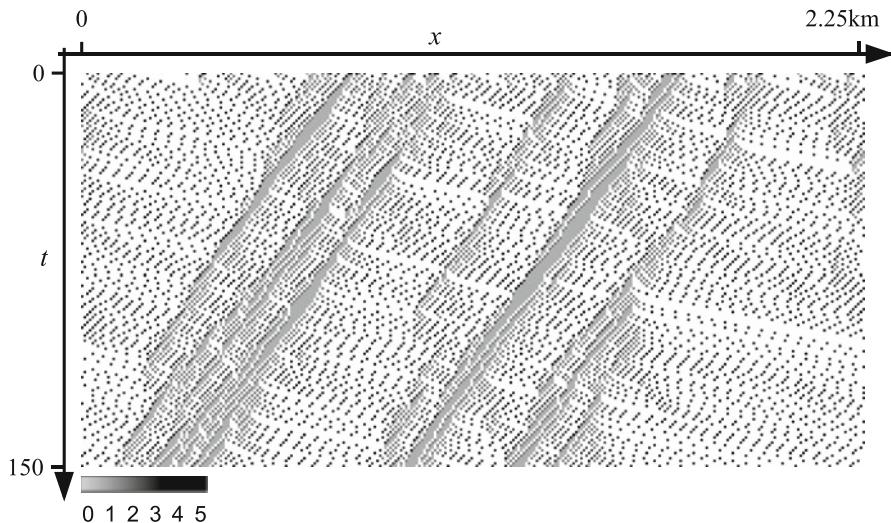


Fig. 8.7 Simulation of a circular road of length 2.25 km over 150 s, $p = 0.2$. 25 % occupancy, initialized randomly. Since the critical density is exceeded, there occur longer traffic jams which are also more persistent

If we would run the simulation with the same settings over a longer time span, we would discover that the traffic jams would persist over the next few hours and would not dissolve again, as previously observed. The two traffic jams on the right would merge into a bigger traffic jam and dominate the scene together with the left one. Meanwhile, it can happen that a traffic jam extends across almost half of the simulation region. In the traffic jam region it would indeed not come to a complete standstill (we have not sent enough vehicles onto the road); but it does develop into a *stop-and-go* behavior as it can be anticipated for the left traffic jam. Given our ability now to simulate stop-and-go waves, our model covers an additional important traffic phenomenon in connection with traffic jams.

8.3.3 Validation and Calibration: Fundamental Diagrams

Even if we can observe and explain quite a few phenomena using our model, it is still an open question regarding the accuracy of it with respect to real traffic behavior. For example, it could appear somewhat questionable that the traffic jam fronts found in the Figs. 8.5 and 8.7 spread with approximately equal *signal velocities* despite the different densities, i.e., 21.3 veh/km and 33.3 veh/km , respectively. (We refer to signal velocity when we consider the expansion velocity of a disruption in the traffic, in this case the signal “end of a traffic jam”.) Alternatively, we would also like to know what probability we have to use to mirror reality as closely as possible.

In both cases the so-called *fundamental diagram* is of use. It represents the relationship between the traffic flow f and the traffic density ϱ . The name stems from the fact that this relationship is so fundamental that it is used to calibrate model parameters as well as to develop the models themselves. For actual road traffic, fundamental diagrams have to be recorded empirically with *traffic measurements* covering different traffic situations. For this, we need paired values for ϱ and f .

In order to determine the flow f , one measures the number of vehicles N that pass a prescribed measuring station during a certain time interval δT , e.g., with the help of an induction loop in the road or a sensor at a bridge. One then determines the measured value as

$$f = \frac{N}{\delta T}.$$

Measuring the density ϱ is, however, slightly more involved: The number of vehicles N_i on a section of the road having length L must be determined at the point in time i of the measurement. One can achieve this with an aerial photograph and then by counting of the vehicles, hopefully through an automatic process. Alternatively, one tries to determine the number of vehicles entering and leaving a roadway (of length L) via sensors that are located at both its start and end. If the roadway has been empty at the beginning, the number of traffic participants in this section can be computed and thus the local density approximated.

and then computes the number of traffic participants in the measuring region. Since the flow has also been averaged over a time interval δT , one can also average the density over m measurements in the time interval δT and obtains

$$\rho = \frac{1}{m} \sum_{i=1}^m \frac{N_i}{L}$$

We can proceed exactly the same way for the simulation, see Fig. 8.8. To this end, we choose a checkpoint on the circular road and a time interval, e.g., $\delta T = 3 \text{ min} = 180$ simulation steps. For each step we consider the $k \geq v_{\max}$ cells behind the checkpoint so that we don't lose a vehicle. We then count for each measurement the vehicles in the roadway section of length $L = k \cdot 7.5 \text{ m}$.

In order to obtain measured values covering the entire density spectrum between $\varrho = 0$ and $\varrho = \varrho_{\max}$, we must systematically populate our circular road with vehicles. We begin with an empty road and add a new vehicle at regular intervals until the road is full. If we always insert the vehicles at $x = 0$, we immediately obtain traffic jams at low densities which are always caused at this location by a new car. In order to disrupt the traffic flow as little as possible through this insertion procedure, we can insert the new traffic participant at the speed v_{\max} and in the middle of the largest free interval.

Before we finally turn to our measuring results, we contemplate the largest maximum flow that we can possibly measure. For only a single lane, at most one vehicle can pass the sensor per time step. Thus we would have a maximum flow

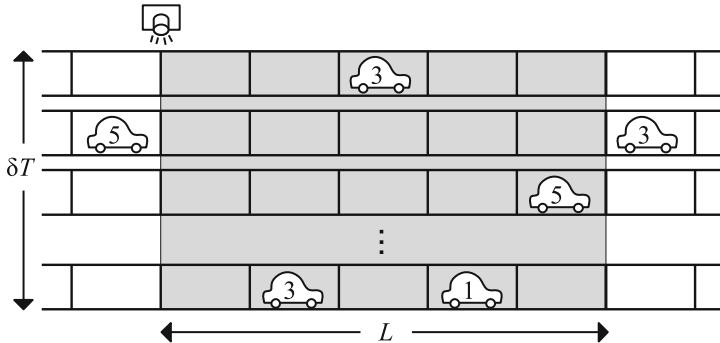


Fig. 8.8 Measurement of flow and density in the simulation

$f_{\max} = 1 \text{ veh/s} = 3600 \text{ veh/h}$. However, we will never reach it: Let us assume that all traffic participants have the same velocity, v . To allow as many as possible to pass the sensor in the measuring time span, they must drive as closely as possible to one another, i.e., with v free cells in-between. This means that every $(v+1)^{\text{st}}$ time step no vehicle passes the sensor, so we observe the flow of $f_{\max} = \frac{v}{v+1} \text{ veh/s}$. This is maximal at $v = v_{\max}$, $p = 0$ and the critical traffic density of $\varrho = 22.2 \text{ veh/km}$ and is reached in our model scenario at

$$f_{\max} = \frac{5}{6} \text{ veh/s} = 3000 \text{ veh/h}.$$

Figure 8.9 shows fundamental diagrams of measurements in our model. Different randomization factors yield different progressions.

The basic progression is correct in a qualitative sense for all fundamental diagrams with $p > 0$. This is also confirmed by a comparison with an actually measured diagram. (The fundamental diagram pictured is the same that is used for the macroscopic simulation in Chap. 7. However, it has been measured for a two-lane road, thus the flow and density have maximum values that are approximately twice as high.)

The basic progression shows in each case that there is a large increase for small densities, a sharp bend and a decrease with noise until the maximum density is reached. In contrast to the simulation, actual measurement values at very high densities are usually a rare commodity. These situations arise in reality only for a complete traffic jam and are avoided by all involved parties by all means and are therefore not that frequent. Since our virtual car drivers are significantly less prone to suffering, we can measure values even for extremely high densities in the simulation.

As can be expected, no noise is present only for $p = 0$ since all traffic participants behave optimally and deterministically. When dallying vehicles are absent, the theoretically maximum possible flow of $f_{\max} = 3000 \text{ veh/h}$ is reached

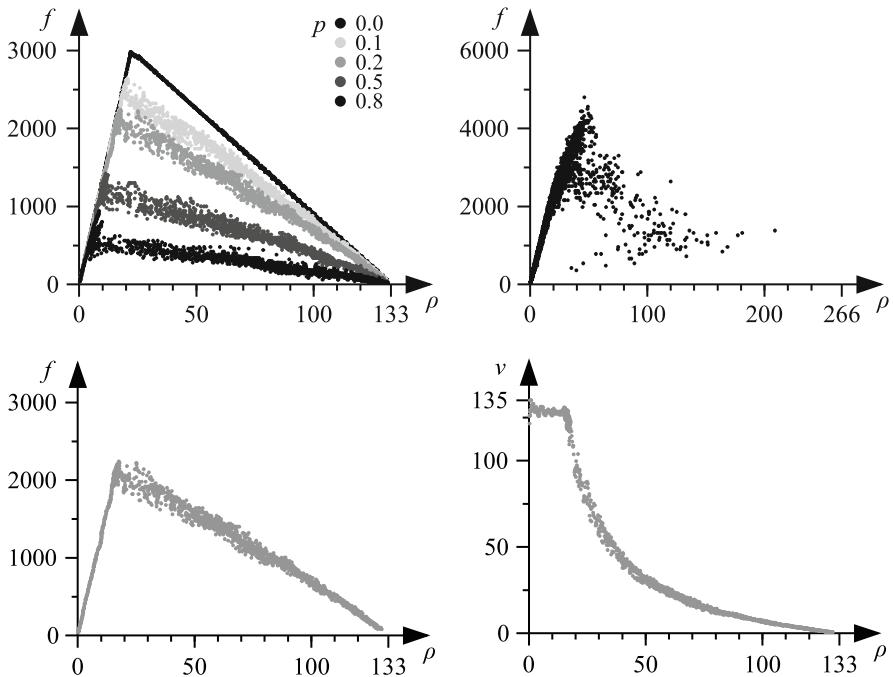


Fig. 8.9 Fundamental diagrams for different values of p (top left). For comparison, a measured fundamental diagram (top right; two-lane highway in Germany, data PTV AG, Karlsruhe). Separately, for $p = 0.2$ again, the fundamental diagram as well as the velocity–density diagram (bottom); f in veh/h , ρ in veh/km and v in km/h

at the critical density and when simultaneously all traffic participants travel at maximum speed.

During the intense increase of the flow, the traffic finds itself at small density values in the free flow phase. All vehicles drive at v_{\max} or, if they dally, at $v_{\max} - 1$ and hardly impede one another. This is clearly demonstrated in the two figures separately depicting the flow–density and velocity–density diagrams for $p = 0.2$: As long as the flow increase is approximately linear, the speed remains almost constant. Starting at a certain density, e.g., for $p = 0.2$, it is around 17 veh/km , the traffic participants increasingly impede one another and must decelerate more intensely. The traffic flow becomes *unstable* and (partially long-lasting) traffic jams form.

The larger the randomization factor p , the sooner the traffic collapses and enters into the *traffic jam phase*. The maximum flow that can be measured at the bend is an important and characteristic quantity. For actual road traffic one can assume just over 2,000 vehicles per hour per lane. We can obtain this for a randomization factor of approximately 0.2, with which we attain the most realistic traffic behavior.

Both in the free flow phase as well as in the traffic jam phase, the progression of the fundamental diagrams from our model is almost linear and only slightly curved.

With this, they differ from actual measured data which displays more pronounced curvatures. For actual road traffic, we find inhomogeneous traffic participants. Slower vehicles such as trucks lead to disruptions at an earlier time than our homogeneous vehicles that drive at the same speed during free traffic flow. This is also the reason why in the simulation the change between the free flow and the traffic jam phase is observed for uncommonly low densities. We cannot, however, readily include inhomogeneous traffic participants into our model. We would also have to allow vehicles to pass since otherwise individual slower vehicles would let everything pile up behind them.

The missing curvature also explains why the ends of the traffic jams in the Figs. 8.5 and 8.7 expand with about the same velocity despite the different densities. The signal velocity describes the effect of a change in density onto the flow, i.e., as

$$f_\varrho(\varrho) = \frac{\partial}{\partial \varrho} f(\varrho).$$

In the fundamental diagram we can read it off as the slope of the tangent. The average densities at the disruption traffic jam end are for both simulations in the traffic jam region. There, the slopes of the tangents are of approximately the same size in view of the linearity which explains our observation. Despite these strong modeling simplifications, for the sake of simplicity we will restrict ourselves in the following to single-lane traffic without the chance to pass.

8.4 Modeling of Traffic Networks

We are now ready to simulate a single-lane roadway without on- and off-ramps, preferably as a circular road. This is obviously insufficient for realistic applications. Whether the highway network in a country or the road network in an arbitrary city—it does not matter, they all are more complicated and contain parts that have many branching roads, intersections, junctions and roads of different characteristics. We also must be able to model a larger network. However, we can consider a directed lane as a building block from which we can build more complex structures.

8.4.1 Traffic Graph

If we consider intersections as important points that are connected by roadways, it then becomes intuitive to model a traffic network as a *directed graph* $G = (V, E)$ with additional edge weights:

Vertex set V : *Vertices* in the graph are important points in the traffic. Initially, these are places in which roads in particular intersect or branch off or where they

join together. However, we have to model all locations from which things can happen that can impact a simulation. This could be locations at which something changes. A few examples are places with traffic signs, in particular changes in speed limits or (pedestrian) lights, bus stops, parking lots or at least their entry points, places in which the number of lanes changes, and the beginning and ending of exit lanes. Also, consideration and attention should be given to all locations that will later permit traffic participants to begin or end their trips. The location of each node must be known and can be measured in degrees of longitude and latitude, for example.

Edge set E : The connecting segments between the vertices, i.e., the roadway sections, make up the *edges*. The weight of the edge is the length of the respective roadway. The weight of an edge can differ from the Euclidean distance between the two neighboring vertices in that curvature can be included. This suffices for some purposes; if one later wants to visualize roads with curves or curvatures again (e.g., serpentines), these then have to be decomposed into smaller roadways requiring additional vertices to be inserted. In order to represent different types of roads, besides the length of the roadway at the very least we must keep in mind the respective maximum speed. The graph is directed since we treat both driving directions separately and since for one-way streets we use only one edge in one direction.

For more refined simulations, additional parameters are of interest such as the number of lanes, no passing rules, the condition of the road surface or the slope of the road—that is, simply everything that can influence the traffic. These are mostly associated with the edges while the information that is relevant to the intersections is mostly assigned to the vertices. For example, at intersections the rules pertaining to turning or for the right of way are already indispensable even for simple simulations. Relationships pertaining to permitted vehicle turns include information about outgoing sections of a road a vehicle may take depending on the roadway on which it enters the intersection.

We typically require the graph to be *connected*. We can then reach every other vertex from any starting vertex. Comparing this to reality, we observe that this is actually the case. There are naturally some exceptions for extreme cases such as those that may be caused by avalanches in remote mountain valleys, or by local closings due to construction sites. However, for these roadways a traffic simulation is not needed anyhow.

There exist multiple ways for the modeled properties to be realized in a concrete manner as data structures. It will certainly be helpful with the decision process to look at real traffic networks that we eventually want to model and simulate: For road traffic networks, on average there are “only” three incoming and three outgoing edges per vertex. With the exception of over- or underpasses, the graph is mostly planar.

By doing this, we have approximately three times as many edges as vertices. The information describing how to get from particular vertices along particular edges to particular vertices will certainly not be stored as an *adjacency matrix* with storage

Table 8.1 Data from different traffic networks in comparison; the different degree of detail of the networks becomes apparent

Traffic network	Vertices	Roads	Districts
Karlsruhe	8.355	23.347	725
Region Stuttgart	149.652	369.009	784
Germany	109.842	758.139	6.928
New York	264.346	733.846	n/a
USA	23.947.347	58.333.334	n/a

requirement of $\mathcal{O}(|V|^2)$. Rather, it is better, for example, to store this information as an *adjacency list* requiring storage of only $\mathcal{O}(|V| + |E|)$. (An adjacency matrix is a matrix in which row i and column j has the entry 1 if the graph has an edge from vertex i to vertex j , otherwise the entry of 0 is given. An adjacency list contains for every vertex i a linked list of all neighboring vertices j reachable by an edge.)

Table 8.1 shows an overview of characteristic data of some traffic networks from simulations. For city networks, the ratio between edges and vertices is approximately 3:1 while for the Germany network it is slightly larger. In general, one can assume $\mathcal{O}(|E|) = \mathcal{O}(|V|)$ for traffic networks. When comparing the traffic network of the Stuttgart region to that of the entire network of Germany, the difference in the resolution of accuracy becomes clearly visible. While the regional simulation also requires a precise representation for smaller roads, these are less relevant for a national traffic simulation having a strong focus on highways and country roads and can therefore be the first to be neglected. If present, the number of the modeled *districts* (connected subgraphs that represent the city districts or regions, for example) is also given which could play a role for the identification of traffic demand as we will later take up in Sect. 8.4.3.

8.4.2 Intersections

We still need to decide what happens at vertices in which edges end or begin in order to simulate traffic on a road network. The easiest case occurs when a vertex contains only geographical information, e.g., if it depicts a sharp curve on a country road. Here we only have to account for relationships involving turning and ensure that a vehicle that enters from one direction cannot turn back but must continue in the other direction.

It gets more complicated for real intersections where several incoming and outgoing roads come together. For the modeling, we must in particular ensure the requirement of absence of collisions. Furthermore, as always it is an implicit goal to make the model not too complicated but rather keep it simple. This also has a positive effect on the required computational effort. In the following, we will introduce some possibilities that allow intersections to be realized in the model. These can be divided into three types of intersections: *uncontrolled intersections*, *roundabouts* and *intersections with controlled right-of-way* (traffic signs, blinking lights or traffic lights). We want to restrict ourselves to the case in which intersections have four (bidirectionally usable) roads. However, the considerations can

always be generalized to other cases (mostly three or five roads) or a different number of incoming and outgoing roads.

We do not model the road surface at the intersection itself. Vehicles will therefore “jump” across the intersection from the incoming to the outgoing road. However, there shall be no chance for a collision at the intersection itself. Vehicles that want to pass through the intersection must check if the own *plan to turn* contradicts the plan of another traffic participant. In particular, no two vehicles may turn into the same outgoing (single-lane) road within the same simulation step, even if they come to a stop on different cells. If a vehicle cannot pass through an intersection, then it must stop on the last free cell of their own driving lane.

Uncontrolled Intersections

We first give a brief remark concerning the international laws governing uncontrolled intersections (i.e., intersections not controlled by signs, signals, etc.). Vehicles may drive on the right- or left-hand side of a road. The most common right-of-way system in countries with right-hand traffic is the so-called “priority-to-the-right” system that was put forth from the United Nation’s Vienna Convention on Road Traffic.

When the priority-to-the-right is in practice, drivers must yield the right-of-way to traffic coming from the right as well as to any oncoming traffic when turning. Such priority-to-the-right laws also apply (with minor modifications) to several countries with left-hand traffic such as Australia, Ireland, and New Zealand.

Although the United States leaves the determination of traffic laws to each state, most states enforce the priority-to-the-right system. Uncontrolled intersections, however, occur less frequently in the United States—most likely because of the popularity of all-way-stops: The first vehicle arriving at the intersection governed by an all-way-stop has priority.

It is nonetheless common practice to first assume uncontrolled intersections for traffic simulations. The reason is simple—it’s more practical. The necessary data that must be collected for such a simulation is easily obtained for the road network of the region, state or country, but it is significantly more difficult (or expensive) to obtain detailed information about the specific traffic rules of the road network itself (e.g., with respect to the right-of-way, one-way streets, etc.) which are controlled through traffic lights or signs.

In the following, we assume a priority-to-the-right system for governing uncontrolled intersections, and leave it to the interested reader to transfer the observations to other right-of-way systems. In order to implement this rule correctly, a lot of different cases have to be considered and implemented for every vehicle.

If, e.g., a driver wants to make a left-turn and a vehicle across the intersection is approaching, it makes a difference whether this vehicle goes straight or also turns left. If it goes straight, the driver then must wait. For the other possibility, both may simultaneously turn left. Therefore, it does not suffice to check only for a clear roadway across the intersection. A vehicle at an intersection must also take

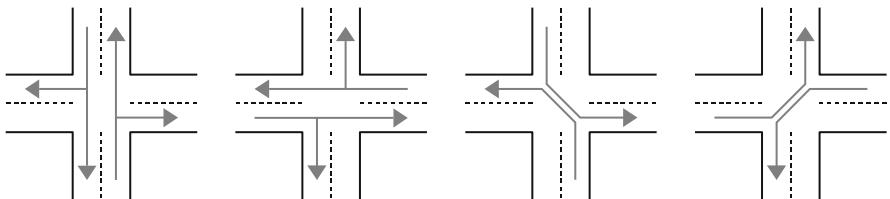


Fig. 8.10 The four-phase model for an intersection

into account all other incoming lanes. Should another traffic participant possibly drive onto the intersection, then this vehicle's turning preferences must be taken under consideration. This is new since until now we have not modeled turn signals (blinkers/direction indicators).

Furthermore, how far ahead a vehicle has to look must be decided. In theory, it would suffice to check the last v_{\max} cells of every incoming roadway. If, however, the other traffic participants travel at a much slower velocity, then gaps existing in the oncoming traffic cannot be exploited and one observes a significantly lower traffic flow than in reality. Thus, the velocities of the other vehicles should also be considered in addition to their turning preferences.

The longer the traffic participants must wait at intersections for a clear lane to turn into, the greater the danger for *gridlocks*. If vehicles from all directions want to go straight across an intersection, then the intersection is blocked. What rarely happens in reality can occur much more frequently in the model. Examples include the effects of a blockage occurring on a frequently traveled road or a significant change in the traffic volume as determined through a simulation when many drivers have to reroute themselves with a detour through a residential area. While for actual road traffic the gridlocks are typically dissolved quickly through mutual agreement, the drivers in the simulation literally starve, and a small intersection can easily lead to additional traffic jams along many other roads; a simple, random-based control is sufficient for a remedy.

A significantly simpler model is given by the *four-phase model*. While it is not as close to reality as the priority-to-the-right traffic rule, it does not require the involvement of the other traffic participants since it is sufficient for a driver to know his own turning preferences. Additionally, it avoids gridlocks. As sketched in Fig. 8.10, it consists of four turn phases that alternate after a certain number of simulation steps. Here, at first those vehicles have the right of way that go straight in one direction or turn right, next the ones in the other direction. The same subsequently holds for those wanting to turn left.

If each phase has the same number of simulation steps, it can then happen that for a realistic traffic progression nothing at all happens for several steps since the phases are only used infrequently if at all. For the first phase, if the first vehicle at the intersection wants to turn left, then it blocks the lane until the third phase. One possibility for encountering this with the model is to choose different lengths for the phases, namely a dependence on the frequency of the turning relations.

The four-phase model essentially corresponds to a traffic light set-up in which the left-turning traffic has a separate phase. It is, for example, suitable to control the right-of-way for uncritical, not too frequently used intersections such as those that appear in residential areas.

Intersections with Controlled Right-of-Way

Intersections involving a main road are slightly less problematic: The control of the right-of-way prevents gridlocks. Vehicles having the right-of-way need only, e.g., pay attention to the oncoming traffic when crossing the opposite lane; should a main road turn to the right and the right-of-way continues on this road, then all vehicles that want to turn left or continue straight are affected. Once again, since the rule scenarios are similar to the right-before-left traffic rule, most of the considerations that we made for it apply as well: The turning preferences and speeds of the other vehicles involved should be taken into consideration in order to allow for a better running and realistic traffic scenario. For high traffic densities, the zipper method can be implemented in order to avoid any situation in which vehicles yielding to traffic “starve” when driving during high traffic volume.

Similar to the four-phase model, intersections with traffic lights are controlled in phases. For the simulation one can imagine that for a red light a virtual vehicle is placed at speed 0 on the end of a roadway and for a green light it is removed. In the simplest case there is a combined traffic light phase for opposite roadways as well as the absence of a phase for a yellow light. Vehicles wanting to turn left must wait until the intersection is clear or until a vehicle in the opposite direction also wants to turn left.

In contrast to the four-phase model, a single vehicle wanting to turn left during strong oncoming traffic can once more block the access to an intersection for a long period of time thus introducing a one-sided gridlock. In real road traffic, there exists at some intersections a separate phase in which one may only turn left, or a left turn lane which must be modeled in the traffic model via an additional lane or a conditional two-lane option. These considerations, however, are beyond the scope and focus needed for our purposes here.

Nonetheless, an interesting question remains: How shall we determine the duration of a traffic light phase? If we let all traffic lights in the traffic network change simultaneously, then we typically prevent the so-called green wave and thus reduce the flow of traffic. Therefore, logic dictates that we sequence the traffic signals asynchronously. But how can this be accomplished? As well, from the perspective of any incoming road, should the time duration for red and green phases be of the same or of different lengths? Preferential consideration should also be given in the case there is a main direction that crosses several intersections. However, excessive preferential treatment can, in the extreme case, lead to a traffic jam on the connecting side roads which in turn can block the main road at another location. One possibility to optimize traffic light control when empirically determined traffic demand is available is through the use of evolutionary algorithms. When the goal is

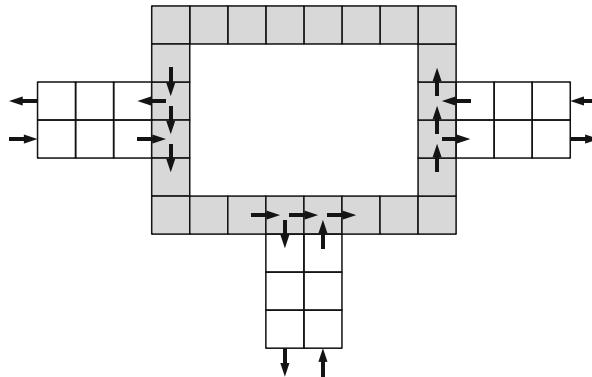


Fig. 8.11 Roundabout with three on-ramps

to optimize the flow in the main direction of traffic, one can optimize the timing of switches and the phase lengths of the lights through simulation.

Roundabout Traffic

A third possibility to control the right-of-way are roundabouts, or traffic circles. We consider the usual case, namely that vehicles that are in the roundabout have the right-of-way. Traffic lanes giving access to the roundabout are blocked for arriving vehicles only as long as vehicles in the roundabout continue to pass by. We can model the roundabout itself as a circular road with several on- and off-ramps, see Fig. 8.11.

The rules defining the entry of a vehicle into a roundabout are rather simple: We only need to check if the k fields to the left of the point of entry which can be reached at the maximum allowable vehicle speed are clear. If the next on-ramp in opposite driving direction is less than k fields away, then one possibly needs to consider vehicles approaching from that direction in order to exclude collisions. This can, however, frequently lead to complete traffic gridlocks for small roundabouts. Instead, one could, for example, reduce the speed limit for vehicles entering into the roundabout.

Also valid for roundabouts is the fact that fewer traffic jams occur if the speed and the turning intentions of the vehicles approaching from the left in the roundabout are also taken into consideration. This means that once again we must inquire or estimate the oncoming speed of the other vehicle and be able to deduce its intention with respect to turning plans (and thus to model the use of turn signals).

Reducing the size of the circular road as much as possible yields mini-roundabouts that can be used to control the traffic at intersections entirely without any external control of right-of-way. For a normal intersection with four incoming and outgoing roads we could explicitly model the intersection itself with four fields.

In order to avoid gridlocks, we could reduce the maximum speed to enter the intersection to $v = 1$ and prohibit the event for which all four intersection fields are simultaneously occupied. A restriction to 27 km/h at uncontrolled intersections could be reasonable in any case.

If we include the four different types of intersections into the model to simulate the traffic network of a city, we quickly discover that the four-phase model inadequately represents reality. This therefore is only rarely employed. In view that for real networks there always exist clear right-of-way rules to govern traffic through traffic lights or main roads at the critical or frequently used intersections, our model nonetheless can still find use in less critical residential areas. Furthermore, it becomes apparent that the traffic will quickly collapse without a sophisticated control of traffic light phases or a cleverly devised plan for the placement of main roads. It should now be clear that traffic engineers have to accomplish quite a bit here.

8.4.3 Plans and Intentions

Once we have modeled a real (or artificial) traffic network and its behavior at intersections is regulated, remaining is to determine the route taken by the vehicle once it reaches an intersection and several options are presented. We must also determine the number of vehicles that will be sent into the traffic simulation as well as the duration of their trips.

We can, of course, populate the network with a certain number of vehicles, randomly placed for the start of the simulation. For the choice of the route taken by the vehicle, the easiest option is simple chance: A vehicle that reaches an intersection hypothetically rolls the dice to determine which of the turning options to choose to continue its journey. If the probabilities given to all exiting roads are the same and if all vehicles start in the same part of the network, a diffusion process then can be observed from the bird's eye view: A uniform distribution of the vehicles across the road network is obtained. Major roads are much too empty and traffic jams form at random locations. Data regarding real traffic behavior must therefore be measured or at least estimated.

Here, empirical methodology is helpful: We measure and count vehicles in traffic. Choosing suitable probabilities at intersections allows for the smallest amount of change to our simulation. In this way, the actual traffic volume becomes the determining factor of the route selected by the initially randomly placed vehicles. From this, we obtain our first realistic results and can observe that traffic jams occur first on the stretches that are traveled the most in reality.

The next higher degree of realism is reached when we allow some degree of individualism on the part of the traffic participants and if drivers can pursue different plans. But first a small excursion: Since we lift the constraint for the uniform treatment of all vehicles, it is sometimes referred to as a *multi agent simulation*. Each vehicle is now viewed as an autonomously acting agent that interacts with the other agents and has the ability to affect its environment and to

react to it. The higher the level of additional independence (individual maximum speed, different randomization factors, vehicle sizes and behavior patterns, . . .), the more applicable this method of approach becomes. For simplicity, our vehicles will continue to share most of the defining characteristics.

With traffic polls and counts the routes traveled can be determined in terms of the number of traffic participants. The goal is to obtain representative data and, for example, to determine the traffic volume for a typical working day. Here, a route is a start–end pair of vertices for our network. These are then used to build so-called *origin-destination matrices* (OD matrices). These matrices include for every start–end vertex pair an entry value depicting the number of times the route has been used from the start point to the end point during the considered time span.

For each vehicle, one can assign an individual plan that follows a computed route. The route planning can follow one of the typical criteria (fastest, shortest, . . . route). During the simulation we can insert cars at the starting points in a way to ensure that the frequencies over the entire simulation time coincide with those from the OD matrix, e.g., at random times or at uniform intervals. When a traffic participant reaches his destination, he is removed from the simulation.

For simplicity the start and end points are often only loosely stated in the traffic polls, for example, with respect to city districts or important landmarks. A traffic participant can then be introduced into the simulation at a random place within the respective district or nearby the particular landmark.

Route Planning

The *selection of the route* can occur statically or ad hoc. To statically compute a route before departure and only update the route for unforeseen events such as big traffic jams or road blocks is a typical procedure that is simpler than making a new ad-hoc decision at every intersection. The goal for most traffic participants is to use the fastest route or, increasingly the case, the shortest route in order to minimize gas costs. Algorithms used to calculate the route are easily extended to include additional criteria, typically concerns such as the avoidance of toll roads.

Most likely the best known algorithm to search for a shortest path from a *start vertex* s to an *end vertex* z using edge-weighted graphs is the *Dijkstra* algorithm (after Edsger Dijkstra, 1930–2002). The prerequisites requiring a connected graph with nonnegative edge weights are satisfied for the traffic graph (lengths of roads should never be negative!). The algorithm is based on the idea that starting at vertex s , proceed next to the vertex who has the smallest total weight and continue repeatedly until we reach the end vertex z . The weight $g(a)$ of a vertex a is, for example, the length of the shortest known path from s to a in the traffic graph.

Somewhat more formally, we define three vertex sets: The set of vertices B that have already been visited, the set of boundary vertices R which contains all vertices that can be reached from the visited vertices within one step (via an edge), and the remaining set of the unknown vertices U (all others). For every vertex a considered,

we store in $\text{pred}(a)$ the information describing the vertex that was used to reach a and the total length of the shortest known route from the beginning vertex, i.e., $g(a)$.

Algorithm 8.3 (Dijkstra).

To initialize, we set $B = \emptyset$, $R = \{s\}$, $U = V \setminus R$ and $g(s) = 0$. Logically, the start vertex has no predecessor. Repeat:

1. Remove vertex a (active vertex) from set R having smallest total weight. If it is the end vertex z , then we are done. Otherwise, add it to set B .
2. Consider every neighboring vertex n that can be reached directly from a . Its total weight via a is $d := g(a) + \text{edgeweight}(a, n)$.
 - a) If it is in R , then check whether d is smaller than the previously known weight. If yes, then set $g(n) := d$, $\text{pred}(n) := a$.
 - b) If it is in U , then remove it from set U and add it to the set R . Set $g(n) := d$ and $\text{pred}(n) := a$.

The above algorithm is repeated until the end vertex z has been reached (or abort with an error if R is empty). The length of the shortest path from s to z is then $g(z)$. This shortest path is obtained if we backtrack from z back to s using the predecessor function.

The Dijkstra algorithm is guaranteed to find the shortest path (or a shortest path if there exist several). It is convenient that the method does not depend on what the edge weights actually represent (as long as they are positive). This means it can therefore be used for route planning with criteria such as the fastest path. With respect to a search for the fastest path, the edge weights are given as the length of the road segment divided by the maximum legal speed or estimated speed on this segment; for other criteria such as the most fuel efficient path, additional factors can be included, e.g., whether the speed can be kept constant (few traffic lights and intersections) or whether many elevation grades have to be overcome. Should a roadway be avoided (accident, construction, traffic jam, toll road, ...), it is then sufficient to set the edge weight to ∞ ; the edge does not have to be removed from the graph.

For large networks, of natural interest is whether the algorithm has practical applicability, i.e., in particular the question regarding its runtime complexity: For each step we must search the boundary set to find the vertex with the smallest weight; for a complete graph (every vertex is connected to every other one), all vertices excluding the start vertex are already in the boundary set at the initial step. In the worst case, the end vertex is the last one visited. Further, we must consider every edge at most one time. Thus the complexity is $\mathcal{O}(|V|^2 + |E|)$. If we use a suitable, efficient priority queue for the administration of the boundary vertices from which the vertex of minimum weight can be removed with little effort, then the effort can be reduced from $\mathcal{O}(|V|)$ when using a linked list down to $\mathcal{O}(\log |V|)$ when using a Fibonacci heap. The resulting runtime complexity then becomes $\mathcal{O}(|V| \cdot \log |V| + |E|)$ and for traffic networks, $\mathcal{O}(|V| \cdot \log |V|)$. Compare

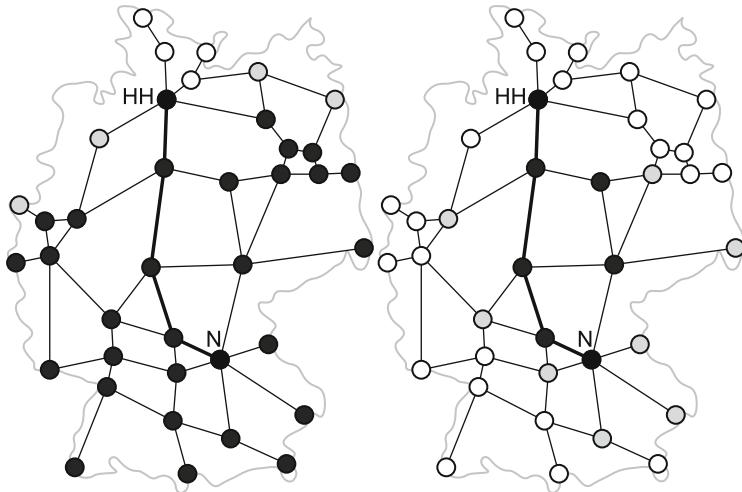


Fig. 8.12 Search for the shortest path with Dijkstra (left) and A* (right) from Nuremberg to Hamburg. Boundary vertices *light gray*, visited vertices *dark gray*. Excerpt from the network of federal highways in Germany. The advantage of the informed search with A* is apparent

this with Sect. 8.4.1. For further information on Fibonacci heaps or other priority queues we refer the interested reader to the textbook [13].

Figure 8.12 shows an excerpt pertaining to the federal highways in Germany. The left diagram gives the shortest path from Nuremberg to Hamburg determined using Dijkstra's algorithm. Start and end vertices are black with the visited vertices shaded dark gray, the boundary vertices shaded light gray and the unknowns white. We can clearly see a disadvantage: The Dijksta search is *uninformed*—i.e., we search uniformly in every direction which in this case included the southern search direction that is opposite to the northern target-oriented direction.

We speak of an *informed* search if we can provide and use a *heuristic* $h(a)$ for the remaining costs (distance, duration, ...) from the currently considered vertex a to the end vertex. By including a heuristic, we can modify Dijksta's algorithm into the A*-algorithm. Under the guidance of the heuristic this always analyzes first those vertices that will presumably lead to the goal. To this end, we must not only use the pure costs $g(a)$ but in addition the estimated remaining costs, i.e., $g(a) + h(a)$, during the first step of Dijksta's algorithm in order to detect the “smallest” boundary vertex a .

If the used heuristic h is *admissible* and *monotonic*, i.e., if h never overestimates the costs towards the target and if it holds for every successor a' of a that $h(a) \leq \text{edgeweight}(a, a') + h(a')$, then it holds for the A*-algorithm: It is optimal (an optimal solution is always found), and it is optimally efficient: There exists no other algorithm that finds a solution faster for the given heuristic. The obvious heuristic to use when searching for the shortest path, the distance between the vertex and the target as the crow flies, is admissible and monotonic. Figure 8.12 shows the

advantage from using A* in contrast to Dijkstra: It was sufficient to consider a significantly smaller number of vertices.

The A*-algorithm has the same time complexity as the Dijkstra algorithm; however, in general it finds a solution much faster. Yet, the use of the A*-algorithm also reaches its computational limits for very large networks, especially due to the storage requirements which here we do not want to dwell on further. Alternatives and significant improvements depending on the application are available through bidirectional search methods that work simultaneously from the start and end vertex and “meet in the middle”. Also available are hierarchical methods which no longer are guaranteed to find the optimal path but are very useful and applicable in practice in particular in large traffic networks. For large traffic networks on a national level, a shortest path can be computed between two cities exclusively using out-of-city roads, and then be suitably connected with a computed path from the city district to the city boundary and a path from the specified address out of the city district. This requires a subdivision of the traffic network into districts, see also Table 8.1 in Sect. 8.4.1.

Rush-Hour and Traffic Load Curves

If on the radio or in the print media the traffic report warns against “gigantic traffic jams” one thing is clear: The holidays have started or come to an end and travel or return trip traffic floods the highways. But as well, extended weekends have their malice. To a certain degree one has to accept as a traffic participant the exceptional situations such as these. After all, one cannot expand highway capacity by putting in twice as many lanes on all highways, especially if otherwise the capacities are sufficient for most times of the year. It is interesting to simulate the traffic especially for such extreme situations in order to intervene in a controlling and regulatory manner. For our model, we could simply introduce many more vehicles onto the roads simultaneously. This, however, is not sufficient for the realistic simulation of a day with holiday traffic: Those who depart very early in the morning usually reach their destinations before the traffic collapses.

The traffic in cities and areas of high population density shows even more clearly that our current methods are insufficient to generate traffic demands for a simulation. Commuter traffic clogs the streets daily. The traffic volume depends strongly on the time of the day and the day of the week.

In order to be able to integrate these phenomena into the model, we rely once again on empirical measurements. The data collected is used to produce a so-called *traffic load curve* for different scenarios, see Fig. 8.13. For every full hour and with respect given to the time of the day the traffic load curve visually shows us the relative quantity of traffic participants that begin their trip in the traffic network. The core working times of most people that commute to work daily are clearly seen in the time periods: The peak times for commuter traffic stand out. Of interest is that the afternoon rush hour traffic involves more traffic participants on the road when

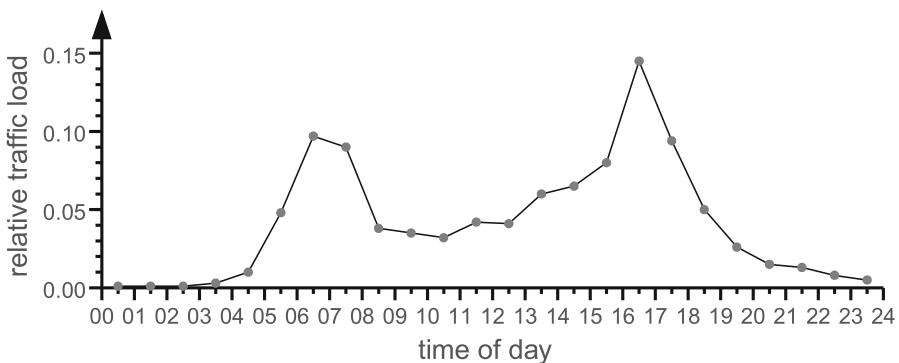


Fig. 8.13 Traffic load curve of the city traffic for a medium sized city, working day and not a holiday

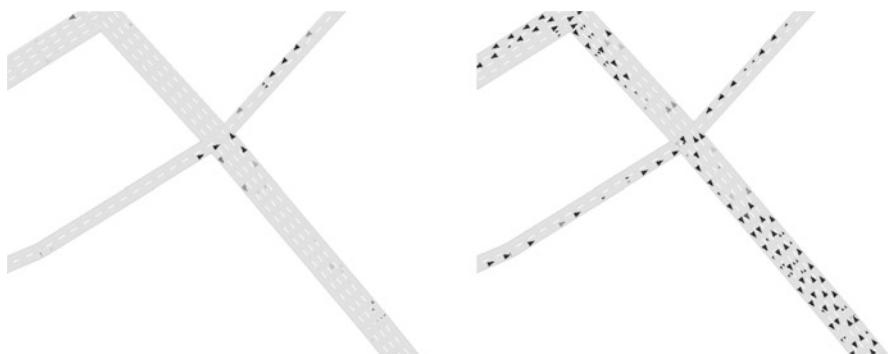


Fig. 8.14 Intersection in Karlsruhe early in the morning and at rush hour. Multiple lane road and inhomogeneous traffic participants, represented as triangles with direction, size and velocity (gray scale)

compared to the morning rush hour, no least because in the afternoon there is more going on in general on the roads than early in the morning.

For the simulation, we can use the traffic load curve to introduce vehicles for a prescribed start-end pair in the OD matrix that is dependent on the traffic demand. The simulation then permits the observation of the characteristic traffic volume and the typical rush hour traffic jam over a day, compare Fig. 8.14. We can now simulate a road network in which many important traffic phenomena can be observed. As important, we can now make traffic jam predictions and determine the effects of changes in the design and lay-out of roads or of a road block or optimize traffic regulations.

8.5 Model Refinements

In this section we shall present as examples some possibilities that would further improve our model. Some of these are very important for the adaptation to realistic traffic and include, for example, the introduction of different traffic participants as well as the admission of passing. This was already established during the discussion on fundamental diagrams, however, others are desirable but would probably cost more in computing time than they would add as a benefit to the model. Examples include a finer resolution of the network or the ability to model brake lights. Ultimately, their relative importance depends on the actual purpose for the simulation.

To this point, we have only considered homogeneous traffic participants all of whom have and desire to reach the same maximum speed v_{\max} . *Inhomogeneous traffic participants* are essential for the course of traffic on real roads. We can distinguish between, e.g.,

- trucks or tractors with a length of two cells and a lower admissible maximum speed,
- motorcycles and bicycles who can ride next to each other in pairs, and
- “Sunday drivers” who are mostly on the road on weekends and with a higher probability of dallying.

If we admit inhomogeneous traffic participants it is especially clear that our simple model is unrealistic: It does not allow for any *passing maneuvers*. This means that slower traffic participants can block the entire road while all faster vehicles inevitably stack up to create a traffic jam. The passing maneuver must therefore be incorporated into the model. For this purpose, the oncoming lane must be checked. If there is sufficient road clearance in order to complete the passing maneuver safely, then a vehicle may pass. Here again, further aspects have to be considered: The passing vehicle should switch back into the original lane in a way that makes it not necessary for the passed vehicle to decelerate; there is no dallying during the passing maneuver; the one who is being passed must not increase his speed during the passing maneuver;...

As holding especially true on highways, multi-lane traffic must be permitted. But this also occurs in inner-city traffic. In the model, the rules for line changing must be obeyed. A lane change can only be performed if no other vehicle is hindered. For two lanes, a simultaneous change from the left to the right lane and vice versa must not occur. Particular attention must be given to highway and interstate on- and off-ramps as well as intersections found in inner-city traffic in which free scripted lane changes are not permitted. If our model allows for several lanes and vehicles having different maximum speeds, the resulting fundamental diagrams obtained from a simulation come very close to the ones from actual highways.

If a traffic jam forms on a highway, consequences can develop that involve the traffic on the neighboring roads as well as more distant highways: Whenever

possible, drivers attempt to circumvent the traffic jam using local or even more distant detours. In order to integrate this into the model, alternative routes have to be computed that take into account relevant parameters such as the length of the traffic jam. This could be accomplished through static route planning for a traffic participant or through a local shortest path search that looks for the next vertex of the previous route plan after excluding the edges affected by the traffic jam. Given the result (detour, estimate of the time in the traffic jam) one needs to decide by means of tolerance values whether the route is changed or not.

To obtain a more realistic traffic behavior, the refinement of the randomization factor may be considered. For example, empirical observations show that the daily probability is higher when first engaging the clutch, or when accelerating as compared to decelerating. This has a particular impact to the temporal development of traffic jams. But it is also possible to represent regional or national characteristics as well as properties of vehicle types. When compared to Europe, for example, significantly more vehicles making up American road traffic are equipped with cruise control. This significantly reduces the daily probability during the free flow phase (i.e., for $v = v_{\max}$).

For our model, since collisions are not allowed, vehicles can easily decelerate from 135 km/h down to zero within a second. Reality dictates otherwise for road traffic since the braking behavior depends on whether the vehicle in front and even possibly the additional vehicles further ahead decelerate. The information about the deceleration is passed on via brake lights as is often witnessed on highways with the quickly expanding waves of brake lights seen at the ends of traffic jams. Two influencing factors include the estimated velocity of the predecessor as well as the relative safe distance to be kept between vehicles depending on one's own velocity. For the model, we essentially are forced to realize an anticipatory driving style.

Relative to available computing power, if we can afford a higher resolution of the network and therefore of the speed levels, we can discretize roads into finer cells. After dividing the cell length in half to 3.75 m, an average vehicle would occupy two cells. With a time step of 1 s, we could then distinguish ten speed levels up to the same maximum speed. By now, there exist models with cell lengths of up to 30 cm.

Literature specializing in this area include the discussion and evaluation of results for many model refinements. Of special interest is the inclusion of the influence of psychological factors which opens up diverse approaches for improvement. Hereby, the empirical determination of parameters (e.g., the probability of selecting an alternative route given a traffic jam or the value of different randomization factors) plays an important role. Unfortunately, comprehensive real network data with sufficient information as well as measured data are oftentimes only difficult or very expensive to obtain.

8.6 Summary and Outlook

We have introduced a simple, microscopic traffic model based on stochastic, cellular automata. Despite all the model simplifications, this is meaningful enough to simulate road traffic and to observe and explain traffic phenomena such as traffic jams out of nowhere.

Already for a simple, perpendicular road network as in Mannheim or Manhattan, one discovers that without prescribed rules for the right-of-way (in particular traffic lights) a higher traffic density quickly leads to the collapse of traffic flow. Traffic continues to build up around neighboring blocks, and wide-ranging gridlocks form. But even in the presence of traffic lights there exists a large difference whether the traffic light control system is simple or optimized. If one inserts a diagonal road into an (artificial) perpendicular network, a disruption of the traffic at intersections may result. This was precisely the case in Manhattan: By blocking a large diagonal road, the traffic flow was improved. This road prevented the green light phases of the remaining perpendicular roads from being optimally adjusted and compatible to each other [34].

Models that are based on the simple, basic framework introduced find and have found diverse applications. In North Rhine-Westphalia, a state in Germany, they serve as predictors of traffic jams. The models were also used to simulate the entire traffic of individuals in Switzerland. They serve as a detector of environmental pollution due to road traffic and, as well, the model finds an application in the traffic simulation software TRANSIMS (developed in the Los Alamos National Laboratory [39]) for the simulation, prediction, and analysis of the effects of traffic, such as, e.g., air quality.

Furthermore, similar models that are based on cellular automata can be developed and adapted to even more scenarios. As an example, we want to mention simulations of evacuations. Here, the “traffic participants” are pedestrians which move within a two-dimensional CA, for example, on hexahedra with a diameter of 80 cm that corresponds to the typical space required for a human. A further look into the literature shows the diversity of the possible applications of cellular automata in simulation, whether for the simulation of gases or the expansion of forest fires.

Chapter 9

Stochastic Traffic Simulation

The third variant for the simulation of traffic considers particular characteristics of traffic such as those observable in computing systems.

To this end, we imagine a university computer room housing many computers. The computers are all part of a network, and the users have the ability to print from a central printer via the network. The complaints are escalating that it takes too long to complete a printing order. To solve the problem, you are tasked to buy a new printer. But how fast and powerful does the new printer have to be so that its capacity suffices? An industrial high-end printer that is employed for “newspaper on demand” would be cost prohibitive, excessively powerful and oversized. Additionally, the purchase prize would be hard to justify! Further, is the printer the actual cause for the problem? If one can only print via the central computer, it is reasonable to question whether the bottleneck could also be within the network and not the printer. Buying a new printer may be simply trusting on good luck that the problem will be fixed.

Or let us consider a telecommunication company. The company’s customer base has increased significantly. Does it now have to extend its line network? Does it need new splitters, thicker cables or additional cell towers? And if so, where? Where does the network first reach its boundaries for capacity, where is the proverbial bottleneck?

Even for systems that have nothing to do with data traffic these and similar questions have to be analyzed and answered—whether at the classical post office with several counters and one or several lines, the toll booth on a section of a highway or the process planning needed at a factory or company: Where is the most critical point in the system? How long do people, vehicles, or orders have to wait before it is their turn? In general, how many do concurrently wait?

From the point of view of the owner or operator, i.e., the administrator, branch manager or toll both manager, these are—once again—sub-questions for a larger optimization problem: How do I maximize the payoff for the given costs? Here, the payoff can obviously be of varying type: for example higher earnings through the closing of a booth at the toll plaza or alternatively through a higher traffic

volume due to shorter waiting times as a result of opening additional booths; the satisfaction and possibly even the resulting gain in loyalty of customers and much more. The individual optimization goals could be in mutual conflict: For example, at a post office branch, the ideal case has the average waiting time of the customers minimized while simultaneously the expected idle time of its personnel is also minimized.

In order to answer these questions we must model and simulate the traffic flow of jobs through the respective systems—as done before for the macroscopic and microscopic traffic simulation in Chaps. 7 and 8. Here, the jobs are printing jobs, telephone calls, customers or vehicles. The natural question comes up why we don't simply go back to the macro- and microscopic modeling and simulation techniques that were already introduced.

Reflecting on those aspects we want to take from both worlds, we find out that an alternative approach makes sense. As in the macroscopic traffic simulation, we are more interested in average quantities for the above mentioned scenarios (even though for example depending on the time of the day) than in the individual job; nonetheless, we want to resolve the individual jobs as we did with the microscopic simulation—however, not as detailed: From the point of view of a printer, the life story of a print job is negligible. Important is when the job arrives and the time it takes to process it. We are less interested in the exact behavior of jobs between two points (spatial development of the road traffic and exact representation of passing maneuvers on a long road, or the route to the post office), but rather interested in the behavior at certain points (intersection with traffic lights or toll plaza, post office). Further, we cannot assume, for example, that print jobs travel in wave form or mostly arrive at the printer in packs.

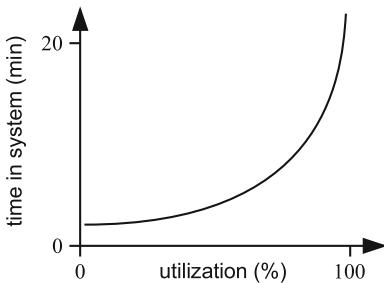
The events occurring behind the scene that produce a job entry could be far too diverse for modeling. With a restricted, partial view they are unpredictable and cannot be assigned or determined. One only has to think about the diversity in rationale that could result in pushing the print button or in making a phone call.

Both the restricted view on the cause of the job as well as our interest in expected, average quantities suggests a stochastic approach. We therefore want to take a closer look at the *stochastic simulation* of traffic using *queueing models*. For simple cases, the models can still be treated analytically with the desired quantities resulting. But for more complex systems we will have to rely on simulation. Since we are only interested in the events themselves and not in the interim time periods, we will discretize the time by the events and we will simulate discretely and event-based. From the tools introduced in Chap. 2, we essentially require Sect. 2.3 concerning stochastics and statistics.

9.1 Model Approach

The modeling goal for the system is the quantitative analysis to assess its performance. Let us consider a simple example: For reasons of illustration we fall back to our good old post office (even though it is more and more in danger of becoming

Fig. 9.1 Average length of stay of a customer in the post office depending on the workload of the counters (qualitative progression)



extinct) rather than a computer system or telecommunications network. As already described, optimization goals, for example, are defined to make customers as well as the post office as happy as possible. Thus, the goals are to minimize the customers' waiting time while simultaneously maximizing the workload at the counters.

It is impossible, however, to reach both goals simultaneously. Direct considerations lead to a qualitative progression similar to the one in Fig. 9.1. To make quantitatively sustainable statements, e.g., pairing the expected average waiting time of a customer given the workload, we must first model the post office and its environment before we can then analyse and simulate the resulting model.

Let us begin by studying the post office itself. We can model it as a simple *wait and serve system* as illustrated in Fig. 9.2. Customers (*jobs*) enter the post office (system) at the entrance, line up at the end of one of possibly several lines and wait until they are served by an employee at an open counter. Afterward, they leave the system again through the exit. The model parameters of interest include the number of open counters, the capabilities of the individual counters or employees, and how fast they perform their duties.

When modeling the environment we should only take into consideration what is important for the post office itself. It is impossible to include the entire world in the model. We therefore restrict ourselves to the two interfaces to the environment, the entry and exit. If we assume that customers leave the shop as soon as they have been served, our focus reduces to the *job entry*, or in our case, customer arrival. The frequency or, equivalently, the time intervals in which customers arrive are of interest. We must also think about what they want (buy stamps, mail or pick up packages, ask for information, ...).

For the input parameters of our system we can quantify typical customer behavior via empirical measurements; for traffic counts pertaining to road traffic, this is exactly what happens. From this, we can either use the obtained data directly, e.g., for a simulation, or alternatively, we can use the data to derive distributions or relationships that in turn can be used to provide analytic statements or to produce synthetic, representative input data.

As always, the selection of model parameters confronts us with the ubiquitous modeling problems: Do we recognize those parameters that are of importance? Do we over- or underestimate their impact? Do we recognize all the important

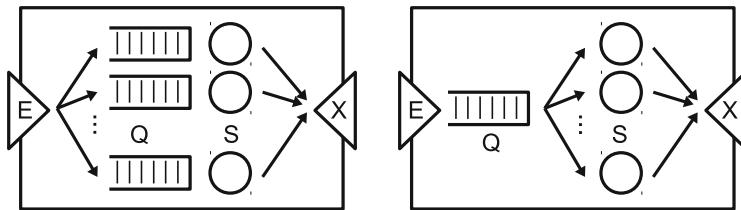


Fig. 9.2 Basic structure of the wait and serve system at the post office. Next to the entry and exit there are one or more queues (Q) as well as service counters (S)

dependencies between the parameters? And how do we handle quantities of influence that are difficult or impossible to model? (What is the influence of the weather on the number of packages that are mailed?) In particular, so-called *feedback effects* are difficult to grasp: If it is well-known that one is served quickly at a post office, then more customers may come from other districts, or customers tend to switch to another line thinking that the other lines always move faster than their own.

Once our model is complete, we can begin to make statements about the performance of the post office or its individual components. To this end, we must either examine the model analytically or rely on simulation. The former is possible either for simple problems or asymptotically for limiting cases. When systems become more complicated, however, the (numerical) simulation becomes the only option. To this end, we require the standard practices in the field of statistics, that is, we must perform suitable tests. The independence of the observations must be ensured as much as a good “randomness” of the *pseudo random numbers* produced by the computer.

9.2 Queuing Systems

Let us now address everything a bit more formally. In the simplest case we speak of an *elementary queuing system* QS (see Fig. 9.3), also called *service station*). This consists of a *queue* (Q) (or also a *queuing unit*) in which incoming jobs are lined up and one or several identical *service units* (SU) that can process jobs in parallel. A single job always occupies only one service unit. An example from the computer world would be a buffer for arithmetic operations and several identical ALUs (arithmetic-logical units) on a processor. Instead of m identical service units one can refer to one service unit of *capacity m*. The queue and the m -fold service unit are also called functional units. Jobs leave the queuing system after successful processing.

Furthermore, we require a model for the processing of orders. For order entry and service, questions arise as to “when?” and “how long?”. Important model parameters

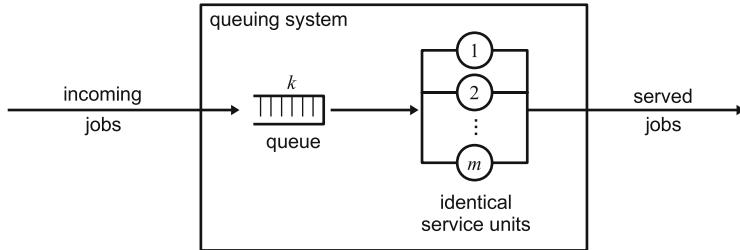


Fig. 9.3 Elementary queuing system, consisting of a queue of capacity k and m identical service units

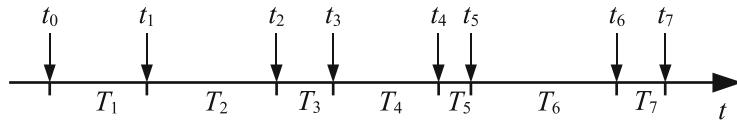


Fig. 9.4 Times of events and interarrival times

are obviously the number of service units m and the (maximum) capacity k of the queue. The process detailing the movement of the orders from the queue to the service unit is also of interest.

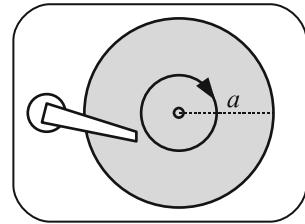
9.2.1 Stochastic Processes

Let us first consider the arrival of orders. Orders arrive at certain times t_i in the queuing system. Between arrival events elapses a time span, T_i , that gives the *interarrival times*, see Fig. 9.4. We can therefore describe the incoming order via points in time as well as via their temporal spacings as will be the case for the following.

The ideal case prescribes that the orders arrive deterministically and the intervals between orders are all of equal length, for example. Typically, this is not the case: An event (customer arrival, occurrence of a bit error, ...) can have too many different causes for us to be able to capture and model all of them. We therefore use a stochastic model and assume that the interarrival times obey a certain probability distribution that can be provided. We call the average number of orders that arrive per time unit the *arrival rate*, λ . It results from the reciprocal of the average interarrival times, $\lambda = 1/\bar{T}_i$.

We therefore view the interarrival times T_i as *random variables*. The sequence $\{T_i, i \in \mathbb{N}\}$ describes similar, time-wise ordered processes and is called a *stationary stochastic process*. Stationary means that the distribution of the number of events in the interval $[t, t + s]$ depends only on the interval length s and not on the start time t , i.e., arrivals are completely random and there are no peak times or slack

Fig. 9.5 Schematic representation for disk storage: The remaining rotation time for random order arrival amounts to at most the time period for one rotation, a



periods. As an illustration, we can think of a random number generator that provides us with a sequence of random numbers $\{T_i\}$ for interarrival times that namely are independent of the previous history. (Unfortunately, this is not the case for pseudo random number generators in the computer.)

The assumption that interarrival times are independent and obey the same distribution (*iid: independent, identically distributed*) is not necessarily realistic here: Vehicles that come to pass a check point on a road are, e.g., extremely sociable, average daily temperatures vary in dependence of the season, and the number of shoppers at the supermarket varies strongly with the time of the day. For many of the previously mentioned applications, however, this assumption is not too restrictive. This holds in particular if we consider long-term average values.

How shall we choose the common density f_T and distribution F_T of the T_i ? As an example, let us consider the *remaining rotation time* for disk storage, i.e., the time from the random order arrival up to the time at which the requested block passes through the read head, see Fig. 9.5. Let the maximum remaining rotation time be a , i.e., the time period of a complete rotation of the storage medium.

As long as no additional specifications are made on the distribution of the blocks, we can assume a uniform distribution of the data. The density and distribution of the remaining rotation time are thus, respectively,

$$f_T(t) = \begin{cases} a^{-1} & \text{for } t \in [0, a], \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad F_T(t) = \begin{cases} 0 & \text{for } t < 0, \\ t/a & \text{for } 0 \leq t \leq a, \\ 1 & \text{for } t > a. \end{cases}$$

As expected, the remaining rotation time that is to be expected on average equates to half of the rotation time: $E(T) = a/2$.

Let us think of ourselves as outside observers. We have been waiting for the desired block to arrive under the read head since the arrival of the request. If the time t has already passed, we are interested in the probability that the event occurs “right now” in the next interval $[t, t + \delta t]$, i.e., the conditional probability

$$P(T \leq t + \delta t | T > t) = \frac{P(t < T \leq t + \delta t)}{P(T > t)} = \frac{F_T(t + \delta t) - F_T(t)}{1 - F_T(t)}.$$

It increases monotonically with increasing δt (as a distribution function, F_T is monotonically increasing).

If we consider the limit as $\delta t \rightarrow 0$, we then obtain the so-called *hazard rate*,

$$h_T(t) := \lim_{\delta t \rightarrow 0} \frac{P(T \leq t + \delta t | T > t)}{\delta t} = \frac{f_T(t)}{1 - F_T(t)}.$$

It describes the momentary rate at which an event occurs at time t , or at which the status changes if this has not yet been the case at time t . The hazard rate, $h_T(t)$, is so-called because it is an important quantity for the description of “hazards” (the breakdown of a machine, radioactive decay, sale of a product, ...).

For our example involving the remaining rotation time we find that,

$$h_T(t) = \begin{cases} \frac{1}{a-t} & \text{for } 0 \leq t \leq a, \\ 0 & \text{otherwise,} \end{cases}$$

which is not a surprise to us. The event has not occurred before time t —we have already waited this long. It must occur before time a , i.e., the closer t gets to a , the more compelling the chance for the desired block to reach the read head “immediately”. There is no way that the event can occur after time a .

It becomes even more exciting if we return to the examples concerning the post office, the initially described network printer or observing the time period elapsed before the next death occurs in the Prussian army that is due to an accidental kick from a horse (which we will come back to later). What are the probability densities and the distributions of customer or print job arrivals or even horse kicking events? Common to all three examples is the existence of many different independent but unlikely reasons for the occurrence of the end of the time interval. The *event risk* is always at the same level and it is independent of the time that we have already waited.

This implies that the hazard rate is constant, $h_T(t) := \lambda$, for example. Given that $f(t) = F'(t)$ and from *separation of variables* and the general *normalization condition*,

$$\int_{-\infty}^{+\infty} f_T(t) dt = 1,$$

we can determine the corresponding density and distribution functions for the interarrival times as an *exponential distribution* with parameter λ ,

$$f_T(t) = \begin{cases} \lambda e^{-\lambda t} & \text{for } t \geq 0, \\ 0 & \text{for } t < 0 \end{cases} \quad \text{and} \quad F_T(t) = \begin{cases} 1 - e^{-\lambda t} & \text{for } t \geq 0, \\ 0 & \text{for } t < 0. \end{cases}$$

The expected value for the interarrival times is $E(T) = 1/\lambda$.

We go another step further and use the function $N(t)$ to count the events in a time interval of length t . $N(t)$ is then a *Poisson distribution* with parameter $\vartheta = \lambda t$, i.e.,

$$P(N(t) = i) = \frac{e^{-\vartheta} \cdot \vartheta^i}{i!}.$$

The expected value is then,

$$E(N(t)) = \vartheta = \lambda t,$$

i.e., we can expect λ events per time unit. This also makes sense graphically: If the expected value of the interval defining the time lapse between arrival times is, e.g., $E(T) = 1/\lambda = 1/2$ min, we expect on average $E(N(1)) = \lambda = 2$ arrivals per minute.

Once again, we have that $\{N(t), t \geq 0\}$ is a stochastic process, i.e., a *Poisson counting process*, whose interarrival times are negative exponentially distributed with parameter λ . With $X := N(1)$, X is also a Poisson distribution but with parameter λ ($= \vartheta$). The parameter λ is hence also called the *event rate* or, in the case of an *arrival process*, the *arrival rate*.

The fact that counting of an event whose occurrence can be sourced to many independent, unlikely reasons is Poisson distributed was discovered long ago by Ladislaus von Bortkewitsch who analyzed a data set that had been accumulated over 20 years [8]. That particular list recorded the number of deaths by hoofbeat in the Prussian army by army corps and calendar year. The underlying assumption of *lack of memory* (the probability of a lethal kick by a horse is always the same and independent of the previous death) has far-reaching consequences as will be illustrated in the next section.

The Hitchhiker's Paradox

Isn't it universally known that the next bus makes us wait much longer at exactly the time when we get to the bus stop? Interestingly, this is no illusion—assuming that the busses do not keep to their schedules and appear with exponentially distributed interarrival times! This phenomenon is called the *hitchhiker's paradox*.

In fact, this is similar to what happens to a hitchhiker who comes to a country road. Let the time span T_i (interarrival times) represent the time between vehicles that come by a point on a country road and assume they are independent and negative exponentially distributed with expected value $E(T) = \gamma = 10$ min (i.e., with parameter $1/\gamma$). The hitchhiker arrives at this point at some random time, t_0 . If the intervals between two vehicles are on average 10 min, how long is the hitchhiker expected to wait until the next car arrives? It is not, as one could intuitively assume, $\frac{1}{2}E(T) = 5$ min.

The reason rests on the lack of memory of the exponential distribution. How large is the *remaining time* until the event, that is, after the time t_0 has already elapsed? It holds that,

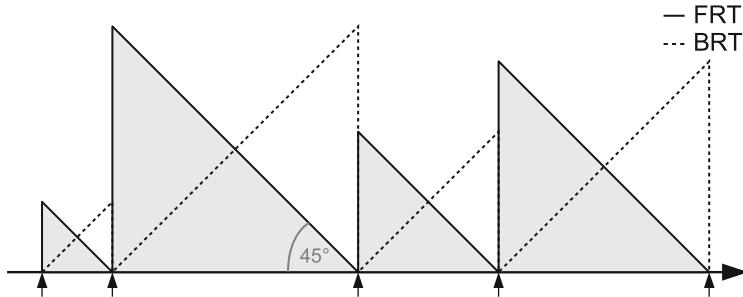


Fig. 9.6 Hitchhiker's paradox: Representation of the remaining time (forward recurrence time, FRT) and the time already passed (backward recurrence time, BRT) while waiting for the next event

$$\begin{aligned}
 F_{T|T>t_0}(t) &= P(T \leq t \mid T > t_0) = \frac{F_T(t) - F_T(t_0)}{1 - F_T(t_0)} \\
 &= \frac{1 - e^{-\frac{1}{\gamma}t} - 1 + e^{-\frac{1}{\gamma}t_0}}{1 - 1 + e^{-\frac{1}{\gamma}t_0}} = 1 - e^{-\frac{1}{\gamma}(t-t_0)} = F_T(t - t_0),
 \end{aligned}$$

which tells us that the remaining time is distributed the same way as T itself. It follows for the expected value that,

$$E(T \mid T > t_0) = t_0 + E(T) = t_0 + \gamma.$$

Considerations respective to the time already passed apply. For the hitchhiker, this means that the previous car was expected to have come by $E(T) = \gamma = 10$ min ago, but that he can expect the next car also in only $E(T) = \gamma = 10$ min. Therefore, he finds himself, as paradoxical it may sound, in an interval that is on average twice as long as $E(T)$.

Let us consider the situation more closely in which an outside observer looks at what is happening at a random point in time and for interarrival times having arbitrary distributions. Figure 9.6 shows the schematic representation of the situation in which the remaining time (*forward recurrence time*, FRT) and the time that has passed already (*backward recurrence time*, BRT) are plotted with respect to the time the observation was made. If our observation time coincides with the beginning of an interval, then we have the entire interval length still ahead of us, while an observation at the end of the interval means the event is imminent, and when observed at some time in-between the remaining time decreases linearly.

How large is the expected remaining time? We can compute it via the areas defined by the saw teeth: For n intervals T_1, \dots, T_n from t_0 to t_n , the average remaining time is

$$\overline{\text{FRT}} = \frac{\int_{t_0}^{t_n} \text{FRT}(t) dt}{t_n - t_0} = \frac{\sum_{i=1}^n \frac{1}{2} T_i^2}{\sum_{i=1}^n T_i} = \frac{1}{2} \frac{\frac{1}{n} \sum_{i=1}^n T_i^2}{\frac{1}{n} \sum_{i=1}^n T_i}.$$

In the limiting case for $n \rightarrow \infty$ we obtain the expected value,

$$E(\text{FRT}) = \frac{1}{2} \frac{E(T^2)}{E(T)}.$$

With standard deviation $\sigma(T)$ and the coefficient of variation $\varrho(T)$ this becomes

$$E(\text{FRT}) = \frac{1}{2} \frac{\sigma(T)^2 + E(T)^2}{E(T)} = \frac{E(T)}{2} \left(1 + \frac{\sigma(T)^2}{E(T)^2} \right) = \frac{E(T)}{2} (1 + \varrho(T)^2).$$

The analog obviously also holds true for the backward recurrence time. The expected value of the remaining time thus depends directly on $\varrho(T)$, i.e., on how much the random variable T scatters:

- If the busses at the bus stop follow the schedule and deterministically arrive every 10 min, then it holds that $\varrho(T) = 0$, and all is well in our world: Given a random arrival time at the bus stop we must, on average, wait $E(\text{FRT}) = E(T)/2 = 5$ min.
- For a uniform distribution on $[0, a]$ as holds for the case of the remaining rotation time, it at least falls below the expected value of the interarrival times from the perspective of an outside observer: For $E(T) = a/2$ and $\varrho(T) = 1/\sqrt{3}$ it holds that $E(\text{FRT}) = a/3 = 2/3 E(T) < E(T)$.
- If cars or busses randomly arrive six times per hour with exponentially distributed interarrival times, then with $\varrho(T) = 1$ the case that our hitchhiker had to suffer through indeed occurs: $E(\text{FRT}) = E(T)$.
- Finally, for a process with $\varrho(T) > 1$, the expected remaining time is larger for an observer than the average length of the intervals between events!

A visual interpretation is as follows: The greater the scatter in the interarrival times, the more likely it will be for an outside observer to land in a large interval rather than a more preferable short interval.

Stochastic processes, especially Poisson processes, appear in very different applications; in any place where temporally ordered, random processes appear or are required, e.g., in mathematical finance or physics.

For the processing of orders, the same considerations apply as held for the arrival of orders. For the example involving the remaining rotation time we had waited for the arrival of a block under the read head: just as well we could have talked about the length of the order processing of the read request and about a *service process*. Instead of the interarrival times T , we would then speak of *processing* or *service times* B , and we would speak of the *service rate* μ instead of the arrival rate λ .

9.2.2 Classification of Elementary Queuing Systems

For the uniform description of a queuing system, the so-called *Kendall notation* has become widely accepted,

$$A/B/m[/ k / n / D].$$

The first three parameters go back to Kendall himself and describe

- A*: the distribution of the arrivals (or the *arrival process*),
- B*: the distribution of the service times (or the *service process*) and
- m*: the number of identical service units working in parallel.

In order to be able to describe a wider bandwidth of queuing systems, they have been extended by up to three additional, optional parameters, namely,

- k*: the size or capacity of the queue,
- n*: the number of all orders in question (world population) and
- D*: the discipline of the queue (or the *service strategy*).

Arrival process *A* and service process *B* are stochastic processes. The interarrival and the service times are subject to a certain respective distribution. For *m* identical service units, up to *m* orders can be processed simultaneously and independently of each other with the same service rate.

The fourth parameter, the capacity of the queue *k*, is the maximum number of orders than can wait for the order processing. If additional orders arrive, they will be rejected. If the parameter is not provided one then assumes *k* = ∞ .

The size of the world population *n* represents the maximum number possible for all orders in question. It is mostly of interest if closed queuing systems are considered, i.e., if orders do not leave the system but are permanently processed further. In the case of open systems this parameter is typically dropped and it then holds that *n* = ∞ .

Lastly, the *queuing discipline D* is still missing. It describes the order in which the waiting jobs are serviced. Here one distinguishes between *non-preemptive* and *preemptive service strategies*.

Non-preemptive strategies have the advantage that no additional overhead results from the abortion of orders that are currently processed. One speaks of a *fair service strategy* if it is impossible for an order to wait infinitely long while some subsequently arriving order only has to wait a finite time. Typical non-preemptive strategies include among others:

- *Random*: The next order is selected randomly.
- *FCFS* (first-come-first-served): As in the post office, the first one in the queue is served first.
- *LCFS* (last-come-first-served): The typical work place strategy—new orders are placed on top of the stack, and one picks up from the top; the lowest one is out of luck.

- *Priority based:* The most important order comes first—practical at the emergency room in the hospital.

For preemptive strategies it cannot happen that a large order completely occupies the queuing system over a very long time period, which ensures a certain degree of “fairness” for greatly varying service times. In addition, it can be worked in a way that is dependent upon the service time without having to know the exact service times or at least B . The disadvantage of preemptive strategies is the overhead that results from the displacement mechanism: An order that is already in the processing stage must be put on hold and the current state must be saved so that it can be continued at a later time. If the processing time per job is so short that it is mainly used to reconstruct the previous state, then it turns to thrashing: The processes require more time to load pages from the storage than for the execution. Examples for preemptive strategies are:

- *RR* (round robin): Each job is serviced one after another for a fixed time span. If it is not completed by then, it is interrupted and put to the end of the queue which is processed FCFS. For small time discs one also speaks of PS (processor sharing) since this corresponds to the multitasking of operating systems.
- *LCFS-preemptive:* A newly arriving order is processed immediately.
- *Priority based, static:* A newly arriving job with higher priority replaces the currently processed job—the typical “fire-drill” job announced by the boss.
- *Priority based, dynamic:* The job priorities can change with time, e.g., to SET (shortest elapsed time, the minimal service time up to now wins) or SERPT (shortest expected remaining processing time).

From a general point of view, the model with displacement becomes significantly more complicated for analytical investigations since the computation of service and waiting times becomes more complex. For the simulation it becomes necessary to contemplate the efficient realization of the queue and to choose a data structure that is optimized for the selected discipline. If the queuing discipline is not stated explicitly, it is assumed to be FCFS.

9.2.3 Examples for the Kendall Notation

Typical abbreviations for the distribution of interarrival and service times are:

- D: Deterministic distribution, constant time intervals, no stochastics
 M: Exponential distribution (“memoryless”)
 G: General distribution, i.e., distributed arbitrarily. Very powerful, but also complex

A few examples using the Kendall notation are now given,

M/M/1: A simple (and yet interesting) non-deterministic system. It is a *single-server queuing system*, in which arrival and service process are

exponentially distributed. Here, we only need to know the arrival and service rates λ and μ . It is simple and treatable and is suitable for the approximation of simple, real systems such as a post office with a single counter.

$M/G/1$: If we know more about the service process, then we can select a better distribution for it. If the occurrence of errors for machines in a factory is a Poisson process, since memory-less, but we can draw from experience and not only provide the expected value of the repair time but also the variance or even the distribution of an individual mechanic, then we can choose $M/G/1$. One additional example is the simulation of a central processing unit (CPU).

$G/M/1$: Here we model additional knowledge about the environment, but not about the service; less potential for applications.

$G/G/n$: A flexible *multi-server system*, that is hardly treatable analytically.

9.2.4 Performance Indices and First Results

With this preliminary work we can now turn to the performance indices of the queuing system. A quantity that is of particular importance from the perspective of the customer is the *total time spent in the system*, y . This is the time that a job stays in the queuing system from its beginning to its completion. This corresponds to the sum of the *waiting time* w and the *service time* b , i.e., $y = w + b$.

The *number of jobs in the system* f provides the number of jobs within a functional unit or the entire queuing system. One distinguishes between *empty* ($f = 0$), *busy* and *occupied* (all waiting spots or service units are occupied) functional units or queuing systems, resp. For $f = 1$ it holds that $w = 0$ and therefore $y = b$, i.e., the time spent in the system corresponds to the service time.

The *throughput*, d , is the average number of jobs completed within a time interval. For a pure queue, the long-term throughput corresponds to the arrival rate, $d = \lambda$, and for a service unit it corresponds to the service rate, $d = \mu$. If we consider an elementary queuing system, then we must distinguish between two cases: For $\lambda < \mu$, the long-term throughput corresponds to the arrival rate, and for $\lambda \geq \mu$ to the service rate (at a continuously growing length of the queue). The *maximum throughput* c is defined as the maximum throughput that is possible.

The *utilization* $\varrho = d/c$ is the relative throughput. For a queuing system at full utilization it holds that $\varrho = 1$. If the customer arrival rate is larger than the maximum throughput, c , then the service unit is used to capacity in the long-term; customers must line up at the end of an ever increasing line and prepare for increasingly longer waiting times. For only a slightly utilized queuing system ($\varrho \approx 0$), customers can get to an available service unit right after their arrival without having to wait, compare this with Fig. 9.1 that was initially shown.

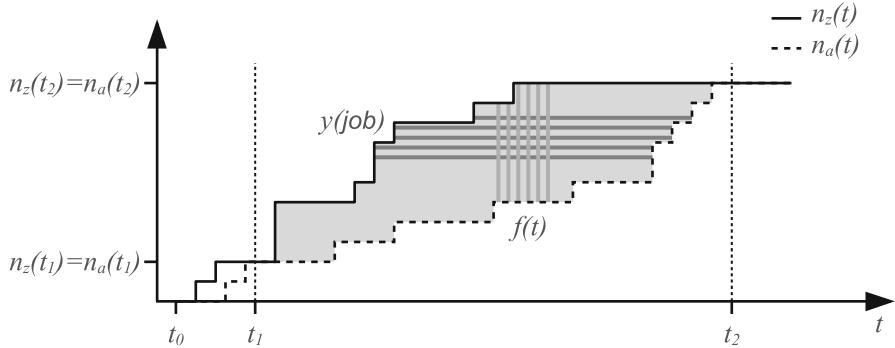


Fig. 9.7 Illustration of Little’s law: Number of arrivals $n_z(t)$ and departures $n_a(t)$ in the interval $[t_1, t_2]$. $y(\text{job})$ is the total time spent in the system depending on the job, $f(t)$ the time-dependent number of customers/jobs for the considered system

Little’s Law

For a given queuing system, if we consider the connection between the performance indices “number of customers”, throughput, and “time spent in system”, we then acquaint ourselves with a central law of traffic theory, *Little’s law*. To this end, we measure the number of arrivals, $n_z(t)$, and the departures, $n_a(t)$, from some starting point in time, t_0 . The “total number of customers in the system” at time t is then $f(t) = n_z(t) - n_a(t)$, as seen in Fig. 9.7.

We now consider a time span $[t_1, t_2]$ in which the system is empty both at the beginning and the end, i.e., $f(t_1) = f(t_2) = 0$. It then can be seen that the throughput is given by,

$$d(t_1, t_2) = \frac{n_a(t_2) - n_a(t_1)}{t_2 - t_1},$$

while the average number of jobs in the system,

$$\bar{f}(t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} f(t) dt,$$

and analogously, the average time spent in the system is,

$$\bar{y}(t_1, t_2) = \frac{1}{n_a(t_2) - n_a(t_1)} \int_{t_1}^{t_2} f(t) dt.$$

Together, these three quantities yield Little’s law:

$$\bar{f}(t_1, t_2) = d(t_1, t_2) \bar{y}(t_1, t_2). \quad (9.1)$$

Table 9.1 Overview of indices of elementary queuing systems

Arrival rate	λ
Service rate	μ
Waiting time	W or w , resp.
Service time	B or b , resp.
Time spent in system	Y or y , resp.
Throughput	D or d , resp.
Customers/jobs in system	F or f
Maximum throughput	c
Utilization	ϱ
	$E(F) = E(D) \cdot E(Y)$
	$E(Y) = E(W) + E(B)$
	$E(F) = E(D) \cdot E(Y)$
	$\varrho = E(D)/c$

Instead of the operational/deterministic perspective we could write Little's law stochastically using expected values of random variables as

$$E(F) = E(D) E(Y). \quad (9.2)$$

In the following, we will mostly use expected values which consider performance indices as random variables and denote them using capital letters.

Little's law plays a major role in performance analysis. It applies as much to elementary queuing systems as to the queue or the (m -fold) service unit itself—we need only provide the respective quantities. It even applies to almost all queuing systems and sub-systems, independent of the arrival and service time distributions, the number of service units and the queuing discipline. In particular, Little's law can be used to compute the third of the quantities using the two given ones, or to check the consistency of data obtained in a simulation or by measurements, analogous to the state equation in the macroscopic traffic simulation (see Chap. 7).

Concluding, Table 9.1 again illustrates an overview of the most important indices for elementary queuing systems (capital letters imply stochastic, lower case letters imply deterministic). From these we can derive as results: The maximum throughput of a queuing system depends only on the maximum throughput of the m -fold service unit. It is m times as large as a single (identical) service unit,

$$c_{\text{SU}} = \frac{m}{E(B)} = m \cdot c_{\text{SU}^{(1)}},$$

while for the utilization it holds that

$$\varrho = \frac{E(D)}{c_{\text{SU}}},$$

and for the expected values of the number of customers/jobs in the queue, service unit and queuing system it holds that

$$\begin{aligned}
E(F_Q) &= E(D) E(W), \\
E(F_{SU}) &= E(D) E(B), \\
E(F) &= E(D) E(Y) = E(D) (E(W) + E(B)) \\
&= E(F_Q) + E(F_{SU}).
\end{aligned}$$

9.3 Queuing Networks

So far, we have considered elementary queuing systems. However, for many interesting and important cases this proves to be insufficient. If we want to simulate the traffic in computer systems we then must address the topic of different queuing systems with different service characteristics. The CPU operates differently than the main storage or the hard drive. While a multicore processor can process several tasks simultaneously, in general this is not possible for a printer.

We therefore move forward yet another step and consider *queuing networks*. Here we employ individual service stations as basic building blocks that can be combined by connecting their entries and exits. We can model a network as a *graph*. The *vertices* are service stations while the *edges* represent potential order pathways. An order will traverse serially certain vertices that correspond to the partial orders being processed. The overall network can simultaneously process several jobs. Figure 9.8 shows such an example network.

We distinguish queuing networks into categories that can be *closed* or *open*. For closed queuing networks the number of jobs always remains the same. They never leave the overall system and no new ones are added. Jobs that have been completed are again restarted from the beginning. The number of jobs in the queuing network are always constant and equal to the size of the world population. For open networks, the orders can also arrive from the outside environment. If a job has been successfully completed at one or several service stations, it then leaves the network. The number of jobs in a network typically varies for open networks.

One way of viewing a job is to decompose it into several subjobs each of which have to be processed in a certain order at suitable service stations. At the moment when the jobs leave a service station they then search for a viable service station they can reach that is suitable for the next subjob. Considering the assembly of cars, for example, it is reasonable to assemble the body before the doors and to begin with the varnishing only after the deep drawing. When completing jobs with deterministic arrival times, we can optimize the processing order with the aid of (stochastic) *process scheduling* (Chap. 5).

We mostly are interested here in the state of the network for the long-term statistical average. If knowledge of the specific network package sent from a router to a printer or mail server is of no importance to us and we are content to know that it is on average only every 500th package, then working with probabilities will suffice. The edges of the graph are then weighted by the *transition frequency* as

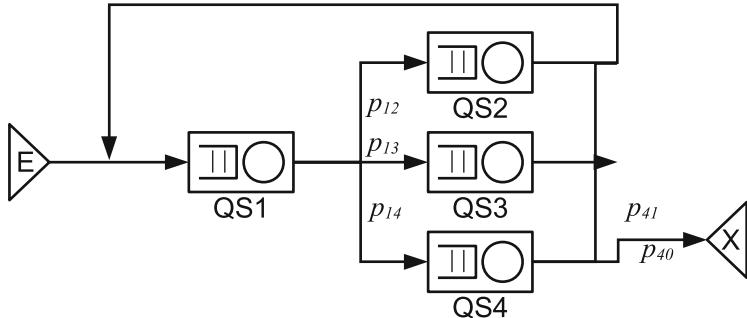


Fig. 9.8 Queuing network, consisting of entry and exit as well as four elementary queuing systems

depicted in Fig. 9.8. The probability that a job is passed from service station i to service station j is denoted by p_{ij} . Here, $i = 0$ implies arrivals from outside the network while $j = 0$ implies that a job leaves the network.

9.3.1 Parameters in Queuing Networks

For queuing networks with N elementary queuing systems we must supply all previous quantities with an index i . $i = 1, \dots, N$ provides the vertex number or, equivalently, the service station number in the network; the index S denotes the respective quantity for the overall system, e.g.,

$$\mu_S, \mu_i, D_S, D_i, \varrho_S, \varrho_i, \dots, \quad i = 1, \dots, N.$$

If in simplifying we only consider orders of a certain type, we then can define the *number of visits*, v_i . Its value gives the number of suborders that require service station i . (If we distinguish between order classes of different character, then v_i also depends on the order type.) The number of visits is given by the ratio between the throughput at service station i to the throughput of the network,

$$v_i = \frac{E(D_i)}{E(D_S)} = \frac{\varrho_i}{E(D_S)} c_i.$$

If we observe a throughput of $1/s$ in the queuing network but a throughput of $4/s$ for service station i , then it must be that the order returned to the service station four times for the processing of a suborder.

An increase of system throughput $E(D_S)$ causes a proportional increase of all individual throughputs, $E(D_i)$, and therefore of all individual utilizations, ϱ_i . If we keep increasing the system throughput, then (at least) one vertex will reach the maximum utilization $\varrho_i = 1$. We label this (not necessarily unique) vertex as a

traffic bottleneck TB in the queuing network. The traffic bottleneck is the critical component in the system for which problems occur first when there are many orders in route. In the case of the network problem described initially the critical question pertained to which component represented the TB. The easiest so-called firefighter strategy of the person responsible for the network would be to simply replace this one component or to optimize it in some other way.

The traffic bottleneck can be described via the maximum utilization,

$$\varrho_{\text{TB}} = \max_i \varrho_i = \max_i \frac{E(D_i)}{c_i}$$

or via the maximum throughput of the queuing network,

$$c_S = \frac{c_{\text{TB}}}{v_{\text{TB}}} = \min_i \frac{c_i}{v_i}.$$

Both perspectives lead to the same bottlenecks:

$$\max_i \frac{E(D_i)}{c_i} = \max_i \frac{E(D_S)v_i}{c_i} = E(D_S) \max_i \frac{v_i}{c_i}.$$

9.3.2 Asymptotic Analysis

In order to assess its overall behavior we can now think about how the queuing network behaves *asymptotically*. To this end, given an order in the system we consider the expected value for both its system throughput $E(D_S)$ as well as its time spent in the system $E(Y_S)$ in relation to the (deterministic) number of jobs, f_S . Here, we specify the number of jobs which is simple in the case of closed networks. To simplify the computations we restrict ourselves to simple service units, i.e., $m_i = 1$.

If at any time there is only one order in the system ($f_S = 1$), then it never has to wait and it holds by Little's law (9.1) that

$$E(Y_S) = E(B_S), \quad E(D_S) = \frac{f_S}{E(Y_S)} = \frac{1}{E(B_S)}.$$

But for a very small number of jobs, it remains that,

$$E(Y_S) \approx E(B_S), \quad E(D_S) \approx \frac{f_S}{E(B_S)},$$

since for a sufficiently large queuing system the orders do not interfere with one another and they essentially do not have to wait. Additionally, the time spent in the system remains largely constant and the throughput grows linearly.

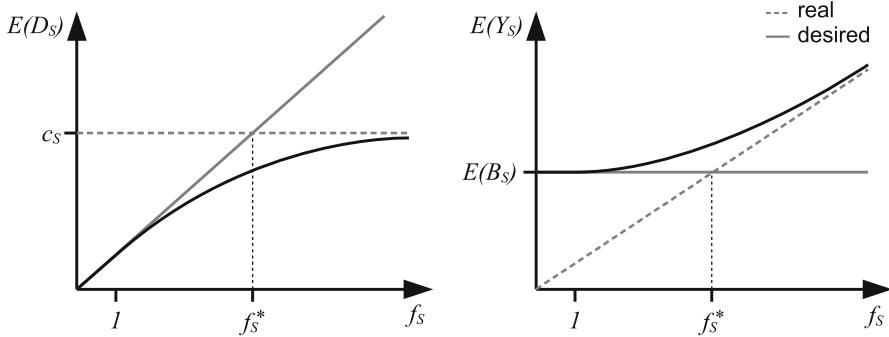


Fig. 9.9 Asymptotic behavior of a system for the expected values of throughput D_S and time spent in the system Y_S in relation to the number of jobs f_S ; f_S^* is the saturation number of jobs

If the number of jobs in the system gradually increases, the mutual interferences of the orders increase as well. The time spent in the system also increases since the waiting time can no longer be neglected while the throughput increases only at a limited rate.

For a very high number of jobs in the system, the orders pile up at the traffic bottleneck. The utilization then becomes maximal there and tends toward the absolute maximum $\varrho_{\text{TB}} = 1$. The system throughput is limited by the traffic bottleneck and it holds that

$$E(D_S) \approx c_S = \frac{c_{\text{TB}}}{v_{\text{TB}}}, \quad E(Y_S) \approx \frac{f_S}{c_S} = \frac{v_{\text{TB}}}{c_{\text{TB}}} f_S.$$

Figure 9.9 shows the asymptotic progression of the expected values of throughput D_S and time spent in the system Y_S in relation to the number of jobs in the system f_S . The intersection of the two asymptotes is given by,

$$f_S^* = c_S E(B_S) = \frac{c_{\text{TB}}}{v_{\text{TB}}} \cdot \sum_i v_i E(B_i)$$

and is called the *saturation number of jobs*. For increasing number of jobs we have to depart from our desired asymptotic behavior even for optimal conditions (no waiting times) since the dominant asymptote changes.

The saturation number of jobs corresponds to the ratio between the maximum throughput and the throughput of a system that is occupied with the number $f_S = 1$ of jobs.

The more bottlenecks, or near bottlenecks, there are, the worse the convergence toward the asymptotes since waiting times will accumulate. The analog holds for the scattering of the times spent in the system and the service times. If the service times are constant, then we reach the asymptotes. If the service times are widely

scattered, however, the times spent in the system then increase and the throughput collapses.

We consider a simple example: In order to analyze the load behavior, we model a computer consisting of a central processor, memory, a hard drive, a DVD and a tape drive as a queuing network comprised of simple service units. For a typical task, the following numbers of visit and expected service times have been determined:

	CPU	RAM	HDD	DVD	TD
v_i	$8 \cdot 10^9$	10^6	60	50	0.1
$E(B_i)$	1 ns	100 ns	10 ms	100 ms	10 s

Which service station is the TB? How large are the maximum throughput and saturation number of jobs in the system? The utilization is maximal at the TB, i.e., with $E(D_i) = v_i E(D_S)$ so it holds that

$$\varrho_{\text{TB}} = \max_i \varrho_i = \max_i \frac{v_i}{c_i} E(D_S).$$

It then holds (simple service units!) that $c_i = 1/E(B_i)$ and, therefore,

$$\varrho_i = (8, 1/10, 3/5, 5, 1)_i \cdot E(D_S),$$

i.e., the utilization becomes maximal for the CPU which is the TB.

In contrast to the service units, it does not typically hold for the maximum throughput that $c_S = 1/E(B_S)$, but it does hold that,

$$c_S = \frac{c_{\text{TB}}}{v_{\text{TB}}} = \frac{1}{8}.$$

An order requires an expected service time of

$$E(B_S) = \sum_i v_i E(B_i) = 14.7 \text{ s},$$

and the saturation number of jobs is $f_S^* = c_S E(B_S) = 1.8375$.

9.4 Analysis and Simulation

We can already model simple wait and service systems and make some initial statements concerning performance parameters and traffic bottlenecks. In the following we want to take an even closer look at how we can analyse and simulate modeled systems. This can only happen on the surface and in segments; a comprehensive treatment would go far beyond the scope of this book.

In many cases, the system behavior of an elementary queuing system can be described sufficiently via a *state quantity*, $X(t)$. All additional performance indices (or at least their respective expected or average values) can then be computed from it. Let us consider again the post office: The possibly most exciting quantity is the number of jobs (or, population) of the system. If the random variable $X(t)$ describes the number of jobs at time t , then $\{X(t), t \in \mathbb{R}\}$ is the corresponding stochastic process for the number of jobs. Since we only admit whole customers it must then hold that $X(t) \in \mathbb{N}_0$. The *state probability*,

$$\pi_i(t) := P(X(t) = i),$$

describes the probability that our system has state i at time t . Obviously, the normalization condition for densities must hold,

$$\sum_{i \in \mathbb{N}_0} \pi_i(t) = 1. \quad (9.3)$$

To begin ($t = 0$), let our post office be empty. Customers arrive and populate the post office; the number of customers, and therefore the state probabilities as well, fluctuate strongly. As long as the arrival and service behavior do not change during the course of time (homogeneity), then at some point, at least considered over a long time span, the state probabilities will remain the same and no longer change with time, i.e.,

$$E(X(t)) = E(X(t + \delta t)) = E(X).$$

The process for the number of customers of the post office is not stationary (the homogeneity does not hold at all instants in time). However, there often exists a *stationary limit process* (stationary for $t \rightarrow \infty$), i.e., a stable distribution of X that is in most cases independent of the initial state. We can distinguish between a *settling phase* or *transient phase* in which many things change, and a *stationary phase*, in which the system is asymptotically stationary.

If we know the initial state of the system (empty post office in the morning), we then can focus our attention on the transient phase, i. e., how the length of the waiting lines develop. For example, we want to determine the time needed before we can expect five customers to be in line since this is when we plan to open a second counter. In general, however, we will be interested in the stationary (limit) state and as such in the long-term behavior of the system. If we know the state probabilities in the post office for the stationary state, we can then compute the expected values of population, throughput and waiting time; the expected service time belongs to the model assumptions and we will come back to that part later.

9.4.1 Markov Processes and Markov Chains

It can be argued that *Markov processes* are the most frequently relied upon models for systems whose temporal development is random. Markov processes are named

after the Russian mathematician Andrej Markov (1856–1922) and will now be introduced and used. We start with a few more general definitions for stochastic processes $\{X(t)\}$, $X(t) : T \rightarrow Z$, with *parameter space* T , mostly called time, and *state space* Z , the set of all elementary events.

With respect to state space Z , we distinguish between *continuous processes*, $X(t) \in \mathbb{R}$, and *discrete processes*, such that Z is countable. If, for example, $X(t)$ measures the temperature (real values), then we are dealing with a continuous process; if $X(t) \in \{1, \dots, 6\}$ represents the numbers on a dice, then we deal with a discrete process which is also called a *chain*. Correspondingly, with respect to the time T , we distinguish between *processes in continuous time* (temperature at every instant in time $t \in \mathbb{R}$) and, for countable T , *processes in discrete time*. The latter consider sequential events such as a repeated toss of a coin, and we can therefore write X_i , $i \in \mathbb{Z}$ in lieu of $X(t)$, $t \in \mathbb{R}$. In the introduction of stochastic processes given in Sect. 9.2.1, the process for the interarrival times $\{T_i, i \in \mathbb{Z}\}$ had been a continuous process in discrete time while the corresponding Poisson counting process $\{N(t), t \geq 0\}$ is a discrete process in continuous time.

A stochastic process $\{X(t), t \in T\}$ is called a *Markov process* if, for all $0 \leq t_1 < t_2 < \dots < t_n < t_{n+1} \in T$, it holds that,

$$\begin{aligned} P(X(t_{n+1}) = x_{n+1} | X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \dots, X(t_1) = x_1) \\ = P(X(t_{n+1}) = x_{n+1} | X(t_n) = x_n). \end{aligned}$$

This means that a new state depends only on the current state and not on the previous history. In particular, the current state does not depend on the initial state. This is called the *Markov property*. Thus, everything that we need to know about the past is contained in the current state.

For a *homogeneous Markov process* (HMP), the transition probabilities are also independent of time and therefore constant, i.e.,

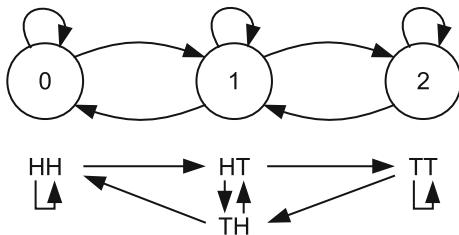
$$P(X(t_j) = x_j | X(t_i) = x_i) = P(X(t_j + t) = x_j | X(t_i + t) = x_i) \quad \forall t \in T.$$

This is, in contrast to the stationarity of processes, no statement on the state probabilities.

A further restriction leads to the subclass of *homogeneous birth–death processes* (HBDP): The state value may only change in increments or decrements of one.

The basic concept behind Markov processes is the existence of states and state transitions. In what follows, we restrict our attention to Markow chains since this is easier to visualize for the discrete case in which there are at most countably many states. It is also better suited for the discrete world of computer systems or for populating the post offices with customers. As expected for the model, we must define the suitable states as well as the state transitions. The potential for the need to handle continuous processes is illustrated in the example given for a *Wiener process* for modeling stock prices that is discussed in Chap. 6.

Fig. 9.10 State graph for the repeated tossing of a coin. The state describes how often heads (H) appeared within the last two tosses (*top*); the state transition depends on the previous history (*bottom*)



One advantage for considering chains is that we can visually illustrate them using *state graphs*. Here, vertices take the role of states while edges depict the possible state transitions. In the time-discrete case, the edges are labeled with the *transition probability* p_{ij} given the respective change from state i to state j ; the sum of the edge weights for all out-going edges must equal 1. For chains in continuous time, the edge weights are *transition rates* λ_{ij} . These specify the transition probabilities per unit of time, i.e.,

$$\lambda_{ij} = \lim_{\delta t \rightarrow 0} \frac{P(X(t + \delta t) = j | X(t) = i)}{\delta t}.$$

Examples of Markov Chains

We will look at several examples, the first not being a Markov chain: Let X_i describe the number of times “heads” appeared for the last two coin tosses. The resulting state graph is then very simple since it has only three states: The homogeneous stochastic process does not, however, satisfy the Markov property: The next state is dependent upon the information pertaining to the previous coin toss. Figure 9.10 illustrates this: The two most recent coin tosses TH and HT both belong to the state $X_i = 1$; for HT , the next state can be $X_{i+1} = 0$ for TT , but for TH this cannot happen. We must consider the sequence of at least the last two tail-head combinations as a state in order to satisfy the Markov property.

If we now define the X_i to represent the number rolled with a dice we obtain six states, $Z = \{1, \dots, 6\}$, see Fig. 9.11. We are now dealing with a homogeneous Markov process; strictly speaking, the successor states are even independent of the current state. Since the state probabilities for a fair dice are $1/6$ from the start (completely independent of the assumed initial distribution!), it even constitutes a stationary process.

A classical example for a homogeneous birth–death process is (besides populating the post office which we will get back to later) a one-dimensional, time discrete undirected *random walk*. A drunk who has left his favorite pub at time t_0 moves one step forward or one step backward per time step with the same probability (see Fig. 9.12). If we watch him for n steps, we consider where he might go as well as how far?

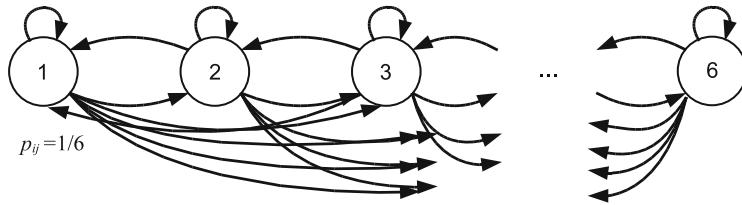


Fig. 9.11 Stationary HMP; the state describes the number rolled with a dice. Hence, all transition probabilities are $p_{ij} = 1/6$

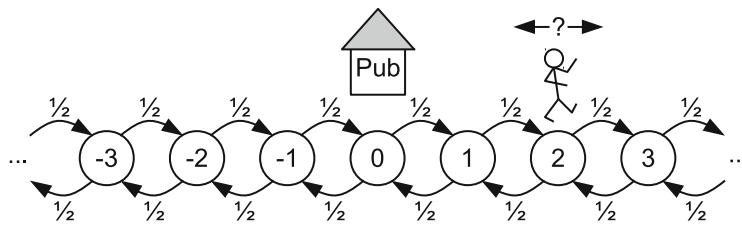


Fig. 9.12 HBDP: Random Walk of a drunk who either staggers one step forward or backward with equal probability per (discrete) time step

We consider a binary random variable Z_i for each step that can assume the values ± 1 with equal probability $1/2$. The stochastic process $\{S_n, n = 1, 2, \dots\}$ with $S_n := \sum_{i=1}^n Z_i$ describes the random walk of the pedestrian. It holds trivially that $E(S_n) = 0$ (symmetric run behavior). Given the *additivity of the expected values* and the *independence* of the Z_i we can compute the variance $V(S_n) = n$. The standard deviation of the pedestrian from the origin is thus $\sigma(S_n) = \sqrt{n}$. On average, he will reach this distance from the pub in n steps. (Note that S_n is a *binomial distribution*.)

If we allowed arbitrarily small time steps such that the step size Z_i had a normal distribution rather than a discrete distribution, the resulting continuous process in continuous time would be a Wiener process that one encounters in the financial world (or in Chap. 6).

An example for the pure birth process is given by the discrete *one-species model* for population dynamics in Sect. 10.4: The discrete HBDP $\{X(t), t \geq 0\}$ in continuous time describes the size of a population. The *birth rate* which provides the number of births per individual and per unit time is γ . Since no one dies in the population the state graph given in Fig. 9.13 results. Here, the transition rate depends on the state and is proportional to the size of the population. Further statements pertaining to this birth process will be given in Chap. 10 on population dynamics for the one-species model in Sect. 10.4.

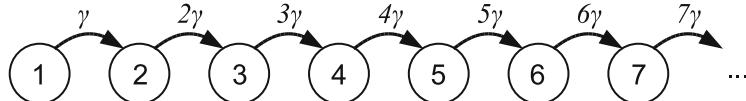


Fig. 9.13 Homogeneous discrete birth process in continuous time for the discrete one-species model (Sect. 10.4) with birth rate γ ; the state is the size of the population

Stationarity of Markov Chains

Since we are generally interested in the stationary, already determined state, the question arises whether or when there will be a stationary (limit) state for a given process $\{X(t)\}$ in which the distribution of $X(t)$ no longer depends on the time, t . In a nutshell, to this end we consider several properties of a chain as well as its states.

For Markov-chains the *Chapman–Kolmogorov equation* holds,

$$\begin{aligned} P(X(t_j) = x_j | X(t_i) = x_i) \\ = \sum_{x_k \in Z} P(X(t_k) = x_k | X(t_i) = x_i) P(X(t_j) = x_j | X(t_k) = x_k) \end{aligned} \quad (9.4)$$

for arbitrary $x_i, x_j \in Z, t_i < t_k < t_j \in T$,

i.e., transition probabilities can also be provided across intermediate states. Given an initial distribution $X(t = 0)$, the state probabilities for subsequent time instants can be uniquely computed.

A homogeneous Markov chain is called *irreducible* if all states are reachable with respect to one another:

$$\forall t \in T, x_i, x_j \in Z \quad \exists \delta t \geq 0 : \quad P(X(t + \delta t) = x_j | X(t) = x_i) > 0.$$

For a given chain, if one is guaranteed to eventually return from a state back to itself, then this state is called *recurrent*. The expected value of the time it takes to return is called the *recurrence time*. A recurrent state with a finite recurrence time is called *positively recurrent* while in the infinite case it is called *null recurrent*. For a *transient* state the probability to return is less than one. If one only returns to the state with an integer multiple of a *period length* $k > 1$, then the state is *periodic*, otherwise *aperiodic*.

For irreducible homogeneous Markov chains we can deduce from one state all the other ones. All states are either transient, or positively recurrent, or null recurrent. If one state is periodic then they all are, and they all must have the same period length as well.

For an irreducible, aperiodic, homogeneous Markov chain there always exists a stationary limit distribution,

$$\pi_i = \lim_{t \rightarrow \infty} P(X(t) = i) \quad \forall i \in Z,$$

that is independent of the initial state of the system. If all states are positively recurrent, then the limit distribution is the unique stationary distribution. For transient and null recurrent states it holds that $\pi_i = 0$. For example, the pure birth process previously introduced is not an irreducible chain and obviously not stationary so there does not exist a limit distribution.

Time Discrete, Finite Markov Chains

Since homogeneous Markov chains $\{X_i\}$ in discrete time having finitely many states allow for a simple demonstration of some basic ideas, we will consider these first. The majority of the statements will become more complicated for chains in continuous time or with unlimited state space and because of this we will reserve our attention to these for later. If we consider only k states (available resources, faces of a coin, ...), then we can set up a $k \times k$ -transition matrix P with

$$P = (p_{ij})_{1 \leq i, j \leq k} \quad \text{and} \quad \sum_{j=1}^k p_{ij} = 1 \quad \forall i$$

which comprises the transition probabilities from state i into state j for one time step.

Starting from an initial distribution for X_0 with the state probabilities $\pi_i(t_0)$, we can now compute the distribution for the next instant of time t_1 by applying the transpose of the transition matrix to the vector $\pi(t_0)$ comprising the state probabilities $\pi_i(t_0)$. In general, it holds that,

$$P^T \pi(t_i) = \pi(t_{i+1}).$$

We can thus observe the transient phase of the system incrementally step by step. We can even state how the system will develop after n steps (compare the Chapman–Kolmogorov equation (9.4)): The matrix

$$P^n$$

contains the probabilities for a state transition from i to j after n steps. If the Markov chain is irreducible and aperiodic, the matrix product P^n then converges for $n \rightarrow \infty$ toward a matrix $\hat{P} = (\hat{p}_{ij})_{1 \leq i, j \leq k}$ with

$$\hat{p}_{ij} = \pi_j \quad \forall i, j.$$

The application of the transpose of this matrix to the start vector yields the probabilities π_i of the unique stationary limit distribution—and this being computed independent of the initial state.

Alternatively, we can also compute the stationary state directly. The vector π of the state probabilities π_i is a *fixed point*, and as such it therefore holds that

$$P^T \pi = \pi.$$

The direct solution of the *linear system of equations*

$$(P^T - I)\pi = 0$$

yields the probabilities π_i up to a constant factor which can be computed via the normalization condition (9.3).

In the example for tossing the dice (compare Fig. 9.11) it already holds that $P = P^2 = \hat{P}$. The solution of the system of equations for the fixed point π yields that all π_i have the same size, $\pi_1 = \pi_2 = \dots = \pi_6 = c$. Normalized, this yields $c = 1/6$ —the probability that one of the six sides of a dice is rolled.

For a smaller number k of states the system of equations can, e.g., be solved directly by means of Gaussian elimination. However, the storage requirements for the (typically) very sparsely populated matrix is $\mathcal{O}(k^2)$ and the time effort for the solution is $\mathcal{O}(k^3)$. For large Markov chains, a direct solution is no longer possible. Instead, numerical iterative methods such as Jacobi or Gauß–Seidel or minimization methods such as the CG-method should be applied (see Sect. 2.4.4).

9.4.2 Queuing Systems

If we consider elementary queuing systems of the form $G/G/m/\infty/\infty$, i.e., arbitrary systems with an infinite capacity with respect to the waiting line k as well as an infinite world population n , we must then require the system to be *stable*. If a waiting line can sooner or later become infinite as is possible if the orders arrive at a faster rate than they are processed, then we call the queuing system *unstable*. For stability, it must be ensured that the arrival rate λ is smaller than the service rate of all service units combined:

$$\lambda < m\mu.$$

Systems with finite waiting capacity are always stable: If all waiting slots are occupied at the arrival of an order, then it is rejected. In the case for finite capacity we speak of a *loss system* instead of a queuing system. Systems with a finite population of jobs are trivially stable as well. We have already become acquainted with an example of an unstable system: For the pure birth process, only new

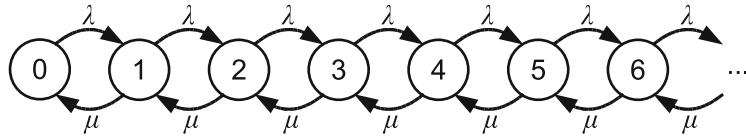


Fig. 9.14 State graph, HBDP for the $M/M/1$ queuing system; the state quantity is the number of customers

individuals arrive with birth rate $\lambda > 0$ per individual; the death rate μ is equal to zero.

The Markov property is satisfied for Poisson processes thanks to the underlying memoryless nature of the interarrival times. In the following, we will exclusively consider elementary queuing systems of the form $M/M/m$, even if corresponding statements could also be made, e.g., for $M/G/1$ - or $G/M/1$ -systems. Furthermore, we assume only homogeneous Markov chains in continuous time, and we are interested in the stationary state.

$M/M/1$

The simplest case is a queuing system with exponentially distributed arrival and service rates λ and μ , respectively, and a single service unit. For example, our post office with one counter applies here. If we consider the number of customers $X(t)$, then the Markov chain $\{X(t)\}$ is an HBDP. Its state graph appears in Fig. 9.14. For $\lambda < \mu$, the system is stable and all states are positively recurrent. As a consequence, there exists a unique stationary limit distribution.

Let us now consider the stationary state. The state probabilities π_i do not change. For the stationary case, in general, it holds that (transition rates γ_{ij} from i to j)

$$\sum_{k \in Z} \pi_k \gamma_{ki} = \sum_{j \in Z} \pi_i \gamma_{ij} \quad \forall i,$$

i.e., “whatever comes along must also leave again”. With this, we obtain an infinite system of equations. Beginning with state 0, this is

$$\pi_0 \lambda = \pi_1 \mu$$

$$\pi_1 \lambda = \pi_2 \mu$$

$$\pi_2 \lambda = \pi_3 \mu$$

...

or, in general and with respect to π_0 ,

$$\pi_i = \left(\frac{\lambda}{\mu}\right)^i \pi_0.$$

The distribution for the state probabilities is just the *geometric distribution*, the memoryless discrete analog to the exponential distribution.

From the normalization condition as well as exploiting the geometric series property to infer that the HBDP is stable, it follows that,

$$\sum_{i=0}^{\infty} \pi_i = 1 = \sum_{i=0}^{\infty} \pi_0 \left(\frac{\lambda}{\mu}\right)^i = \pi_0 \frac{1}{1 - \lambda/\mu}, \text{ d. h. } \pi_0 = 1 - \frac{\lambda}{\mu}.$$

The system is used to capacity unless it is at state 0:

$$\varrho = 1 - \pi_0 = \frac{\lambda}{\mu} = \frac{E(D)}{c}.$$

Long-term, we have that the departure rate from the stable system equals the arrival rate and it holds that,

$$E(D) = \lambda$$

(Poisson arrival process). The maximum departure rate (the maximum throughput) possible corresponds to the service rate (any more would be impossible),

$$c = \mu,$$

and the average service time is

$$E(B) = \frac{1}{\mu}.$$

The expected value of the number of customers in the stationary state is one of the quantities central to the elementary queuing system and is computed as,

$$\begin{aligned} E(F) &= \sum_{i=0}^{\infty} i \pi_i = \sum_{i=0}^{\infty} i \pi_0 \varrho^i = \pi_0 \varrho \sum_{i=0}^{\infty} i \varrho^{i-1} = \pi_0 \varrho \frac{d}{d\varrho} \sum_{i=0}^{\infty} \varrho^i = \pi_0 \varrho \frac{d}{d\varrho} \frac{1}{1-\varrho} \\ &= \frac{\varrho}{1-\varrho} = \frac{\lambda}{\mu - \lambda}. \end{aligned}$$

We can use Little's law (9.1) to determine the average time spent in the system if we know the average number of customers,

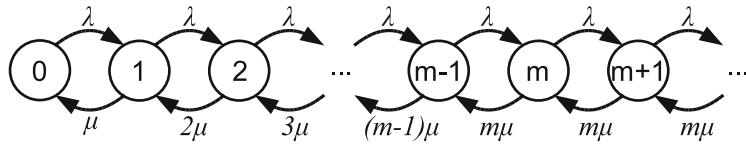


Fig. 9.15 State graph, HBDP for the $M/M/m$ queuing system; the state quantity is the number of customers

$$E(Y) = \frac{E(F)}{\lambda} = \frac{1}{\mu - \lambda}$$

and use this to compute additional quantities such as the average waiting time $E(W)$ and the average number of customers of the queue $E(F_Q)$:

$$\begin{aligned} E(W) &= E(Y) - E(B) = \frac{1}{\mu - \lambda} - \frac{1}{\mu}, \\ E(F_Q) &= \lambda E(W) = E(F) - \varrho. \end{aligned}$$

$M/M/m$ and $M/M/\infty$

For m service units it becomes slightly more complicated since up to m jobs are serviced simultaneously. The requirement for the stability of the system now becomes $\lambda < m\mu$; the utilization of the multi-server is,

$$\varrho = \frac{\lambda}{m\mu}.$$

The state graph is illustrated in Fig. 9.15. The new HBDP has been changed given the new service transition rates: If only one customer is in the post office, only one postal worker can attend to him and the other $m - 1$ are unoccupied. Full capacity can occur only when at least m customers are present.

The computation of the state probabilities becomes significantly more complicated. For the first m states we have that,

$$\pi_i = \frac{1}{i!} \left(\frac{\lambda}{\mu} \right)^i \pi_0, \quad 0 \leq i < m,$$

and for all others,

$$\pi_i = \left(\frac{\lambda}{m\mu} \right)^{i-m} \pi_m = \frac{1}{m!} \left(\frac{\lambda}{\mu} \right)^m \left(\frac{\lambda}{m\mu} \right)^{i-m} \pi_0, \quad m \leq i.$$

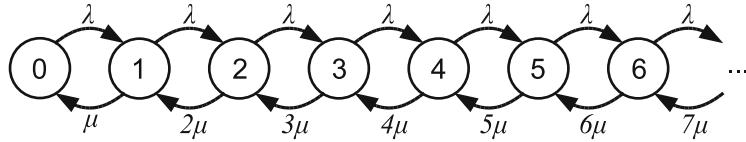


Fig. 9.16 State graph, HBDP for the $M/M/\infty$ queuing system; the state quantity is the number of customers

Results for the performance indices can be determined analogous to the $M/M/1$ system, however, their representation no longer is as compact due to their dependency on m . We therefore only state the result; the average number of customers who are currently served is $\lambda/\mu = m\varrho$.

We now also consider the other extreme, the $M/M/\infty$ system. Here, the queuing system has infinitely many service stations; it is, for example, a self-service system as illustrated in Fig. 9.16. We begin with a simplified situation in which customers help themselves to the shelves of a supermarket and the fact that only finitely many customers fit into a supermarket is ignored. Alternatively, the determination of the number of service units (e.g., telephone lines) required to minimize the customer waiting time (or the number of customers rejected) if the arrival and service rates are known can be modeled as an $M/M/\infty$ system. For stability we do not have to require any restriction since $m\mu$ will always become larger than λ for sufficiently large m .

For the stationary state it holds that,

$$\pi_i = \frac{1}{i!} \left(\frac{\lambda}{\mu} \right)^i \pi_0,$$

and with the normalization condition, we have that,

$$\sum_{i=0}^{\infty} \pi_i = 1 = \pi_0 \sum_{i=0}^{\infty} \frac{1}{i!} \left(\frac{\lambda}{\mu} \right)^i = \pi_0 e^{\lambda/\mu}, \text{ d.h. } \pi_0 = e^{-\lambda/\mu}.$$

All other quantities that are considered become simple as well:

$$E(F) = \frac{\lambda}{\mu},$$

$$E(Y) = \frac{1}{\mu},$$

$$E(W) = E(Y) - E(B) = 0.$$

9.4.3 Queuing Networks

With simple queuing networks one can still make analytical statements as well. For example, these could provide reference points for the simulation of more complicated queuing networks. The following considers the stationary state for networks with N vertices, as introduced in Sect. 9.3. Many of the previous observations continue to hold when consideration is given to the individual vertices.

Let the service times at vertex i have an exponential distribution with service rate μ_i and let the elementary queuing systems be stable. Thus, for all vertices with $m_i < \infty$ service units it must hold that,

$$\varrho_i < 1, \text{ with } \varrho_i = \frac{\lambda_i}{m_i \mu_i}.$$

Tasks that leave vertex i reach queuing system j with probability p_{ij} or, with $j = 0$, leave the system. For the long-term behavior of stable queuing systems, the departure rate is equal to the arrival rate. Given the arrival rate γ_i , at vertex i , for orders from outside the network, the overall arrival rate, λ_i , at vertex i is then computed to be,

$$\lambda_i = \gamma_i + \sum_{j=1}^N \lambda_j p_{ji}. \quad (9.5)$$

If the arrival processes from outside of the net are independent Poisson processes, then the queuing network is called an (open) *Jackson network*. In particular, it holds that every queuing system i behaves like an $M/M/m_i$ system with arrival rate λ_i and the system of equations for the arrival rates (9.5) has a unique solution (which once more we can determine by means of Gaussian elimination or iterative methods). With this we can make statements about the individual vertices in the queuing network as well as the entire net!

As a brief example consider a wholesale cheese store as depicted in Fig. 9.17. Customers arrive at the store at a rate of $\lambda = 100$ customers per hour from which $p_{02} = 10\%$ of them are served at the cheese counter (service station 2) and proceed directly to the cash register (service station 3). Another $p_{01} = 90\%$ help themselves to the cheese shelves (service station 1). On average, $p_{13} = 8/9$ of them find what they were looking for and subsequently continue directly to the cash register. Every ninth ($p_{12} = 1/9$) customer has yet to make a selection at the cheese counter.

The cheese shelf can be modeled as an $M/M/\infty$ -system. On average, a customer needs two minutes to make a selection, i.e., $\mu_1 = 30$. At the $M/M/1$ cheese counter there is a server with a service rate of $\mu_2 = 25$. There are three cash registers ($M/M/3$) each having respective service rates of $\mu_3 = 40$. It holds that $\lambda_1 = p_{01}\lambda = 90$, $\lambda_2 = p_{02}\lambda + p_{12}\lambda_1 = 10 + 10 = 20$ and, since all customers eventually have to go to a cash register, it holds then that as expected, $\lambda_3 = p_{13}\lambda_1 + \lambda_2 =$

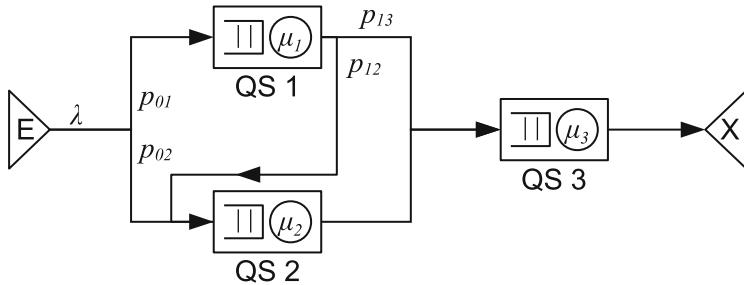


Fig. 9.17 Schematic representation of a wholesale cheese store modeled as a Jackson network. Customers choose between self-service at the cheese shelf (QS1) followed possibly by the cheese counter (QS2) with a server, or they go straight to the cheese counter to be served. Finally, everyone must go to the cash register (QS3) with 3 cashiers

$80 + 20 = 100$. We can determine the utilizations at the cheese counter and the cash register as,

$$\varrho_2 = \frac{\lambda_2}{\mu_2} = 4/5 \text{ and}$$

$$\varrho_3 = \frac{\lambda_3}{m_3 \mu_3} = 5/6, \text{ respectively.}$$

The utilization at the cash registers is higher than at the cheese counter. For a greater onrush of customers this is where to expect the traffic bottleneck. The expected value of the number of customers in the queuing network is the sum of the expected numbers of customers of the three service stations:

$$E(F) = E(F_{SU1}) + E(F_{SU2}) + E(F_{SU3}) = 3 + 4 + 535/89 \approx 13.01.$$

Now we can compute additional quantities such as the expected time for a customer to be in the system; for our purposes, the utilization and “number of customer” computations should suffice as examples.

9.4.4 Simulation

If, in addition, we want to allow tasks the utilization of several facilities simultaneously, or to wait for the release of rights, the mechanisms previously introduced are no longer sufficient. The queuing networks that result must become more complex but can, for example, be modeled by *Petri networks*. Analytical statements are, in general, no longer feasible; the system must be simulated.

Since we observe events at discrete instants in time in the queuing network, e.g., the arrival of an order at a vertex or the completion of an order, it makes little sense to discretize the time into time discs of equal size. Rather, we can simulate event-driven and speak of *discrete event simulation*. The simulation clock then makes a jump from event to event.

We need a global list that can be used to record all events. These events will be processed consecutively one after another and are able to trigger subsequent events. In the example involving the cheese store, the event list is empty at the beginning of the simulation. The interarrival time T_1 for the first customer is determined via an exponentially distributed random variable with parameter λ and entered into the list of events as event “customer arrival” with time stamp $t = T_1$. This will in turn be immediately taken out as the chronologically next event and triggers two subsequent events. Firstly, the next customer arrival at time $t = T_1 + T_2$, and secondly, in the case of a choice to visit the cheese counter, the event “order at service station 2 completed” at time $t = T_1 + B_{21}$, whereby the service time B_{21} has been determined via a random variable having an exponential distribution with parameter μ_2 . Next, the temporal event occurring subsequently is taken from the list of events.

Similar to the graph search methods of the microscopic traffic simulation reviewed in Sect. 8.4.3, one must pay special attention to ensure that suitable data structures for (event) lists are used in the realization, for example, the use of Fibonacci heaps. With these, the effort needed to manage the list of events and to search for the event with the smallest time stamp can be kept as small as possible. For further information on Fibonacci heaps or other priority queues we refer the reader to the textbook [13].

An additional item deserving our attention is the use of a suitable random number generator. Random numbers generated by computers are never really random. However, a table of true random numbers may be used that has been generated beforehand (obtained from “true” randomness, for example via the measurement of radioactive decay processes). Unfortunately, this is not feasible for comprehensive event simulations due to the large number of required random numbers, the corresponding large storage requirement, not to mention the fact that the table should not be used repeatedly.

It is for this reason that algorithms are employed that produce so-called *pseudo random numbers*. They use a deterministic rule (an algorithm), f , that takes a random number, x_i , and produces the next random number $x_{i+1} = f(x_i)$, e.g., via permutations. *Statistical tests* must confirm that the pseudo random numbers adequately simulate the true distribution (the average value, standard deviation, etc. should coincide) and that they are “independent enough”. If we can produce sufficiently good, uniformly distributed random numbers, we can then use suitable transformations to obtain random numbers with different distributions.

Throughout a simulation we can keep a record of the unknown performance indices and thus determine the expected values. Here, we must distinguish between the transient settling phase and the almost stationary phase. If we are interested in the stationary state, we should try to only use measured values after the time that the system has settled and the measured values no longer fluctuate

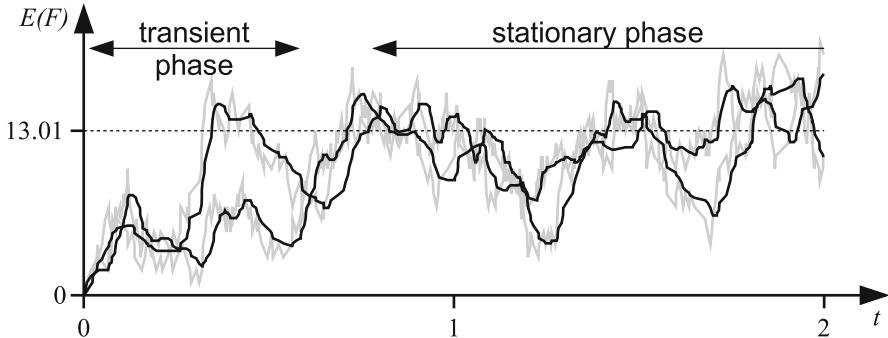


Fig. 9.18 Two paths of the number of customers of a discrete event simulation of the cheese wholesale (gray) as well as averaged over respective ten measuring points (black) for the first 2 h

that strongly. For the cheese store example it becomes clear that the distinction between the transient and stationary phase is not unique. Figure 9.18 shows two progressions (paths) of the number of customers of the system as well as their paths obtained by averaging over 10 measuring points. The system has only about 13 customers and is therefore small, the expected value of the number of customers can already be reached after a very short time. It is also seen in the later course that the number of customers can have very large fluctuations.

But how can we guarantee that the accuracy is sufficient given the number of measurements collected? If we have a sufficiently large set of n iid measured values, x_i , then the computed average value, $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$, is arbitrarily close to the actual expected value μ . Here, *statistical tests* are of help and can tell us when the previously recorded set of measured values satisfies a specified confidence level. Unfortunately, in our case the measured values obtained for a path are not at all independent. One remedy is to perform several independent simulation runs from which the average of the iid average values can be obtained; but this drives up the required computational effort. If we consider the stationary phase of our wholesale cheese scenario over a very long time period, the average number of customers, in fact, then amounts to approximately 13 customers which corresponds approximately to the analytically determined value.

9.5 Summary and Outlook

We have considered stochastic techniques for the modeling and simulation of traffic. Wait- and serve-systems can be represented with the help of queues and service units. Elementary queuing systems can be used to compose complex queuing networks. Here, the memoryless property satisfied for exponential distributions plays an important role and enables us to simulate events which may have many different, unlikely and independent causes.

In addition to statements made regarding the asymptotic behavior and the identification of traffic bottlenecks that were used to find starting points for optimization, it is the determination of the stationary state that plays an important role. This is used to make statements about the long-term behavior of the system. Analytical statements are still possible for simple queuing networks. But for the evaluation and assessment of more complicated scenarios, simulations must be relied upon.

Because of the limited space we have restricted our analysis exclusively to the $M/M/m$ systems; for these the unknown performance indices can be derived in a sufficiently compact form. This being the case, we still do not want to hold back an interesting result for the $M/G/1$ case (single server system for a Poisson arrival process and general distribution of the service times). If the first two moments $E(B)$ and $E(B^2)$ of the joint distribution of the service times are known, then the Pollaczek–Khinchin formula provides us the result,

$$E(W) = \frac{\lambda E(B^2)}{2(1 - \varrho)} = \frac{1}{2} \left(1 + \frac{\sigma^2(B)}{E^2(B)} \right) \frac{\varrho E(B)}{1 - \varrho}, \quad \varrho = \lambda E(B).$$

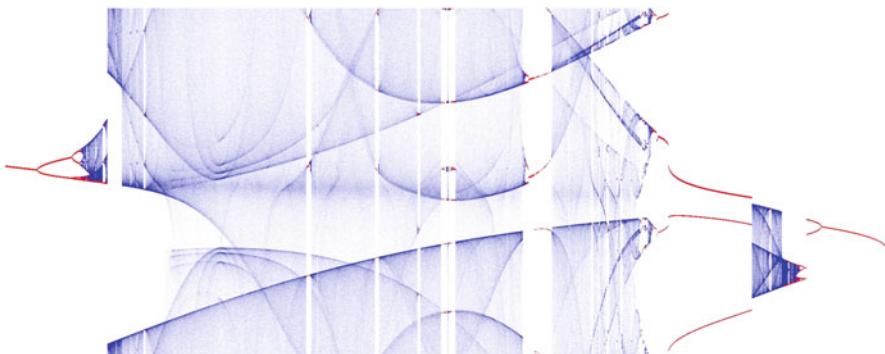
This says that the expected waiting time depends quadratically on the coefficient of variation $\sigma(B)/E(B)$! If the scattering of the service times is strong, this then has a catastrophic effect on the waiting times—and, in fact, this holds independent of the underlying distribution. Interestingly, we had observed a similar effect for the hitchhiker's paradox.

Queuing models find their applications in many areas that include, for example, the analysis of computer systems, telephone and communication networks or the simulation of production systems and production processes. And, to build the bridge toward the macro- and microscopic traffic simulation (Chaps. 7 and 8), they find applications in the simulation of road traffic: Traffic participants are orders that move through the system and occupy resources. Roads can be modeled as queuing systems with a finite capacity whose service time depends on the current number of customers (the average traffic density on the road). Correspondingly, intersections are service stations that must be occupied during a turning process. In this model as in the microscopic traffic simulation, traffic participants can pursue individual plans and intentions. Finally, driving on a road requires only one action (the determination of the service time) that enables the simulation of larger networks and facilitates the parallel and distributed computation.

There exists versatile literature about stochastic traffic modeling, even entire books on individual subparts to it. For those interested in pursuing a deeper immersion we mention, in particular, [6, 20, 54].

Part III

Dynamical Systems: Cause, Effect, and Interplay



In this part we consider models and simulations from the perspective of a *dynamical system*, i.e., we are interested in time-dependent processes that here can be described by a motion in the state space. This state space is defined by all quantities that determine the state of the system—including all information involved with the further development of the system. With respect to the simple models for the population dynamic in Chap. 10, one can think of the size of a penguin population when neglecting all external influences. By contrast, in a physical system the positions and velocities of bodies will play a role as most likely will be true for some additional parameters as well. The state of our system for a particular instant in time is now a point in the state space and, tracked over time, it results in a path (trajectory) which describes the development of the system.

Continuous models, which generally lead to differential equations, will be analyzed for the example of population dynamics (including a small excursion to include a discrete state space).

And since it is often the case that one is not content to just observe and analyze systems but as well desires to influence (control) them in a specific way, which now

results in also external influences that affect the state, we too will be concerned with this. Here, modeling will also play a role (by means of a mechanical example) since it would not work without knowledge of the system that is to be controlled.

Not everything is always as easy as it is for population dynamics. In some systems whose appearance may be interpreted as straightforward, one may observe very rare, chaotic behavior. But chaos is not necessarily the same as it is perceived in a colloquial sense. It will be explained what chaos actually is, how it manifests itself and where it can appear.

Chapter 10

Population Dynamics

The development of a population of one or more species of animals or plants is a manageable example permitting the study of system dynamics. Here, the number of individuals of the respective species define the state space—the term “space” is somewhat misleading since it is not a spatial resolution of the considered domain but instead it is assumed that the rates of increase and decrease of the population depend only on the size of the populations.

It would be straightforward to describe this state space with the set of natural numbers or (in the case of several species) by tuples of natural numbers since we are concerned with the numbers of individuals. Such a model, in which the increase and decrease are modeled by a stochastic process, will be discussed at the end of this chapter. However, these models quickly become relatively complex; one obtains models that are significantly easier to treat by describing the growth through the use of (ordinary) differential equations. This requires the population sizes to be described using (non-negative) real numbers. In the case of very small populations, this is problematic—a population of 2.5 penguins is not very meaningful—but in the case of large populations this model is justified. In this case, we can also assume that the overall rate of increase can be modeled deterministically as the sum of independent individual events since random fluctuations averaged over a large population cancel each other out.

The continuous models will lead to (ordinary) differential equations, i.e., they will employ tools from analysis (Sect. 2.2.2), but especially, they will extend the considerations from the first part of Sect. 2.4.5 (linear differential equations with constant coefficients, systems of differential equations, convergence behavior, direction fields and trajectories). For the discrete model, however, we will employ methods from stochastics (discrete distributions, conditional probabilities) (Sect. 2.3).

10.1 Malthusian Growth Model

The simplest model for population dynamics goes back to the British economist Thomas Robert Malthus, who, in 1798, considered in his “Essay on the Principle of Population” a single species model with constant birth and death rates.

For the differential equation formulation let $p(t)$ be the population at time t ($t, p(t) \in [0, \infty]$). The *birth rate* $\gamma > 0$ specifies the number of births per individual and per time unit, and correspondingly, the *death rate* $\delta > 0$ specifies the number of deaths per individual and per time unit. We assume that the population of a closed region is analysed in such a way so that births and deaths are the only sources for changes in the population size. The net change in population is the *growth rate*

$$\lambda = \gamma - \delta.$$

It is essential for this model that the rates γ, δ and thus λ are constants that in particular are independent of the population size p .

The change in population per time unit now becomes the product of the growth rate and the population. We obtain the linear differential equation (see Sect. 2.4.5)

$$\dot{p}(t) = \lambda p(t)$$

having the solution,

$$p(t) = p_0 e^{\lambda t},$$

in which p_0 describes the size of the initial population at time $t = 0$.

What knowledge do we derive from this model? The realization that undamped growth with a constant growth rate proceeds exponentially, i.e., in the case of a positive growth rate (birth rate greater than death rate) there is exponential increase, is indeed not too profound, but it can be an essential factor in the prediction of the population. For smaller population sizes, the observed course rather resembles linear growth; with increasing population, the growth picks up more and more speed and results in a “population explosion”, and in its aftermath the resources no longer provide ample supplies: The knowledge about the threat of exponential growth leads to an entirely different prognosis than the assumption of linear growth. By the way, in his investigations on population growth, Malthus assumes only linear growth for the production of food and therefore concludes that undamped population growth must lead to a supply crisis.

10.2 Refined Single Species Models

For large populations, models that describe a damped increase have been analysed in the middle of the nineteenth century by the Belgian mathematician Pierre-François Verhulst. Two of his models will be considered in the following.

10.2.1 Linear Model with Saturation

To allow for the growth of p to decrease for large p while continuing to pursue a linear model we have that

$$\dot{p}(t) = \lambda_0 - \lambda_1 p(t)$$

with $\lambda_0 > 0$ (for small populations growth prevails) and $\lambda_1 > 0$ (the larger the population, the smaller the growth). For

$$p = \bar{p} := \lambda_0 / \lambda_1,$$

it holds that $\dot{p} = 0$: \bar{p} is an *equilibrium point*. For $p < \bar{p}$ it holds that $\dot{p} > 0$ while for $p > \bar{p}$ it holds that $\dot{p} < 0$. Depending on the initial value the solutions converge toward \bar{p} from below ($p_0 < \bar{p}$) or from above ($p_0 > \bar{p}$) so that \bar{p} is a *stable equilibrium point*. The solution is still easily represented in closed form:

$$p(t) = \bar{p} + (p_0 - \bar{p})e^{-\lambda_1 t}.$$

Figure 2.1 in Sect. 2.4.5 (page 70) displays solutions to this equation. Here, $\lambda_0 = \lambda_1 = 1/10$ and thus $\bar{p} = 1$ which is unrealistic if one assumes the unit “individuals”, the qualitative course of the solution curves is nonetheless representable for different scalings.

This model now describes the approximation of the population to a stable equilibrium point in which the number of births and deaths are balanced. However, the extent in which the assumed linear behavior (\dot{p} grows proportionally with distance $\bar{p} - p$ from the equilibrium point) is realistic cannot be answered from within the model. Regardless, it is a detractor that for this model small populations do not behave as expected, that is, similar to the Malthusian model—as long as the equilibrium point is far away, exponential growth should occur; the current model, however, displays a decrease in growth at the beginning. It is for obvious reasons impossible for a linear model to exhibit moderate growth for very small populations as well as when the population is in proximity to the equilibrium point yet exhibit very large growth for populations in-between. This motivates the following non-linear model.

10.2.2 Logistic Growth

The easiest way to produce growth with the desired characterization (for small populations similar to the exponential growth, for large populations transition to a saturation), is to adjust the right hand side of the differential equation to the next higher polynomial degree, i.e., a quadratic in p .

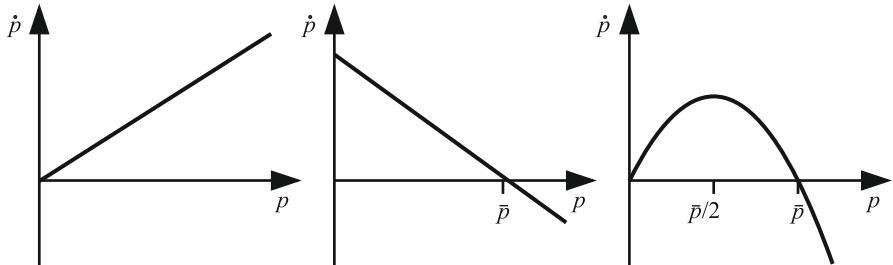


Fig. 10.1 Right hand sides of the differential equations for the three single species models: to the left exponential growth (Malthus), in the middle the linear model with saturation and to the right the quadratic model (logistic growth)

For this, we choose the growth rate $\lambda(p)$ (computed per individual and per time unit) linear in p :

$$\lambda(p) := (a - b \cdot p)$$

with parameters $a \gg b > 0$ (the justification why a should be large relative to b will follow) and obtain (after multiplication with the population size) the *logistic differential equation*

$$\dot{p}(t) = \lambda(p(t)) \cdot p(t) = ap(t) - bp(t)^2, \quad (10.1)$$

where we again can specify the initial population $p(0) = p_0$.

Figure 10.1 shows the right hand sides of the three previously treated population models in comparison. One can already expect from this sketch that the quadratic model (right) behaves similar the Malthusian model (left) for small p but similar to the saturation model for larger p .

More precisely, we see that the new model has two equilibrium points:

- $p = 0$ leads to $\dot{p} = 0$, independent of a and b . If one linearizes the right hand side $ap - bp^2$ for $p = 0$, then one obtains $\dot{p} \doteq ap$ for small p . In view of $a > 0$, the linearized differential equation has an unstable equilibrium point at $p = 0$: For arbitrarily small perturbations $p_0 > 0$ the exponential growth will dominate. For sufficiently small p this property carries over to the original equation with a quadratic right hand side.
- $p = \bar{p} := a/b$ leads to $\dot{p} = 0$ as well. Linearization at $p = a/b$ however yields $\dot{p} \doteq -a(p - \bar{p})$ for $p \approx \bar{p}$. In view of $-a < 0$, we have an attracting equilibrium of the linearized differential equation which again carries over to the original equation if p lies sufficiently close to \bar{p} .

The above considerations give us a qualitative description of the solution curves:

- For $0 < p_0 < \bar{p}/2$, the population grows monotonically. The derivative \dot{p} increases until an inflection point is reached at $p = \bar{p}/2$; there, $\ddot{p} = 0$ and is \dot{p} maximal. The growth then slows and p converges toward \bar{p} .

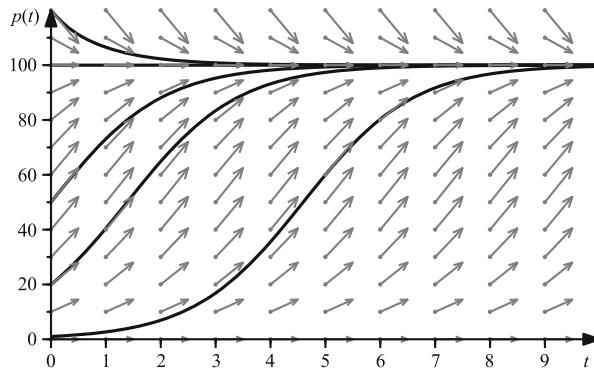


Fig. 10.2 Solutions and direction fields of the differential equation (10.1); $a = 1$, $b = 0.01$, $\bar{p} = a/b = 100$. Initial values $p_0 = 1, 20, 50, 100$ and 120 (direction vectors are represented at half of their lengths)

- For $\bar{p}/2 \leq p_0 < \bar{p}$ the solution behaves almost the same, only the phase of accelerated growth is omitted.
- For $p_0 = \bar{p}$, p remains at the equilibrium point (and would also not be permanently diverted due to small disruptions).
- For $p_0 > \bar{p}$, it holds that $\dot{p} < 0$: The population decreases monotonically and converges toward \bar{p} .

To summarize, it holds that the solution converges toward $\bar{p} = a/b$ for arbitrary initial values $p_0 > 0$, thus justifying the requirement that $a \gg b$. Figure 10.2 shows the direction field and solution curves for $a = 1$, $b = 0.01$ and for different initial values p_0 .

Incidentally, for the logistic differential equation the solution can be given in a closed-form, that is

$$p(t) = \frac{a \cdot p_0}{b \cdot p_0 + (a - b \cdot p_0) \cdot e^{-at}}.$$

A similar model will again appear in the chapter on chaos theory (Sect. 12.2.1), but there it will involve discrete time steps and work under the assumption that $p_{n+1} = r \cdot p_n \cdot (1 - p_n)$ with parameter $r > 0$, so that (up to a constant factor) the right hand side of the logistic differential equation (10.1) is used as the iteration function. Hence, it no longer describes the change in the population, but the entire population size at the next time step.

10.3 Two Species Models

More interesting than single populations are models with several species that interact with each other so that additional effects are possible. For the sake of clarity we will restrict our attention to two species—for the linear case see Sect. 2.4.5.

We introduce a parametrized model from which we can study two scenarios of interaction: first, the fight of two species for the same resources, and second, a predator-prey relationship.

In the following we consider two species P and Q , the functions $p(t)$ and $q(t)$ denote their respective population sizes. The first assumption is for the growth rates to be known, as with the model by Malthus and for logistic growth, and given per individual and per time unit, $f(p, q)$ for P and $g(p, q)$ for Q , where now they depend both on p as well as on q ; let the vector F describe the growth (i.e., growth rate times the population):

$$\begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} = \begin{pmatrix} f(p, q) \cdot p \\ g(p, q) \cdot q \end{pmatrix} =: F(p, q). \quad (10.2)$$

Since we are only interested in the case in which $p > 0$ and $q > 0$ (otherwise it would once again be less than two populations), the equilibrium points of the system are characterized by the fact that both growth rates disappear: We are looking for pairs (\bar{p}, \bar{q}) with $\bar{p}, \bar{q} > 0$ such that

$$f(\bar{p}, \bar{q}) = g(\bar{p}, \bar{q}) = 0.$$

To analyse the stability of an equilibrium point analogously to the considerations given in Sect. 10.2.2, in the following we replace the differential equation by one that is linear. This is accomplished by linearizing the right hand side F at the equilibrium point (\bar{p}, \bar{q}) :

$$F(p, q) \doteq J_F(\bar{p}, \bar{q}) \begin{pmatrix} p - \bar{p} \\ q - \bar{q} \end{pmatrix}.$$

For the Jacobi matrix $J_F(\bar{p}, \bar{q})$, given the structure of F and in view of $f(\bar{p}, \bar{q}) = g(\bar{p}, \bar{q}) = 0$, it holds that,

$$\begin{aligned} J_F(\bar{p}, \bar{q}) &= \left(\begin{array}{cc} \frac{\partial(f(p, q) \cdot p)}{\partial p} & \frac{\partial(f(p, q) \cdot p)}{\partial q} \\ \frac{\partial(g(p, q) \cdot q)}{\partial p} & \frac{\partial(g(p, q) \cdot q)}{\partial q} \end{array} \right) \Big|_{p=\bar{p}, q=\bar{q}} \\ &= \left(\begin{array}{cc} f_p(p, q)p + f(p, q) & f_q(p, q)p \\ g_p(p, q)q & g_q(p, q)q + g(p, q) \end{array} \right) \Big|_{p=\bar{p}, q=\bar{q}} \\ &= \begin{pmatrix} f_p(\bar{p}, \bar{q})\bar{p} & f_q(\bar{p}, \bar{q})\bar{p} \\ g_p(\bar{p}, \bar{q})\bar{q} & g_q(\bar{p}, \bar{q})\bar{q} \end{pmatrix}, \end{aligned}$$

where $f_p(p, q) := \partial f(p, q)/\partial p$ etc. denotes the partial derivative. For a sufficiently small neighborhood of (\bar{p}, \bar{q}) the linear differential equation,

$$\begin{pmatrix} \dot{p} \\ \dot{q} \end{pmatrix} = J_F(\bar{p}, \bar{q}) \begin{pmatrix} p \\ q \end{pmatrix} - J_F(\bar{p}, \bar{q}) \begin{pmatrix} \bar{p} \\ \bar{q} \end{pmatrix} \quad (10.3)$$

behaves as the original equation (10.2); in particular, the eigenvalues of $J_F(\bar{p}, \bar{q})$ determine the degree of attraction to the equilibrium point.

For the next step in the model, the growth rates f and g must be defined. Here we choose a linear approach (dividing by the components of the equilibrium saves us from having to include them as additional constants in subsequent considerations):

$$\begin{aligned} f(p, q) &= (a_1 - b_1 p - c_1 q)/\bar{p}, \\ g(p, q) &= (a_2 - c_2 p - b_2 q)/\bar{q}, \end{aligned}$$

in which the limiting case leads to the disappearance of a population and as a consequence this in turn leads to a quadratic model for the remaining population that is analogous to logistic growth. For the Jacobi matrix $J_F(\bar{p}, \bar{q})$ at an equilibrium point, it then holds that

$$J_F(\bar{p}, \bar{q}) = \begin{pmatrix} f_p(\bar{p}, \bar{q})\bar{p} & f_q(\bar{p}, \bar{q})\bar{p} \\ g_p(\bar{p}, \bar{q})\bar{q} & g_q(\bar{p}, \bar{q})\bar{q} \end{pmatrix} = \begin{pmatrix} -b_1 & -c_1 \\ -c_2 & -b_2 \end{pmatrix}.$$

The parameters still need to be determined: To this end, the following conditions shall be satisfied:

- The terms b_i that describe the reduced growth for large populations should be non-negative:

$$b_1, b_2 \geq 0. \quad (10.4)$$

- A somewhat artificial condition which will be beneficial for the subsequent computations asks for the growth rate of at least one species (here without loss of generality for Q) to be reduced due to a large population reached by the other species (in our case P):

$$c_2 > 0. \quad (10.5)$$

In order for the growth rate $g(p, q)$ of Q to even be positive (which should obviously be the case for a reasonable model), this implies the next requirement for a positive constant term

$$a_2 > 0. \quad (10.6)$$

- The eigenvalues of the Jacobi matrix $J_F(\bar{p}, \bar{q})$ are the roots $\lambda_{1/2}$ of $(\lambda + b_1)(\lambda + b_2) - c_1 c_2$. We require

$$b_1 b_2 > c_1 c_2, \quad (10.7)$$

since this then prevents the existence of any roots with positive real part in view of (10.4). (The best way to see this is to think of the parabola for $(\lambda + b_1)(\lambda + b_2) - c_1 c_2 = \lambda(\lambda + b_1 + b_2) + b_1 b_2 - c_1 c_2$: For $b_1 b_2 - c_1 c_2 = 0$, the roots

are $\lambda_1 = 0$ and $\lambda_2 = -b_1 - b_2 \leq 0$, while increasing $b_1 b_2 - c_1 c_2$ moves up the parabola so that it has two negative roots until it no longer intersects with the abscissa; we then will have two complex conjugate solutions each having a negative real part.) If there exists an equilibrium point under these conditions, it is consequently stable.

- Still lacking is the criterion for the existence of (positive) equilibrium points. The linear system of equations

$$\begin{aligned} 0 &= a_1 - b_1 \bar{p} - c_1 \bar{q} \\ 0 &= a_2 - b_2 \bar{q} - c_2 \bar{p} \end{aligned}$$

has, given (10.7), a unique solution

$$\bar{p} = \frac{a_1 b_2 - c_1 a_2}{b_1 b_2 - c_1 c_2}, \quad \bar{q} = \frac{b_1 a_2 - a_1 c_2}{b_1 b_2 - c_1 c_2},$$

which is an equilibrium point if the numerator (and thus \bar{p}, \bar{q}) are positive. With (10.4)–(10.6), this can be stated as

$$\frac{b_1}{c_2} > \frac{a_1}{a_2} > \frac{c_1}{b_2}, \quad (10.8)$$

where it can be seen that in the case $b_2 = 0$ the second inequality has to be replaced by the restriction $c_1 < 0$.

Conditions (10.4)–(10.8) guarantee the existence of a stable equilibrium point. In the following we will analyze two models that differ with respect to the sign of the parameter c_1 . We will consider two species that are in mutual *competition* for the same resources: A large population for Q also reduces the growth rate of P ; in addition to (10.5), there then exists the condition

$$c_1 > 0. \quad (10.9)$$

With the second example, a large population for Q causes an increase in the growth rate of P —we think of P as a *predator* which thrives in relation to the amount of *prey* Q it finds. This condition manifests itself through the condition,

$$c_1 < 0. \quad (10.10)$$

For the competition example, we choose as parameters,

$$a_1 = \frac{300+5\sqrt{3}}{3}, b_1 = \frac{5}{2}, c_1 = \frac{\sqrt{3}}{6}, a_2 = \frac{35+60\sqrt{3}}{4}, b_2 = \frac{7}{8}, c_2 = \frac{3\sqrt{3}}{8}. \quad (10.11)$$

This results in an equilibrium point at $(\bar{p}, \bar{q}) = (40, 10)$. The eigenvalues of the Jacobi matrix $J_F(\bar{p}, \bar{q})$ are $-1/10$ and $-1/20$.

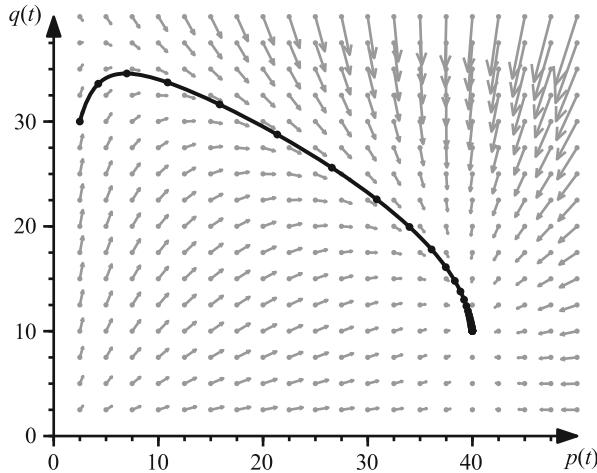


Fig. 10.3 Solution given the initial value $(2.5, 30)$ and direction field of the two species model with parameters given in (10.11). (Markers *filled circle* with distance $\delta t = 1/4$, direction vectors represented by $1/20$ of their lengths)

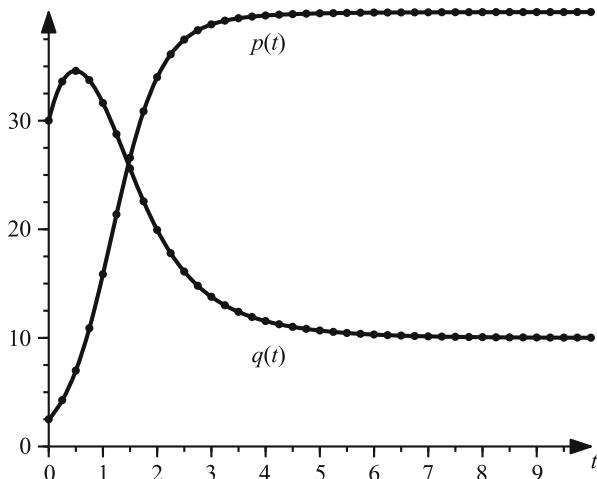
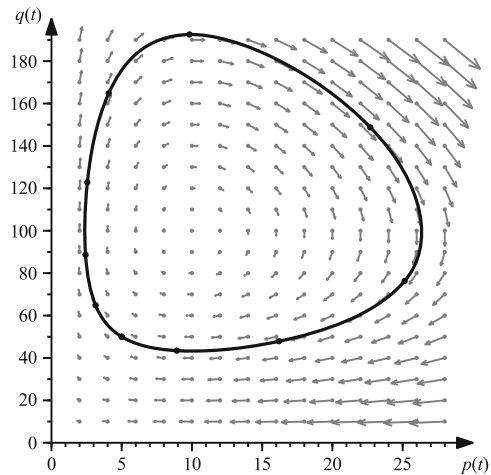


Fig. 10.4 Solution components for Fig. 10.3

Figure 10.3 shows the direction field and solution given the initial value $(p_0, q_0) = (2.5, 30)$. Figure 10.4 graphs the individual components $p(t)$ and $q(t)$. One sees that near the equilibrium point the solution behaves very similar to the linear case (see Sect. 2.4.5, and in particular Fig. 2.2 on page 72), while different effects develop when further away.

For the predator–prey model ($c_1 < 0$), the situation is simplified by neglecting the impediment brought by a large population of one's own species. This is

Fig. 10.5 Solution given the initial value $(5, 50)$ with the direction field of the two species model with parameters given by (10.12). (direction vectors are represented by 1/5 of their lengths)



accomplished by choosing $b_1 = b_2 = 0$. In return, the constant term a_1 for the growth rate of the predator P is chosen to be negative: If no prey is present the population of the predator then shrinks. The sign conditions $\bar{p} = a_2/c_2 > 0$, $\bar{q} = a_1/c_1 > 0$ are then automatically satisfied.

The eigenvalues of $J_F(\bar{p}, \bar{q})$ are purely imaginary (since $b_i = 0$): Even though the equilibrium point is stable, it is not attracting. Instead of convergence toward the equilibrium point, there evolves a periodic oscillation.

If we choose, for example,

$$a_1 = -5, \quad c_1 = -\frac{1}{20}, \quad a_2 = 20, \quad c_2 = 2, \quad (10.12)$$

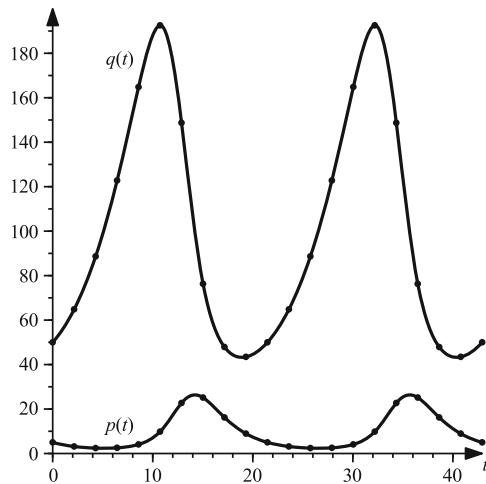
we then obtain an equilibrium point at $(\bar{p}, \bar{q}) = (10, 100)$, and the resulting eigenvalues of the Jacobi matrix $J_F(\bar{p}, \bar{q})$ are $\pm i/100$.

Figure 10.5 shows the solution given the initial value $(p_0, q_0) = (5, 50)$ while Fig. 10.6 shows the individual components $p(t)$ and $q(t)$. One clearly sees how a large predator population p leads to a collapse of the prey population q , which then—time-delayed—also leads to a decline of the predators. As a result, the population of the prey recovers in turn and the cycle begins anew.

10.4 A Discrete Single Species Model

To bring the models for population dynamics to a close, we now consider briefly a discrete model. In reality the population is a discrete quantity so that this approach initially appears to be obvious; but it turns out that the discrete models

Fig. 10.6 Solution components for Fig. 10.5



are significantly more difficult to handle than the continuous ones (which is the reason why only a very simple case shall be introduced here).

First, the state space is a discrete quantity $X(t) \in \mathbb{N}$ that describes the number of individuals at time t . The problem now is that we cannot update the population with the given population, birth and death rates as for the continuous case: births and deaths can only produce integer changes in the population and therefore must now be modeled as random events. We thus have a *stochastic process* that is continuous in time but discrete in the state space (see Sect. 9.4).

In reality, we are no longer determining the solution $X(t)$ but instead the distribution of X that specifies the probability $P(X(t) = x)$ that there will be exactly x individuals at time t . The temporal development of this distribution is to be determined next.

In the following, we will use

$$\pi_x(t) := P(X(t) = x)$$

to abbreviate the unknown probabilities; in order to keep things simple, we assume the birth rate γ to be constant and the death rate δ to be zero. (This is in contrast to the continuous model in which here both quantities will be important and not just their difference; in our example we have a pure birth process, see Fig. 9.13).

If we know the distribution of $\pi_x(t)$, $x \in \mathbb{N}$ we can then use it later to draw a conclusion about the distribution over a small time interval δt : The probability that there is a birth under the condition that $X(t) = x$ is, in the limiting case $\delta t \rightarrow 0$, the product of the birth rate, the population and the time interval, i.e., $\gamma x \delta t$. The absolute probability that the state $X(t) = x$ transitions into the state $X(t + \delta t) = x + 1$ is in the limiting case $\delta t \rightarrow 0$ then given by the computational rules for conditional probabilities, i.e., $\gamma x \delta t \pi_x(t)$.

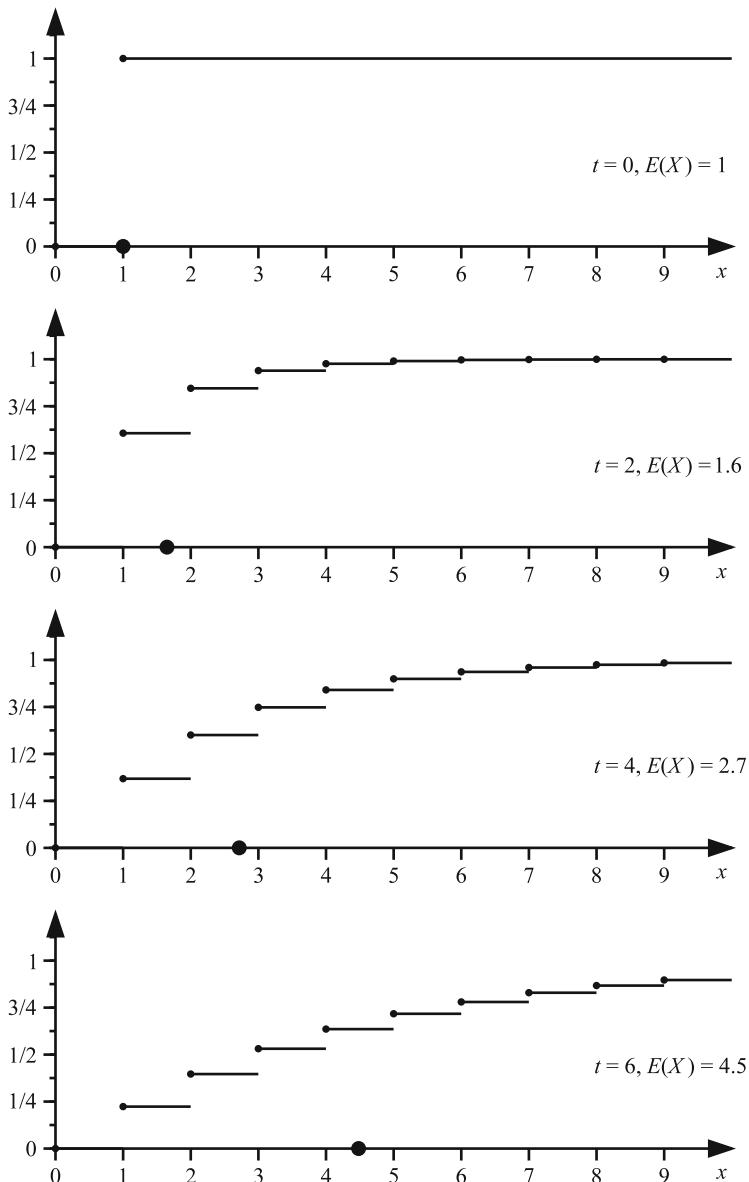


Fig. 10.7 Temporal development of the distribution function and the expected value (Markers filled circle on the abscissa) in the discrete population model ($\gamma = 1/4$)

The probability to find a population of size x at time $t + \delta t$ satisfies for $\delta t \rightarrow 0$,

$$\begin{aligned}\pi_x(t + \delta t) &\doteq \pi_x(t) \\ &\quad - \underbrace{\gamma x \delta t \pi_x(t)}_{\text{births given the state } X = x} \\ &\quad + \underbrace{\gamma(x-1) \delta t \pi_{x-1}(t)}_{\text{births given the state } X = x-1}\end{aligned}$$

(where we set $\pi_0(t) \equiv 0$) and hence,

$$\dot{\pi}_x(t) = -\gamma x \pi_x(t) + \gamma(x-1) \pi_{x-1}(t).$$

We have an infinite system of differential equations which describes the development of the distribution.

For the initial values we assume—after putting a desire for biological exactness into the back seat—that the initial population for $t = 0$ consists of a single individual:

$$\pi_1(0) = 1 \text{ and } \pi_x(0) = 0 \text{ for } x > 1.$$

The solution of the system then can be given in closed form,

$$\pi_x(t) = e^{-\gamma t} (1 - e^{-\gamma t})^{x-1}.$$

Figure 10.7 shows the temporal development of the distribution function $P(X(t) \leq x)$ for the birth rate $\gamma = 1/4$ and for the times $t = 0$ (initial population), $t = 2$, $t = 4$ and $t = 6$.

It is also interesting to compute the expected value $E(X(t))$ of the population at time t , that is

$$E(X(t)) = e^{\gamma t}.$$

This establishes a relationship with the (continuous) model of Malthus, that in this case perfectly mirrors the expected value of the discrete model.

Chapter 11

Control Engineering

The goal of *control engineering* is to apply external influence to a *dynamical system* so that it behaves in a desired way. Here, the behavior is under constant monitoring through a comparison between the *desired* and the *actual measured values* of the system. When these deviate, the goal is to manipulate the system in a way to minimize future deviations. In view of this *feedback* one speaks of *closed-loop control* or *feedback control*. If there is no feedback, i.e., if the manipulation happens without knowledge on the actual state of the system, then one does not speak of a closed-loop controller but instead of an open-loop or non-feedback control.

In everyday life, the human himself very often acts as a controller, e.g., in the shower when adjusting the water temperature. The desired state is a comfortable water temperature while unfortunately the actual state strongly deviates much too often. We keep changing the setting of the faucet until the deviation is within our threshhold of acceptability.

There appear controls in nature as well. In our body there exist a multitude of control loops which keep quantities such as the body temperature or the blood pressure at certain values, for example.

Nowadays, control engineering finds its predominant application in technical systems. Well-known application areas include air conditioning systems or systems that enhance driving safety such as ABS and ESC. But control engineering also finds applications in non-technical systems, for example in economics or in the analysis of biological controls. In any case, the modeling of the system that is to be controlled is always a part of control engineering. In Sect. 11.2, we will provide examples whose purpose is to provide details of such modeling by means of a mechanical system.

There exist many different approaches that can be tapped to reach a given control objective. Methods in the area of “classical control theory”, which we will briefly cover in Sect. 11.1.4, require that a controller is developed in such a way that the closed control loop displays certain mathematically definable properties. One advantage of this procedure is that one can easily precompute the reaction of this control loop to certain inputs which therefore allows the ability to satisfy quite diverse quality criteria through a suitable design of the controller.

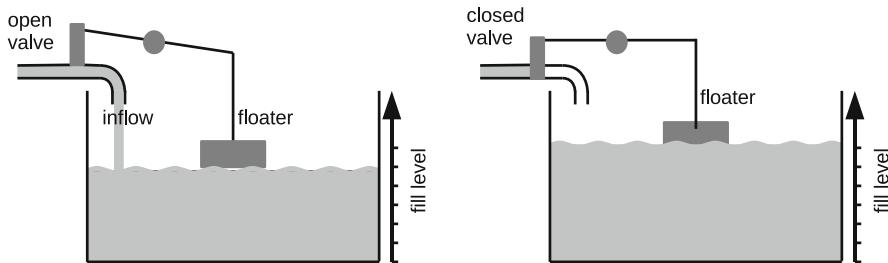


Fig. 11.1 Simple mechanical controller to control inflow

A problem with classical control theory, however, is the fact that a lot of information about the controlled system has to be known in order to set up a reasonably reliable model of the system. Furthermore, the subsequent design of a controller is often very complicated. This effort may not be necessary for all cases. In *fuzzy control*, which accounts for the main part of this chapter, the design of a controller attempts to follow the ideal of a human “controller”. For example, if a controller is to be designed so that a vehicle keeps a safe distance from the preceding vehicle, then this would be quite complex using classical control theory. The human method that heavily breaks when the distance is excessively small or to accelerate when the distance slowly increases, also usually works quite well. Fuzzy control attempts to copy just this behavior.

For the basics required for the comprehension of this chapter we require Sect. 2.1 from the basic tools and the sections on the analysis and the numerics of ordinary differential equations. Furthermore, a rudimentary basic knowledge of physics is helpful.

11.1 The Basics of Control Theory

In this chapter, we will make frequent use of the term “system”. In the context of control theory, this refers to a delimitable unit with an arbitrary number of input and output channels. This spongy definition gives a sense that everything may somehow be viewed as a system, and this is precisely what control theory is about—namely, to delimit a part of reality that shall be controlled, to determine what information is available about this part (*outputs*) as well as the quantities that can be influenced (*inputs*). For example, the device to be controlled as well as the controller itself are systems. Nowadays, many controllers, e.g., driver assistance systems such as ABS in the car, are electronic. But there already existed purely mechanical controls a lot earlier that nonetheless still have an application today.

An illustrative example of a mechanical controller is, for example, given by an inflow control that is used in the fill tanks of toilets. In Fig. 11.1, the schematic for such a control is represented. In the left part of the figure, the desired fill level has

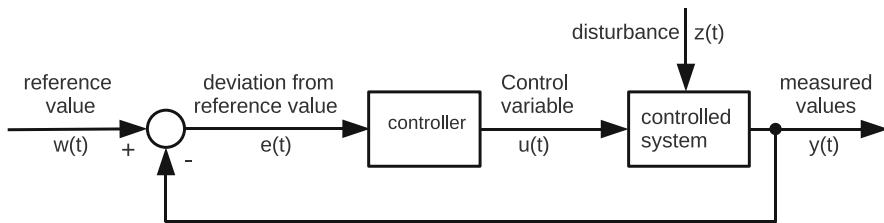


Fig. 11.2 Structure of a simple control loop

not yet been reached so the water continues to flow via the inlet into the tank. The rising water level causes the float to rise up. This in turn causes a lever to close the valve at the inlet. In the right figure, the maximum fill level has been reached and the valve is closed completely. While this example is quite elementary, it is nonetheless comparable to more complex control systems with respect to its basic mode of operation.

11.1.1 Control Loop

Regardless whether a classical controller or a fuzzy controller is used, the principle of operation as illustrated in Fig. 11.2, is mostly the same. The blocks in the figure represent individual dynamical systems, in this case the *controlled system* and the controller itself. The arrows are *signals* that influence the system. For the above inflow control, the controlled system corresponds to the fill tank which has the incoming amount of water as an *input signal* and the measurement of the fill level as an *output signal*. Indirectly, this level of fill is in turn an input quantity for the controller, in this case the system consisting of a float, a lever, and a valve. This feedback is indispensable since it closes the control loop and makes the difference between a closed-loop and open-loop control. Through this feedback, the controller realizes how the control system performs, and thereby how the system is reacting to the current controller settings. The controller, in turn, can react correspondingly, i.e., in a controlling way. This happens via the output signal of the controller. In the case of the above inflow control, this is the valve position or the amount of inflow, respectively.

Essential for the design of the controller is the knowledge about the controlled system. The more one knows about the controlled system, the easier it is in general to reach the required control objective. In the worst case, the system is a black box for which nothing is known about the system besides the measurements of the output quantity. Through measured values $y(t)$ (and only through these), conclusions can be drawn regarding the state of the system. In order to recognize whether the system is performing as desired, one first has to know the measured values if it were to be performing in an optimal way. From here on we will call these “desired measured

values” the *reference values* $w(t)$, and the difference between the reference values and the actual measured values is called the *deviation from the desired reference value* $e(t)$. It is the task of the controller—which is typically also influenced by a disturbance $z(t)$ —to influence the controlled system with the help of the control variable $u(t)$ in such a way that the deviation from the desired reference value moves toward zero. The closed control loop of the figure can in turn be interpreted as a system that has the reference value $w(t)$ and the disturbance $z(t)$ as input signals and then puts out the measured value $y(t)$.

11.1.2 Description of Linear Dynamical Systems

The control loop in Fig. 11.2 is a dynamical system which itself is composed of two other dynamical systems, namely the controller and the controlled system. There exist different possibilities to describe dynamical systems. In the following we assume that a differential equation is available that describes the entire behavior of the system. How one can obtain such a differential equation will be demonstrated by example in Sect. 11.2. Beginning with a differential equation of higher order, a system of differential equations of first order can be derived. This leads to a *state space model* (see Sect. 11.2.1). The control system, the controller, and the closed control loop can be represented by a state space model.

An alternative representation for the description of dynamical systems is obtained through the Laplace transform of the differential equation that transforms the time domain into the frequency domain. In doing so one obtains an algebraic equation instead of the differential equation. For many of the controller design methods, this representation is more illustrative when compared to the one in the time domain. However, in this book we refrain from going too deeply into control engineering, thus in the following we will only use the time domain representation. The basic considerations nonetheless hold for both representation forms.

In Sect. 11.4 we will be acquainted with a completely different means for the description of dynamical systems that, in general, is neither linear nor suitable for the exact mathematical representation of real systems. These rule-based *fuzzy systems* provide the foundation for the design of *fuzzy controllers*.

11.1.3 Requirements for the Controller

The requirements that are imposed on the controller can be very versatile depending on the application. Some of the requirements overlap while others may contradict each other. We will briefly discuss here some of the important *quality criteria*.

Reference Value Tracking

In general, a controller has the objective to influence a system in such a way that it behaves in a desirable way. As indicated in Fig. 11.2, this happens by specifying a reference value that the system has to obey. An obvious requirement for the controller is to reduce the *deviation from the reference values* to zero.

Transient Behavior

The *transient behavior* describes how the controller reacts to a change in the reference value. Normally, it is impossible to respond to the reference value immediately without any time-delay. One can design a controller so that the reference value deviation decreases very quickly, but this frequently leads to overshooting. In some applications such an overshooting must be avoided. A controller that should keep a vehicle inside the traffic lane obviously must not overshoot when it drifts off the lane with the consequence of putting the vehicle directly into the lane of oncoming traffic. Depending on the application, the desired transient behavior is different and the controller must account for it accordingly.

Compensation of Disturbance

As illustrated in Fig. 11.2, the controlled system is influenced by *disturbances*. Disturbances may also appear at other places in the control loop, for example, where there are measurements or where a reference value is specified. Naturally, these disturbances should not have any influence on the controlled system. This requirement, however, is in contrast to the required reference value sequence. After all, the controller cannot recognize whether the reference value deviation that it receives as input is an actual deviation or is caused from disturbance. A compromise must therefore be found that takes both requirements into account.

Robustness

As we will see later, the classical controllers are in special need of a model for the controlled system. This model is only an approximation of the actual controlled system. The controller that has been designed based on this model must still satisfy the requirements imposed on the real system. This property is called *robustness*.

11.1.4 PID Controller

We have already indicated that in classical control engineering, the modeling of the controlled system and the subsequent design of the controller are very complex.

For this reason, we will only touch briefly in this book on the classical control engineering and reference to [44] for a detailed exposition of the mathematical basics. As illustrated in Fig. 11.2, the controller receives the reference value deviation as an input signal and outputs what will be the system input. The controller can thus be described as a *transfer element* in that it transfers an input signal to an output signal. Internally, the controller can typically be represented as a collection of different transfer elements. Three types of transfer elements most frequently occur, namely the *proportional-* (P), the *integral-* (I) and the *derivative element* (D). Consequently, controllers built from these components are also called *PID controllers*.

Proportional Element

For a proportional transfer element, the output signal is proportional to the input signal, i.e., the input signal is multiplied by a constant factor K_P . At time t_i , it holds for the output that

$$u(t_i) = K_P \cdot e(t_i).$$

The input signal is the deviation of the reference value from the actual value. The larger this deviation, the greater the corrective action the controller has to take. Hence, the task of the proportional element is to work against the deviation from the reference value. Some very simple problems can be controlled by a simple proportional element. The inflow control in Fig. 11.1 is such an example.

Integral Element

For an integral element, the output quantity is proportional to the integral of the input quantity. The longer a reference value deviation persists, the larger becomes the integral and, consequently, this component will have a stronger influence on the controlled system. This avoids a permanent reference value deviation. If we use the sum of the discretized reference value deviations instead of the integral, we then obtain the following formula for the transfer of the input signal:

$$u(t_i) = K_I \cdot (t_i - t_{i-1}) \cdot \sum_j e(t_j).$$

In many real systems, the P-element cannot prevent a reference value deviation since its response to a reference value deviation is always proportional with a constant proportionality factor. This factor had been previously determined during the design of the controller. But the real system typically reacts differently than the model used for the design, or some other unforeseen disturbances appear. The I-element ensures that even a small reference value deviation will eventually lead to a change in the system input.

Derivative Element

The derivative element reacts proportionally to changes in the input quantity. Therefore, the magnitude of the reference value deviation is irrelevant for the D-element, it only depends on the increase or decrease of the deviation. The output is

$$u(t_i) = K_D \cdot \frac{e(t_i) - e(t_{i-1})}{t_i - t_{i-1}}.$$

If the controller does not react to a strong increase in the deviation and the deviation continues, the system will then quickly leave its equilibrium state. Only then will the P- and I-portions react to it. If for example the temperature of a room drops, then the temperature control of the radiator should not wait until it becomes too cold before starting its heater but rather as soon as it becomes obvious that it is getting colder. This is exactly what will be accomplished with the use of the derivative element.

Determination of the Parameters

The precise specification of the aforementioned elements requires one respective parameter per element, making this the actual challenge of control theory. As mentioned in Sect. 11.1.2, the control loop can be represented as a state space model. It is then the stability of this state space model that is essential. The parameters of the PID controller have to be chosen so that the system is stable. We will not pursue stability analysis further since in principal the procedure is similar to the one for models in population dynamics (Chap. 10) and in chaos theory (Chap. 12).

Many simple systems from various application areas can be divided into just a few categories by means of their properties. As long as one knows the approximate behavior for such a system, one can sometimes dispense with the modeling and forego the need for setting up a state space model. In this case, a typical controller is used for the respective category with the parameters adjusted to the particular problem. Complex modeling is almost indispensable, however, for more complex systems as well as for mostly all unstable systems.

11.2 Exemplary Modeling of a Multibody System

As already mentioned, the classical controller design requires a mathematical model of the system to be controlled. But this is by no means the only reason to set up a model. For example, it may be necessary to test a controller during the development process. Experiments involving the actual system are often not possible or only with a great deal of effort. On the one hand, the costs for such experiments can be very high while on the other hand experiments can also be quite dangerous for some

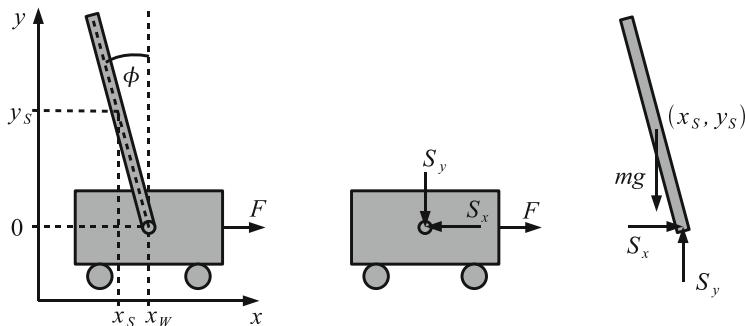


Fig. 11.3 Inverted pendulum (*left*) and separate components (*right*)

systems. In these cases it is advisable to test the controller through a simulation of the system. Therefore, a model first has to be set up. Since at a minimum the classical controller is designed with the help of a model, it will typically perform well for this specific model. If the robustness of a controller is to be tested through a simulation, then a different model or at least a perturbed model should be employed.

The quality required for the model depends entirely on what the control task involves. Let us again consider the example of the anti-lock braking system. Naturally, a model of the vehicle is necessary for the design of such a controller, but one can readily dispense with some of its parts. As an example, the windshield wipers or the airconditioning system have only a very limited influence on the driving behavior. Additionally, the temperature control for the vehicle does not require a modeling of the brakes. Thus, the model is always kept as simple as possible.

Equalling the diversity of the application scenarios for controllers are the approaches to develop models for these scenarios. As an illustration for control engineering methods we restrict ourselves to a simple mechanical example, namely the so-called *inverted pendulum*. Here, the objective is to vertically balance a pole that is mounted on a cart capable of horizontal movement. This placement of the pendulum is an unstable equilibrium, i.e., for a small disturbance the pendulum starts to tilt thus requiring that the controller react correspondingly. Such a system is illustrated schematically on the left side of Fig. 11.3.

Even for this relatively simple mechanical system there exist different methods for the modeling of it. In the following, we will use two of these methods in order to set up a model. A detailed introduction into engineering mechanics should not, and cannot, be given here, rather the objective is to convey some intuition regarding how the modeling of mechanical systems may proceed. For a deeper understanding of the underlying mechanics we refer the interested reader to [32].

11.2.1 Linearized Model with Conservation of Linear and Angular Momentum

The inverted pendulum is a so-called *multibody system*. It consists of two rigid bodies that are coupled. One possibility for its modeling is to set up momentum and angular momentum equations for each body and to derive a system of differential equations that results after consideration of the coupling. We do this in the following, however, in a strongly simplified form, namely linearized about the unstable equilibrium. This implies that our model is valid only when the pole is at an almost vertical position. This is sufficient as long as the model's design purpose is to keep the pole in the vertical position and as such there will not occur any large displacements from this vertical position. Any effects caused by friction are also neglected.

With this in mind, the first step is a free body diagram. Here, the individual components of the system (in this case the cart and the pole) are considered separately and the respectively acting forces are sketched (Fig. 11.3 right).

At the point coupling the cart and the pole, both bodies apply forces mutually toward each other which here are decomposed into x - and y -components. In view of Newton's third law, the forces that the cart applies to the pendulum correspond exactly to those that the pendulum applies to the cart. With the help of the conservation principles of momentum and angular momentum, the equations of motion can be set up. Expressed in a somewhat simplified manner, the principle of linear momentum,

$$m\ddot{x} = \sum_i F_i \quad (11.1)$$

says that the sum of the forces acting on a body correspond to the product of the mass and the acceleration. Analogously, the principle of conservation of angular momentum,

$$\Theta\ddot{\phi} = \sum_i M_i, \quad (11.2)$$

says that the sum of the moments corresponds to the product of the moment of inertia and the angular acceleration. From (11.1) and Fig. 11.3, we obtain the principle of linear momentum of the cart in the x -direction

$$m_W \ddot{x}_W = F - S_x, \quad (11.3)$$

in which m_W is the mass of the cart. We do not need the principle of linear momentum in the y -direction since the cart can only move horizontally. For the pole having mass m_S and length l , we obtain in an analogous manner the principle of linear momentum in the x - and y -direction,

$$m_S \ddot{x}_S = S_x, \quad (11.4)$$

$$m_S \ddot{y}_S = S_y - m_S g. \quad (11.5)$$

Since both pole and cart are coupled together, the position of the pendulum's center of gravity depends on the position of the cart as follows:

$$x_S = x_W - \frac{l}{2} \sin \phi, \quad (11.6)$$

$$y_S = \frac{l}{2} \cos \phi. \quad (11.7)$$

For the sake of simplicity, we assume that the pendulum must be considered only for very small displacements from the unstable equilibrium position. We can therefore approximate $\sin(\phi)$ by ϕ and $\cos(\phi)$ by 1. This simplifies (11.6) and (11.7) to

$$x_S = x_W - \frac{l}{2} \phi, \quad (11.8)$$

$$y_S = \frac{l}{2}. \quad (11.9)$$

Differentiating (11.8) two times yields

$$\ddot{x}_S = \ddot{x}_W - \frac{l}{2} \ddot{\phi}.$$

Since the y -position is constant according to (11.9), there is no acceleration in the y -direction. After direct substitution into (11.4) and (11.5), one obtains the equations

$$S_x = m_S \left(\ddot{x}_W - \frac{l}{2} \ddot{\phi} \right), \quad (11.10)$$

$$S_y = m_S g \quad (11.11)$$

for the forces at the point of coupling. After substitution into (11.3) one obtains the equation of motion

$$(m_W + m_S) \ddot{x}_W - m_S \frac{l}{2} \ddot{\phi} = F \quad (11.12)$$

of the cart. With respect to the equation of motion of the pendulum, one also has to set up the principle of conservation of angular momentum. To this end, all forces that act on the pole have to first be identified. Only S_x and S_y cause a moment since moments result only from forces whose line of action does not pass through the center of gravity (see Fig. 11.3 right). With (11.2) and taking into consideration the linearization assumption, it follows that the principle of conservation of angular momentum for the pendulum is given by,

$$\Theta \ddot{\phi} = \frac{l}{2} S_y \phi + \frac{l}{2} S_x.$$

Substitution of (11.10) and (11.11) together with the moment of inertia $\Theta = 1/12m_S l^2$ of the pole (that applies for thin poles with respect to the center of gravity) yields the equation of motion,

$$\left(\frac{l^2}{3}\right) \ddot{\phi} = \frac{l}{2} (\phi g + \ddot{x}_W) \quad (11.13)$$

for the pole. Equations (11.12) and (11.13) describe the motion of the inverted pendulum for the region around the unstable equilibrium. Despite the relatively simple system and the simplification through linearization, the modeling already requires a relatively high degree of effort.

Linear State Space Model

The model that was derived in the previous section consists of two second order differential equations. Appearing in each equation are both $\ddot{\phi}$ as well as \ddot{x}_W . Since the position of the pendulum's center of gravity x_S no longer explicitly appears in the equations but rather only that of the cart x_W , in the following we only make reference to the position x which refers to the position of the cart. But this representation of the model in terms of differential equations is neither particularly descriptive nor practical for the subsequent simulation. A more suitable representation is a state space model into which most every dynamical system can be converted. The state of each system can be precisely described through a certain number of variables or states. For example, the system in Fig. 11.1 can be described by one state, namely the height of the fill. For the inverted pendulum, exactly four states are required: The position of the cart, the angle of the pendulum, the velocity of the cart, and the rotational speed of the pendulum. The state space of the pendulum is spanned by these four variables. The dynamical behavior of the pendulum corresponds to a trajectory in this space in which a single point $(x, v, \phi, \omega)^T$ in the space defines a unique state of the system. The change of a state not only depends on the state itself but on external influences as well. If we assume that a linear system with n states has an input signal $u(t)$ and no output signal, one then obtains the general form

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + bu(t). \quad (11.14)$$

Here, the vectors $\dot{\mathbf{x}}(t)$, $\mathbf{x}(t)$ and b each have n elements and A is an $n \times n$ -matrix.

We now use the equations of motion (11.13) and (11.12) of the linearized pendulum to derive the state space model. Since $\ddot{\phi}$ as well as \ddot{x} occur in both equations, we first eliminate one of the two quantities in each equation through substitution to obtain,

$$\ddot{x} = c \left(l m_S \phi g + \frac{4}{3} F l \right), \quad (11.15)$$

$$\ddot{\phi} = 2c (\phi g(m_W + m_S) + F), \quad (11.16)$$

where the constant,

$$c = \frac{3}{l(4m_W + m_S)}.$$

We next transform the system of two differential equations of second order into a system of four differential equations of first order. To this end, we introduce the two additional variables v and ω :

$$\dot{x} = v,$$

$$\dot{\phi} = \omega.$$

These new variables have a physical meaning in this case. They correspond to the velocity of the cart and the rotational speed of the pendulum, respectively. After substituting these new variables into (11.15) and (11.16) we obtain,

$$\begin{aligned} \dot{v} &= c \left(l m_S \phi g + \frac{4}{3} F l \right), \\ \dot{\omega} &= 2c (\phi g(m_W + m_S) + F). \end{aligned}$$

We therefore have one equation for each of \dot{x} , \dot{v} , $\dot{\phi}$ and $\dot{\omega}$ and after substituting c we obtain the state space model in the form of (11.14):

$$\begin{pmatrix} \dot{x} \\ \dot{v} \\ \dot{\phi} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{3m_S g}{4m_W + m_S} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{6g(m_W + m_S)}{l(4m_W + m_S)} & 0 \end{pmatrix} \begin{pmatrix} x \\ v \\ \phi \\ \omega \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{4}{4m_W + m_S} \\ 0 \\ \frac{6}{l(4m_W + m_S)} \end{pmatrix} F.$$

Although this model is mathematically equivalent to the two differential equations (11.12) and (11.13), it has several advantages. On one hand, it is much better suited for the analysis of the system and for the design of an appropriate controller since the interdependencies of the system quantities as well as the influence of the input quantity on the system are both immediately apparent. On the other hand, it is also better suited for the computer-based treatment.

11.2.2 Complete Model with Lagrange Equations

In the previous two sections we used *Newton's laws of motion* in order to derive the equations of motion for an inverted pendulum. Despite the simplification realized

through linearization, this turned out to be relatively complex. There exist other formalisms in mechanics that simplify the derivation of the equations of motion. One such formalism is provided through the *Lagrange equations* in which energies rather than forces play the predominant role. Often, the derivation of the equations of motion becomes easier. But here, however, we refrain from a presentation of the details regarding the derivation of the Lagrange equations. Rather, we choose to demonstrate the use of the Lagrange equations as they apply to the example of the previously introduced pendulum. This time, however, we will dispense with the linearization assumption and consequently derive a model for the complete motion of the pendulum. This is meaningful, for example, when the swinging of the pendulum should be controlled as well or to enable a simulation of the pendulum. Friction will still be neglected.

When using Newton's laws, the body will first be cut free, i.e., it will be decomposed into its individual components. Conservation laws for momentum and angular momentum are then set up for each component. In the extreme case, there will be six equations per component (three degrees of freedom each for motion and rotation). Due to the coupling between the components as well as from external restrictions, the actual number of true *degrees of freedom* is smaller. The movement of the pole in our example is not independent of the cart. For the Lagrange equations, only the equations for the actual, mutually independent degrees of freedom q_i are set up. These degrees of freedom are also called *generalized coordinates*. For each of these f coordinates, one equation is set up:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = Q_i, \quad j = 1, \dots, f. \quad (11.17)$$

Q_i is the external force that acts on the i th degree of freedom, and

$$L = T - U \quad (11.18)$$

is the *Lagrange function* which corresponds to the difference between the kinetic energy T and the potential energy U . As a first step, the degrees of freedom of the system have to be determined. There are exactly two in this case, namely

$$q_1 = x, \quad (11.19)$$

the horizontal coordinate of the cart, and

$$q_2 = \phi, \quad (11.20)$$

the angle that describes the displacement of the pole. Because there are no external forces acting on the pole (the gravitational force will be taken into consideration as a component of the potential energy), Q_2 is therefore zero. The driving force F of the engine acts on the cart.

$$Q_1 = F, \quad (11.21)$$

$$Q_2 = 0. \quad (11.22)$$

The kinetic and potential energy have to be determined next. In general, the potential energy of a body is the result of the product of the mass m , the gravitational acceleration g and the height of the body's center of gravity h . We take the y -coordinate of the cart to be the zero level of the potential energy, therefore making the cart's potential energy equal to zero. Since the height of the pendulum is $\frac{l}{2} \cdot \cos \phi$, the pendulum's potential energy becomes

$$U_P = m_S g \frac{l}{2} \cos \phi. \quad (11.23)$$

The kinetic energy of a rigid body with mass m and moment of inertia Θ (with respect to the center of gravity) is composed of translational and rotational energy,

$$T = \frac{1}{2} m v^2 + \frac{1}{2} \Theta \omega^2, \quad (11.24)$$

where v is the velocity of the center of gravity and ω the rotational speed of the body. The cart does not display any rotation and moves with velocity \dot{x} , thus the kinetic energy is

$$T_W = \frac{1}{2} m_W \dot{x}^2. \quad (11.25)$$

For the translational energy of the pendulum, the velocity of the center of gravity has to be determined. This is composed of the velocity of the cart \dot{x} and the relative velocity difference \tilde{v}_P between the pendulum's center of gravity and the cart. We first decompose the velocity (rather, the square of the velocity) into its x - and y -components,

$$v_P^2 = v_{P_x}^2 + v_{P_y}^2 = (\dot{x} + \tilde{v}_{P_x})^2 + (\tilde{v}_{P_y})^2. \quad (11.26)$$

The velocity of the cart has no influence on the y -component. The relative velocity differences \tilde{v}_{P_x} and \tilde{v}_{P_y} depend on the length l of the pendulum, the angle ϕ and the angular velocity $\dot{\phi}$:

$$\tilde{v}_{P_x} = -\frac{l}{2} \dot{\phi} \cos \phi,$$

$$\tilde{v}_{P_y} = \frac{l}{2} \dot{\phi} \sin \phi.$$

Substitution into (11.26) yields

$$v_P^2 = \left(\dot{x} - \frac{l}{2} \dot{\phi} \cos \phi \right)^2 + \left(\frac{l}{2} \dot{\phi} \sin \phi \right)^2.$$

Through a simple reformulation it follows for the square of the velocity of the center of gravity that

$$v_P^2 = \dot{x}^2 - \dot{x}l\dot{\phi} \cos \phi + \frac{1}{4}l^2\dot{\phi}^2. \quad (11.27)$$

The moment of inertia Θ of the pole is $1/12m_S l^2$, and its rotational speed ω corresponds to the derivative $\dot{\phi}$ of the angle. Together, with (11.27) substituted into (11.24), one obtains

$$\begin{aligned} T_P &= \frac{1}{2}m_S \left(\dot{x}^2 - \dot{x}l\dot{\phi} \cos \phi + \frac{1}{4}l^2\dot{\phi}^2 \right) + \frac{1}{2}\frac{1}{12}m_S l^2\dot{\phi}^2 \\ &= \frac{1}{2}m_S \left(\dot{x}^2 - \dot{x}l\dot{\phi} \cos \phi + \frac{1}{3}l^2\dot{\phi}^2 \right). \end{aligned}$$

This, combined with (11.25) and (11.23) substituted into (11.18), yields the Lagrange function

$$L = \frac{1}{2}m_S \left(\dot{x}^2 - \dot{x}l\dot{\phi} \cos \phi + \frac{1}{3}l^2\dot{\phi}^2 \right) + \frac{1}{2}m_W \dot{x}^2 - m_S g \frac{l}{2} \cos \phi.$$

From this, one obtains the following partial derivatives:

$$\frac{\partial L}{\partial x} = 0, \quad (11.28)$$

$$\frac{\partial L}{\partial \dot{x}} = (m_S + m_W)\dot{x} - \frac{1}{2}m_S l \dot{\phi} \cos \phi,$$

$$\frac{\partial L}{\partial \phi} = \frac{1}{2}m_S l \sin \phi (\dot{x}\dot{\phi} + g), \quad (11.29)$$

$$\frac{\partial L}{\partial \dot{\phi}} = -\frac{1}{2}m_S \dot{x}l \cos \phi + \frac{1}{3}m_S l^2 \dot{\phi},$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}} \right) = (m_S + m_W)\ddot{x} - \frac{1}{2}m_S l \ddot{\phi} \cos \phi + \frac{1}{2}m_S l \dot{\phi}^2 \sin \phi, \quad (11.30)$$

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\phi}} \right) = -\frac{1}{2}m_S \ddot{x}l \cos \phi + \frac{1}{2}m_S \dot{x}l \dot{\phi} \sin \phi + \frac{1}{3}m_S l^2 \ddot{\phi}. \quad (11.31)$$

If one substitutes (11.30) and (11.28) together with (11.19) and (11.21) into (11.17), one then obtains the first equation of motion

$$(m_S + m_W)\ddot{x} - \frac{1}{2}m_S l\ddot{\phi} \cos \phi + \frac{1}{2}m_S l\dot{\phi}^2 \sin \phi = F. \quad (11.32)$$

Likewise, after a substitution involving (11.31), (11.29), (11.20) and (11.22), the second equation of motion is determined,

$$2l\ddot{\phi} - 3\ddot{x} \cos \phi - 3g \sin \phi = 0. \quad (11.33)$$

Equations (11.32) and (11.33) describe the complete motion of the system. This modeling by means of the Lagrange function appears at first to be no simpler than the use of Newton's laws. However, we now have a more accurate, not linearized model, and furthermore an approach that is more systematic and thus facilitates the modeling of larger systems.

Non-linear State Space Model

The model we derived by means of the Lagrange equations is non-linear and as a consequence also cannot be represented by a linear state space model of the form (11.14). But true as well for a non-linear model, the state vector specifies the current state of the system exactly. Furthermore, a change of the state still depends only on the state itself and the input quantity, just no longer in a linear way. This is expressed in the form

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t)), \quad (11.34)$$

where f denotes a vector function. In order to express the non-linear model in this form, we can again derive four differential equations of first order from the two differential equations of second order (11.32) and (11.33). This approach is analogous to that detailed in Sect. 11.2.1 with the difference that it results in a non-linear state space model.

11.2.3 Simulation of the Pendulum

Toward the end of the chapter we will design a fuzzy controller which shall keep the pendulum in the vertical position. Since in the previous section we derived a model of the pendulum, it can be used to test the fuzzy controller. For a simulation, the state space model in the form (11.34) must be discretized. In Sect. 2.4.5, several different methods for the discretization of ordinary differential equations of first order were introduced. Here, one quickly sees one of the advantages of the representation in the state space in which all differential equations are already of first order. The discretization techniques can therefore be used directly without further manipulations. Obviously, in the general sense it cannot be said which method is to be chosen. Since we have modeled the pendulum without friction, it is easy to

determine how good the chosen discretization technique is and whether the length of the time interval is appropriate. In the optimal case, as long as the external force F is zero and the pendulum is released from an arbitrary initial position, it always returns exactly to this initial position. If one chooses Euler's method and avoids the choice of a very small step width, one then quickly finds out that the simulated pendulum starts to oscillate more and more even though it is not affected by any outside influences. We pause now from presenting further details of the discretization here. Rather, we carry out a discretization for a very similar example in Sect. 12.4.2.

11.3 Fuzzy Set Theory

Before we can come to the topic of fuzzy control, we must first busy ourselves with the underlying *fuzzy set theory*. While fuzzy control does not require any modeling in the form of differential equations, it is nonetheless and obviously necessary that a certain knowledge about the system is available. The modeling of such knowledge using fuzzy sets is treated in detail in [59]. Colloquially, the term “fuzzy” means “blurred” or “out of focus”. Here, departing from the classical set theory, it means that there do not exist sharp bounds for the membership of an element to a set. This often correlates to the human perception much better. Correspondingly, such *blurred* values are often associated in terms of one's natural language use such as, e.g., “fast”, “cold” or “very high”. In order to get a computer to deal with such values, they must be quantified. This happens with the help of fuzzy set theory which is an extension of the classical set theory, but by no means an “inexact” set theory. A very detailed treatment of fuzzy set theory can be found in [61].

11.3.1 Membership in Fuzzy Sets

The scope of normal sets in mathematics is defined crisply. For example, for every vehicle one can determine in a unique way whether it belongs to the set of vehicles with a maximum speed of at least 200 km/h. In contrast, however, if one wants to know whether a vehicle belongs to the set S of “fast” vehicles, then the question of admission is not as easily answered. With a Porsche, one would be pretty certain that it clearly belongs to the set of fast cars, but for a Toyota Camry the classification quickly becomes much harder.

For crisply defined sets it always holds that an element either completely belongs to a set or not at all. But for fuzzy sets there is a *degree of membership* $\mu_A(x) \in [0; 1]$ for an element which gives the degree in which the element x belongs to the set A . For the Porsche described in the above example, it holds that $\mu_S(\text{Porsche}) = 1$, while one could, for example, decide to specify the degree of membership $\mu_S(\text{Passat}) = 0.6$ for the Passat. A classical subset A of a basic set Ω can therefore be defined as

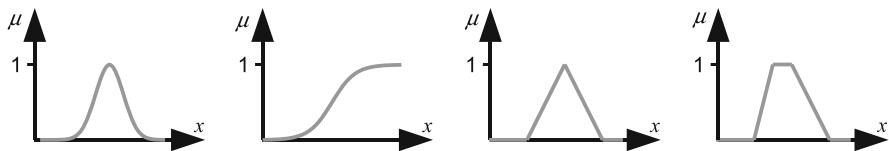


Fig. 11.4 Different possibilities for membership functions

Table 11.1 Monthly average temperature 2007

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Temperature [°C]	4.8	4.5	7.1	12.3	14.6	18.1	17.5	16.9	13.1	9.1	4.2	1.4

$$A = \{x | x \in \Omega, \mathcal{A}(x) \text{ is true}\}.$$

Here, $\mathcal{A}(x)$ is a predicate which is true if and only if x belongs to the set A . For every element x of the basic set Ω (e.g., the set of all vehicles, natural numbers, ...) one can therefore decide uniquely whether or not it belongs to the set A .

This is no longer true for the fuzzy set \tilde{A} ; it consists of pairs in which every $x \in \Omega$ is associated with a *membership function* $\mu_{\tilde{A}}(x)$:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) | x \in \Omega\}.$$

Thus, \tilde{A} contains all elements x of the basic set Ω , even though in parts it is true that the degree of membership is $\mu_{\tilde{A}}(x) = 0$. While the membership function of the crisp set only takes on the values 0 or 1, the fuzzy membership function permits gradual transitions; the crisp sets remain within the fuzzy sets as a special case. A few examples for membership functions are illustrated in Fig. 11.4.

Example of the Average Temperature

To again illustrate the difference between fuzzy sets and classical sets an example is given. Table 11.1 shows the monthly average temperatures in Germany for the year 2007. If now the warm months should be captured with a classical set, then a threshold value has to be determined above which a month is counted as warm so that it fully belongs to the set of warm months. In this example we set this value to 15°C. In order to facilitate our comparison with fuzzy sets, we also introduce a degree of membership which is one if and only if an element belongs to a set and zero otherwise (see Fig. 11.5 left). The right side of Fig. 11.5 gives the degree of membership that is “computed” for every month and depicted graphically. For the months of June, July and August we have that $\mu = 1$ since these three months have an average temperature above 15°C. All other months have a lower average

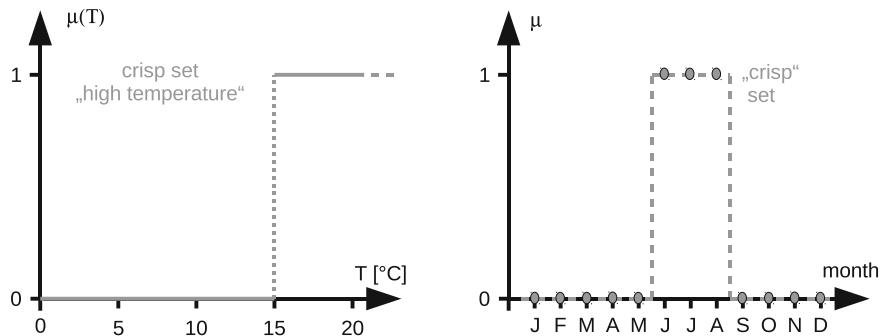


Fig. 11.5 Degree of membership for a crisp set (left), degree of membership of individual elements for the crisp set (right)

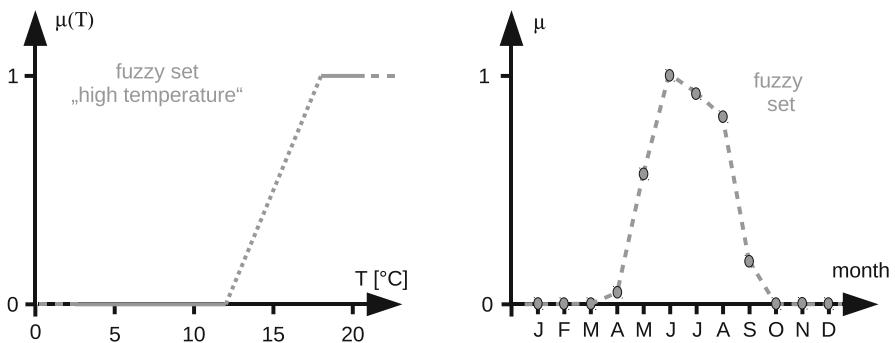


Fig. 11.6 Degree of membership for a fuzzy set (left), degree of membership of individual elements to the fuzzy set (right)

temperature and therefore are completely excluded from membership in the set of warm months (degree of membership $\mu = 0$).

Most people would agree with the association of June with the set of warm months as they would as well agree that December is not a warm month. For the spring and fall months, this impression is no longer as clear. For the transition to a fuzzy set, a function $\mu(T)$ has to be determined which specifies the degree of membership for a given temperature T . While there are various possibilities to do so, here for this chapter we will restrict ourselves to piecewise linear functions. For the example considered we have to specify an average temperature T_0 below which the month should not at all be considered as warm ($\mu(T_0) = 0$), and a temperature T_1 above which the month is considered to be wholly warm ($\mu(T_1) = 1$). Between these two values the degree of membership increases linearly (Fig. 11.6 left). This procedure results in the degrees of membership for the set of warm months that are shown on the right side of Fig. 11.6.

11.3.2 Operations with Fuzzy Sets

In order to be able to work with fuzzy sets in a reasonable way, the usual set operations (intersection, union and complement) need to be extended. To explain the operations we first start with the following definitions:

- Ω is the basic set,
- x always denotes elements in Ω ,
- \tilde{A} and \tilde{B} are fuzzy sets in Ω ,
- $\mu_{\tilde{A}}$ and $\mu_{\tilde{B}}$ are the degrees of membership of x to the sets \tilde{A} and \tilde{B} .

Complement

For crisp sets, the *complement* \bar{A} of a set consists of all elements in the basic set that are not contained in A . For a fuzzy set \tilde{A} , it is customary to form the complement set $\bar{\tilde{A}}$ by subtracting the degree of membership from one, i.e.,

$$\mu_{\bar{\tilde{A}}}(x) = 1 - \mu_{\tilde{A}}(x).$$

An important property with forming the complement, namely that the complement of the complement is again the original set, is satisfied for this formula. Nonetheless, there exist a few problems with this definition, for example, it is not guaranteed that the intersection—however this will be defined—of a set with its complement yields the empty set. After the following definition of intersection and union we will revisit this problem.

Union and Intersection

The most common operator used to formulate the *union* is the *maximum operator*:

$$\tilde{A} \cup \tilde{B} = \{(x, \mu_{\tilde{A} \cup \tilde{B}}) \mid \mu_{\tilde{A} \cup \tilde{B}} = \max(\mu_{\tilde{A}}(x); \mu_{\tilde{B}}(x))\}. \quad (11.35)$$

Analogously, the *minimum operator* can be used to define the *intersection*

$$\tilde{A} \cap \tilde{B} = \{(x, \mu_{\tilde{A} \cap \tilde{B}}) \mid \mu_{\tilde{A} \cap \tilde{B}} = \min(\mu_{\tilde{A}}(x); \mu_{\tilde{B}}(x))\}.$$

The gray areas in Fig. 11.7 illustrate the two operations.

As already indicated when the definition of the complement was introduced, there are properties that hold for crisp sets but may not hold for fuzzy sets, this depending on the definitions of the intersection and union operations. An example of this can be seen in Fig. 11.8. It shows a fuzzy set and its complement. If we now use the above definition for the intersection, then this is no longer the empty as it

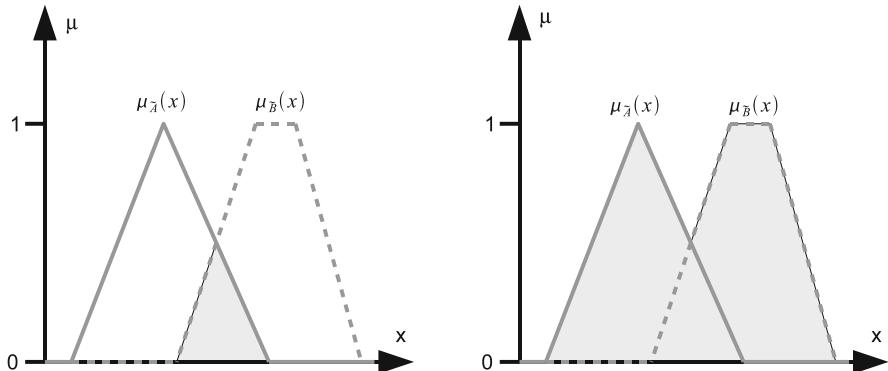


Fig. 11.7 Intersection of two fuzzy sets (*left*), union of two fuzzy sets (*right*)

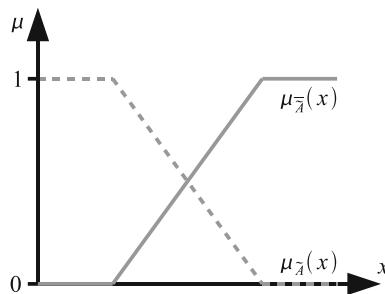


Fig. 11.8 Degree of membership of a set and its complement set

would be in the case for crisp sets. Likewise, for crisp sets it always holds that the union of a set and its complement returns the basic set. But, as well, this property is no longer satisfied with the union operation defined above.

For a suitable choice of union and intersection operations the properties just described can still be satisfied, e.g., with

$$\begin{aligned}\mu_{\tilde{A} \sqcup \tilde{B}} &= \min\{1, \mu_{\tilde{A}} + \mu_{\tilde{B}}\}, \\ \mu_{\tilde{A} \sqcap \tilde{B}} &= \max\{0, \mu_{\tilde{A}} + \mu_{\tilde{B}} - 1\}.\end{aligned}$$

However, other properties are now no longer satisfied for these operations, such as, e.g., the distributive property $(\tilde{A} \cap (\tilde{B} \cup \tilde{C})) = (\tilde{A} \cap \tilde{B}) \cup (\tilde{A} \cap \tilde{C})$.

As already mentioned, fuzzy sets are well suited to capture regions whose boundaries are blurry in linguistic formulations. Here, the fuzzy sets present a mathematically exact formulation of a blurry term. Although the above operations for fuzzy sets are defined precisely, they also have a blurry correspondence with respect to the normal language use. The straightforward correspondence would, e.g., be the use of the words “and” for the intersection and “or” for the union of

two sets. Since these linguistic terms can be interpreted differently depending on the situation (e.g., Or as an exclusive or not exclusive Or), there are also different possible realizations for set operations. The important part is that the operations used for the union and intersection satisfy certain norms (t-norm and s-norm [59]). We stop here and without further detail use exclusively the maximum and minimum operators.

11.3.3 Linguistic Variables

In the introduction to this chapter we stated that people tend toward vague or fuzzy specifications of values in many areas. We then demonstrated through the use of fuzzy set theory how such blurry value specifications can nonetheless be represented as a set. Now we want to take a closer look at the technique one can use to obtain fuzzy sets that correspond to given quantities. To this end we introduce the term *linguistic variable*. A linguistic variable is a variable whose values are fuzzy sets. The variable itself can map any physical quantities, such as temperature or color, for example, as well as quantities that can be quantified linguistically. The variable is therefore always defined over a certain basic set. The fuzzy values that the variable can assume are called *linguistic values* or *linguistic terms*. Some examples include fast, big, very cold, green, For a given linguistic variable it is important that, mostly due to fuzzy control and provided it is possible, the entire basic set be covered by linguistic terms. In the following example we introduce a linguistic variable that describes the rainbow colors.

Example: Rainbow Colors

Let the name of the linguistic variable be “rainbow color”. First, we must specify the basic set Ω . Light is electromagnetic radiation. What color we perceive depends on the wavelength λ of the radiation. A meaningful specification of the basic set is given to be,

$$\Omega = [380 \text{ nm}, 770 \text{ nm}],$$

and is therefore the region of the wave length that is perceivable to humans. The linguistic terms are the six rainbow colors “violet”, “blue”, “green”, “yellow”, “orange” and “red”. The transition between the individual colors is gradual, i.e., one cannot specify a crisp interval of wavelengths for every color. Each color, i.e., each linguistic term, is associated with a fuzzy set of wavelengths. To this end, there again naturally exist various methods that among other things, also depend on the perception of the individual person. Figure 11.9 shows one possibility for associating fuzzy sets with all of the terms. In this case, the interval lengths are all the same, but this could have been modeled differently as well.

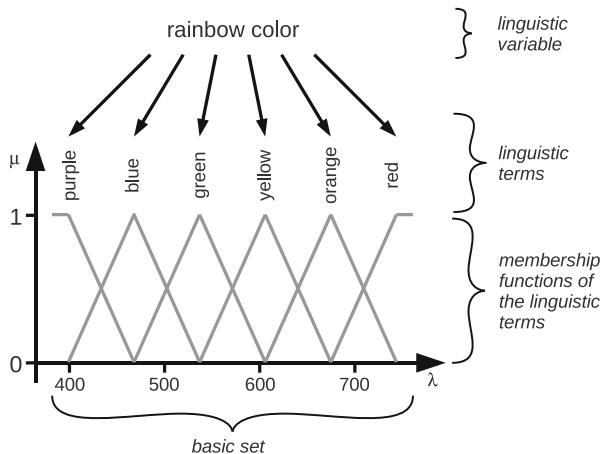


Fig. 11.9 The linguistic variable “rainbow color”

There also exist a few properties that do not necessarily have to be satisfied but would normally be of advantage for an application in fuzzy control. On one hand, the sets located at the boundary of the basic set are “unfolded”, i.e., at the outermost boundary they are assigned a value of 1. Furthermore, not only is the entire basic set covered, but for every element of the basic set, the degree of membership to at least one linguistic term is above 0.5. In fact, it even holds that for every element the sum of the degrees of membership is exactly 1.

11.3.4 Fuzzy Logic

Fuzzy set theory has given us the possibility to represent blurry information. In order to design a fuzzy controller, we still need the ability to process rules. In general, a controller is a system that gives a certain system input in response to a measured value. To this end, a fuzzy controller uses an arbitrary number of *implications* (rules), which are structured as follows:

$$\text{IF } <\text{premise}> \text{ THEN } <\text{conclusion}> .$$

As long as the *premise* is true, this implies that the *conclusion* is true as well. The premise as well as the conclusion are *fuzzy statements*. A fuzzy statement \tilde{A} has the form

$$x = \tilde{A}, \quad (11.36)$$

where x is a sharp value of the basic set of a linguistic variable, and \tilde{A} is a term associated with the linguistic variable. Analogous to the normal propositional

logic, fuzzy statements assume a truth value $v_{\tilde{A}}$. This corresponds to the degree of membership of the value x with respect to the fuzzy set \tilde{A} :

$$v_{\tilde{A}} = \mu_{\tilde{A}}(x). \quad (11.37)$$

Example Temperature Control

A simple example is a primitive temperature control. In the case the room temperature is too low, a valve at the radiator opens up, i.e., the measured value is the temperature and the system input is the valve position. We therefore have the two linguistic variables “temperature” and “valve position”. The linguistic terms have to be specified for both variables. For example, with respect to the variable temperature, let these be the terms T_1 up to T_5 while the valve position is represented by the terms V_1 up to V_5 . Each of these terms is associated with a fuzzy set. One possible rule could now be given as follows:

$$\text{IF } T = T_2 \text{ THEN } V = V_1$$

In practice, one can read this rule as “If the temperature assumes the fuzzy value T_2 , then the valve position must assume the value V_1 ”. Because the premise can also be composed of several statements, we will next look at how to link fuzzy statements.

Operations on Fuzzy Statements

Analogous to classical logic, operations can also be applied to fuzzy statements:

- $\neg \tilde{A}$: The *negation* corresponds to the complement of a fuzzy set. The truth value is computed by,

$$v_{\neg \tilde{A}} = 1 - v_{\tilde{A}}.$$

- $\tilde{A} \wedge \tilde{B}$: The *conjunction* corresponds to the intersection of fuzzy sets. The truth value is given as,

$$v_{\tilde{A} \wedge \tilde{B}} = \min(v_{\tilde{A}}, v_{\tilde{B}}). \quad (11.38)$$

- $\tilde{A} \vee \tilde{B}$: The *disjunction* corresponds to the union of fuzzy sets. The truth value is found using,

$$v_{\tilde{A} \vee \tilde{B}} = \max(\mu_{\tilde{A}}, \mu_{\tilde{B}}).$$

The last operation that is relevant to us is the *implication*, i.e., the fuzzy rule itself. We are not interested in the truth value of the implication since this would

Table 11.2 Implication from propositional logic

\mathcal{A}	\mathcal{B}	$(\mathcal{A} \rightarrow \mathcal{B})$
0	0	1
0	1	1
1	0	0
1	1	1

Table 11.3 Truth value of the conclusion

\mathcal{A}	$(\mathcal{A} \rightarrow \mathcal{B})$	\mathcal{B}
0	0	Impossible
0	1	Arbitrary
1	0	0
1	1	1

only correspond to the truth value of the rules we set up. Although it is conceivable that the established rules are not correct as it might occur for the automatic set up of rules (see [59]), here we assume that the rules are set up by an expert and as such there were not any grave mistakes made. With this, we know that the truth value of the implication will always be one. We can determine the truth value of the premise using the corresponding degree of membership so that the only piece that is not yet known is the truth value of the conclusion. In order to obtain it one employs so-called *approximate reasoning*. Although here we will refrain from a detailed mathematical justification, one can, for example, be found in [61]. Of use for our purposes here is to show how it works along with a brief illustrative explanation. To this end, we first look at the truth value table (Table 11.2) of the classical implication $(\mathcal{A} \rightarrow \mathcal{B})$ from the two-valued propositional logic. The truth value of \mathcal{A} equals zero in the first two rows. The truth value of the implication $(\mathcal{A} \rightarrow \mathcal{B})$ is seen to be one, i.e., it is independent of the truth value of \mathcal{B} since anything follows from something false. The truth value of \mathcal{A} is one in the last two rows and in these cases the truth value of the implication is identical to the truth value of \mathcal{B} .

In the framework of fuzzy control we want to describe the influence of the truth value $v_{\mathcal{A}}$ of \mathcal{A} on \mathcal{B} . Let us first consider Table 11.3, in which the truth value of the conclusion \mathcal{B} is given for given truth values of the premise \mathcal{A} and the implication $(\mathcal{A} \rightarrow \mathcal{B})$. Since we assume the correctness of the implication $\mathcal{A} \rightarrow \mathcal{B}$, only the second and fourth row are relevant and from these rows it is suggested that a large truth value $v_{\mathcal{A}}$ causes a large truth value for \mathcal{B} . This is realized in such a way that $v_{\mathcal{A}}$ serves as a restriction to the membership of \mathcal{B} :

$$\mu_{\mathcal{B}}(x) := \min\{\mu_{\mathcal{B}}(x), v_{\mathcal{A}}\}. \quad (11.39)$$

This strategy of approximate reasoning is called *Mamdani implication*. It is illustrated graphically in Fig. 11.10. The previous fuzzy set \mathcal{B} is bounded from above, i.e., the degree of membership of a value to the new set \mathcal{B} cannot exceed the truth value of the premise. For the practical realization we refer to Sect. 11.4.2.

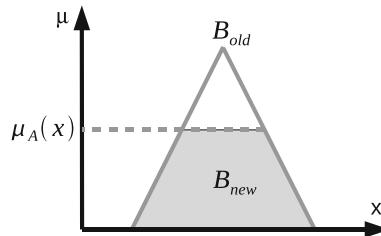


Fig. 11.10 Implication by Mamdani

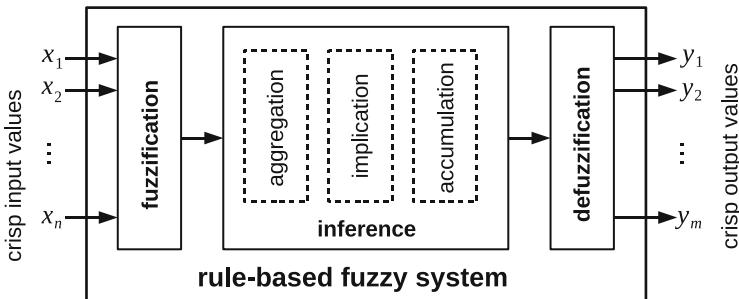


Fig. 11.11 Structure of a rule-based fuzzy system

11.4 Rule-Based Fuzzy System

Now that we have studied the basics of set theory, we can take the next step toward a fuzzy controller. We know from Sect. 11.1.1 that both the controlled system and the controller as well as the closed control loop are systems themselves. Our objective is to develop a controller based on fuzzy set theory. To this end, we work here to demonstrate how such a *fuzzy system* should look. The schematic structure for a rule-based fuzzy system can be seen in Fig. 11.11. As it applies to other systems as well, it is possible to have an arbitrary number of crisp inputs and outputs. But internally, however, a fuzzy system works with fuzzy sets. This means that the crisp input quantities must first be transformed into fuzzy quantities. This process is called *fuzzification*. The rule base of the system is evaluated using the fuzzy quantities that were obtained. This process is called *inference*. The result consists again of fuzzy quantities which are transformed to crisp output quantities through a process called *defuzzification*.

In the following, we will restrict our consideration to fuzzy systems with two inputs x and y and one output z . The inputs and the output are each associated with a respective linguistic variable having its own corresponding linguistic terms (fuzzy sets). Let these be given as,

Table 11.4 Rule base for $n = 3$ and $m = 4$

	\tilde{B}_1	\tilde{B}_2	\tilde{B}_3	\tilde{B}_4
\tilde{A}_1	$\tilde{C}_{(1,1)}$	$\tilde{C}_{(1,2)}$	$\tilde{C}_{(1,3)}$	$\tilde{C}_{(1,4)}$
\tilde{A}_2	$\tilde{C}_{(2,1)}$	$\tilde{C}_{(2,2)}$	$\tilde{C}_{(2,3)}$	$\tilde{C}_{(2,4)}$
\tilde{A}_3	$\tilde{C}_{(3,1)}$	$\tilde{C}_{(3,2)}$	$\tilde{C}_{(3,3)}$	$\tilde{C}_{(3,4)}$

$\tilde{A}_1, \tilde{A}_2, \dots, \tilde{A}_n$ for x ,

$\tilde{B}_1, \tilde{B}_2, \dots, \tilde{B}_m$ for y ,

$\tilde{C}_1, \tilde{C}_2, \dots, \tilde{C}_l$ for z .

For every input and output, the degree of membership to the respective linguistic terms can be computed. From this, one can also compute the truth value for the following statements.

$$\tilde{A}_i : (x = \tilde{A}_i), i \in \{1; n\}, \quad (11.40)$$

$$\tilde{B}_j : (y = \tilde{B}_j), j \in \{1; m\}, \quad (11.41)$$

$$\tilde{C}_k : (z = \tilde{C}_k), k \in \{1; l\}. \quad (11.42)$$

The rules of the fuzzy system have the form

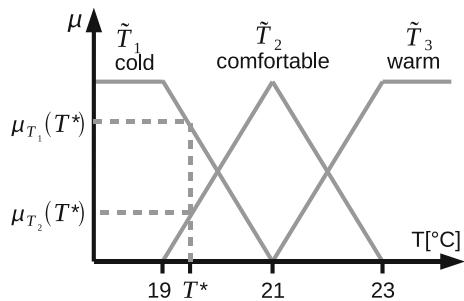
$$\text{IF } \tilde{A}_i \wedge \tilde{B}_j \text{ THEN } \tilde{C}_{(i,j)}.$$

Here, it holds that $i \in \{1; n\}$, $j \in \{1; m\}$ and $\tilde{C}_{(i,j)} \in \{\tilde{C}_k, k \in \{1; l\}\}$. It should also be pointed out that instead of the conjunction the other operations (negation and disjunction) could also have been used for the premise. But for reasons of simplicity we restrict ourselves in this chapter to the conjunctive composition of two statements. In the ideal case there exists a rule for every combination $(\tilde{A}_i, \tilde{B}_j)$. In this case, the complete rule base can be represented as a table. For $n = 3$ and $m = 4$, this can be seen in Table 11.4.

11.4.1 Fuzzification

Every input quantity of a fuzzy system must be associated to a linguistic variable and its corresponding linguistic terms. At the fuzzification, the degree of membership of the input quantity is calculated for each of the linguistic terms. Figure 11.12 shows the fuzzy variable temperature with its three linguistic terms cold, comfortable, and warm. The sharp measured value of the temperature is $T^* = 19.5^\circ\text{C}$. This results in the degrees of membership $\mu_{\text{kalt}}(T^*) = 0.75$, $\mu_{\text{angenehm}}(T^*) = 0.25$ and $\mu_{\text{warm}}(T^*) = 0.0$.

Fig. 11.12 Fuzzification of a temperature measured value



11.4.2 Inference

Inference is the evaluation of the rule base and can be divided into three steps:

- *Aggregation*: Determine the truth value of the premise for all rules
- *Implication*: Execute the conclusion for all rules
- *Accumulation*: Determine the overall result for the rule base

As input from the fuzzification, the inference receives the degrees of membership of the sharp measured values to the linguistic terms of the respective linguistic variable. These degrees of membership can be interpreted as truth values of corresponding fuzzy statements (see (11.36) and (11.37)).

Aggregation

In general, an arbitrary number of statements can be combined in an arbitrary number of ways to define a premise. But since we have restricted ourselves to systems with two input quantities and to rules with conjunctive composition, the premise is always of the form $\tilde{A} \wedge \tilde{B}$. Following (11.38), the truth value of the premise is therefore $v_{\tilde{A} \wedge \tilde{B}} = \min(v_{\tilde{A}}, v_{\tilde{B}})$. In the aggregation process, every truth value of the premises for all rules must be determined in this way.

Implication

In Sect. 11.3.4, the basic principle of approximate reasoning was introduced. Thus, Eq. (11.39) must be applied to each rule in the rule base. Since we restrict ourselves to systems with one output, all rules make statements regarding the same linguistic variable (e.g., valve position). There exists a certain number of linguistic terms for this variable. Each rule restricts one of these terms. But it may happen that a linguistic term surfaces for several different conclusions. In this case, the maximum of the two truncated sets is determined.

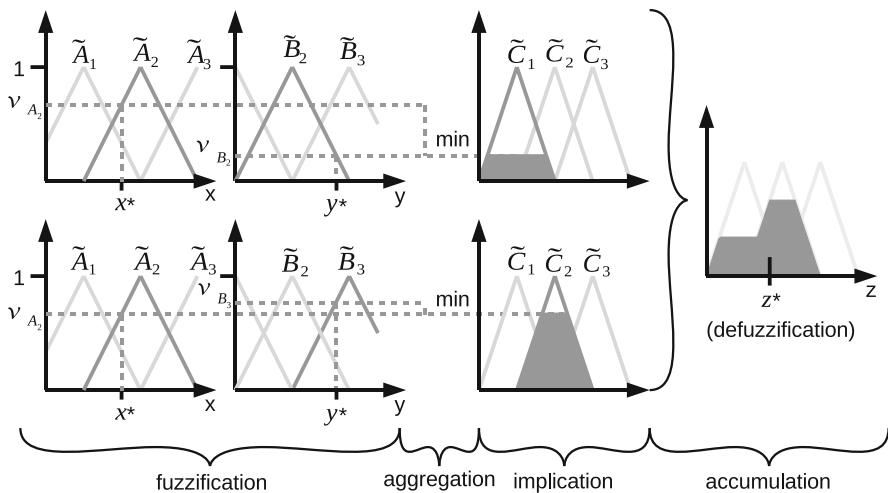


Fig. 11.13 Example for the evaluation of two rules

Accumulation

After evaluating all rules there now exists a new membership function for each linguistic term and thereby a new fuzzy set. Before a crisp result value can be computed, these sets have to be combined into one set. This happens by forming a union as in (11.35). Since the sets are obviously defined over the same basic set, this can be accomplished without any problems. This can be seen in the illustration given in Fig. 11.13. For systems having several outputs, the union would have to be performed for the linguistic terms associated with an output quantity, i.e., a linguistic variable.

11.4.3 Defuzzification

The resulting fuzzy set from the inference is used to form a crisp output quantity. There again exist different methods (see [37]) from which we will only consider the so-called center of gravity method. For this method, one calculates the abscissa value of the center of gravity for the area computed for the accumulation. Mathematically, this value is computed for a resulting set \tilde{C} defined over z as follows:

$$z^* = \frac{\int_z z \mu_{\tilde{C}}(z) dz}{\int_z \mu_{\tilde{C}}(z) dz}.$$

With the help of this formula one can see the disadvantage of the center of gravity method, namely its relatively expensive computation cost. By comparison, other

methods directly use the z value for which $\mu_{\tilde{C}}(z)$ is maximal. This is significantly easier to compute, but has the disadvantage that only a single rule determines the result. This has the consequence that slight changes in the input quantities can cause large jumps in the output quantity. The center of gravity method is used quite frequently despite its complexity since it ensures that all rules always contribute partially to the result.

11.4.4 Example

Let there be three linguistic terms ($A_i, B_i, C_i; i \in 1, 3$) defined for each of the linguistic variables A , B and C . Further, the inputs x and y are associated with the variables A and B , and the output z is associated with the variable C . Figure 11.13 shows the complete course of action for a rule-based fuzzy system given the measured values x^* and y^* and a rule base having the following two rules:

IF $\tilde{A}_2 \wedge \tilde{B}_2$ THEN \tilde{C}_1 ,
IF $\tilde{A}_2 \wedge \tilde{B}_3$ THEN \tilde{C}_2 .

The statements \tilde{A}_i , \tilde{B}_j and \tilde{C}_k are formed according to (11.40)–(11.42).

11.5 Fuzzy Control of the Inverted Pendulum

We finally have the theoretical prerequisites now that are needed to design a fuzzy controller. In this section we will take a closer look at one of our previous examples, namely the inverted pendulum that had been introduced in Sect. 11.2. As mentioned before, the design of the fuzzy controller does not require a mathematical model of the controlled system. Yet, it is nonetheless true that knowledge about the behavior of the controlled system must be available and included in order to manipulate it in a controlling manner. This knowledge is incorporated into a rule base (see Table 11.4). The set-up of the rule base thus corresponds in a certain way to the modeling of the controlled system. The inverted pendulum is a system that is already relatively difficult to control since the state that we want to maintain through the controller is unstable. It would become even more difficult if, starting from an arbitrary initial position, one would require the controller to move the system into the vertical position and stabilize it there. In a realistic case, the pendulum would initially hang down and would have to be swung up first.

From the point of view of a human controller it is better to treat the two tasks “swing up” and “stabilize” separately, i.e., one can design a separate controller for each of them and switch over from one to the other. Since the rule base of a fuzzy controller is, among other things, built upon human know-how, it does no harm to

ponder about the manner in which one would solve these two control problems as a human. The stabilization corresponds to the problem of balancing a pencil (from the pointed end, of course) on the fingertip. Those having tried this already know that this is impossible without a great deal of practice. In contrast, the swinging up is really easy. Those not having easy access to a pendulum rod with a freely rotary joint can instead attempt to swing a piece of rope using a horizontal movement of the hand so that it performs a complete 360° revolution. We point out that in this section our objective is not to design two controllers in full detail and then apply these to an actual system. This is beyond the scope of this example. But we do, however, want to illustrate, from the beginning to the end, the basic procedure needed in solving a control problem with the help of a fuzzy control.

11.5.1 Parameters and Constraints

Before we begin with the design of the controller we must briefly specify the test environment: On one hand, there exist direct parameters of the pendulum, namely the mass m_W of the cart, the mass m_S of the pendulum rod and the length l of the pendulum rod. The pendulum will behave differently depending on the choice of these parameters. Now, a fuzzy controller is often relatively robust toward slight changes in the system, but the change of these parameters still has an effect. If, for example, the masses of the cart and pole are doubled but the controller still only applies the same force as before, the acceleration of the cart or the rotational acceleration of the pendulum are then only half as big.

An additional constraint for a real system is the restriction of the input, output and state quantities. For the inverted pendulum we have, as we have shown in Sect. 11.2.3, the state quantities ϕ , ω , x and v . Only the angle ϕ is not subject to any restriction; the pendulum can rotate arbitrarily. The x position typically has a very sharp bound, namely the length of the track on which the cart moves. The velocity v depends on both the maximum number of revolutions of the driving engine as well as on the quality of the device in general. The angular velocity ω is mainly bounded by the proportion of pole mass to cart mass. If the pendulum turns too quickly, the cart can take off if it is not kept mechanically at the bottom. There are no additional output quantities since we assume that the states of the system can be measured directly. The only input quantity is the force applied by the engine, which in turn is naturally bounded by the properties of the engine. Instead, one could use as well the voltage applied to the engine or possibly even the velocity of the cart as the input quantity. The latter would simplify all things significantly but is realistic only for very special pendulums. In the following we stick to the force as the input quantity.

The multitude of possible parameters and constraints leads to the understanding that the design of a fuzzy controller is not something that one does only once and then can apply to all systems that are similar. Even though mathematical modeling is unnecessary, the specific properties of the considered system must still be taken into consideration. For this reason, we do not specify here any particular values for

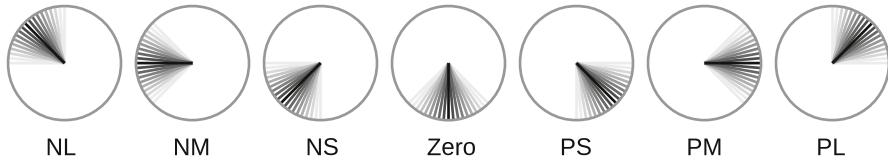


Fig. 11.14 Linguistic terms of the variable “angle”

the parameters and constraints. This holds as well for the linguistic variables that are predominantly of qualitative and not quantitative nature.

11.5.2 Swinging Up the Pendulum

Since swinging up the pendulum is the problem that is easier to solve from a human’s perspective, we will treat it first. The first step toward a controller is the introduction of the linguistic variables for the state quantities ϕ , ω , x and v of the system. As names for the corresponding linguistic variables, we take for obvious reasons the meaning of the variable, i.e., “angle”, “angular velocity”, “position” and “velocity”, respectively. We now must assign linguistic terms to these variables. This is one of the most critical steps for the design of a fuzzy controller and is a kind of modeling of the system since it must readily incorporate a lot of knowledge about both the system as well as later for the controller concept. In order to specify the linguistic terms for the angle, we should already know beforehand the region in which the controller should become active. For this, we need a basic knowledge of the physical relationships present. The objective is to give the pendulum a rotational velocity sufficient so that it just barely reaches the upward vertical position. A change in the rotational velocity is obtained via a respective rotational acceleration. Remaining is the question regarding the manner to influence the rotational acceleration according to our ideas. The rotational acceleration depends on the acting moments via (11.2). A moment in turn arises when a force acts on the pole whose line of action does not pass through the center of gravity. In our example, we apply a horizontal force which acts on one end of the pole. If the pole itself is aligned horizontally, then we cannot influence the rotational acceleration at all. At first sight this sounds like a drawback but this can also be viewed as an advantage. Since the length of the guide rail is typically bounded, the controller should take care that the cart remains in the middle of the track. If the pendulum rests in an almost horizontal position, the cart can then be accelerated in the direction of the middle of the rail without having much influence on the behavior of the pendulum. The region above the horizontal can also be used again to accelerate or to decelerate the rotation if necessary.

One possibility for the assignment of terms to the linguistic variable “angle” is represented in Fig. 11.14. We see that the angle is approximately zero in the region in which the pendulum is close to the downward directed vertical position. This

Table 11.5 Rule base for the upswinging controller; the top row shows the terms for the angle while the left column shows the terms for angular velocity

	NL	NM	NS	Zero	PS	PM	PL
NL	2	–	–1	0	1	–	–2
NS	1	–	0	1	2	–	–1
Zero	0	–	–2	0	2	–	0
PS	–1	–	–2	–1	0	–	1
PL	–2	–	–1	0	1	–	2

fuzzy region is covered by the linguistic variable “zero” and uses the typical linear hat function that is nonzero in the lower quarter of the circle. For the remaining terms, the first letter (N or P) stands for negative or positive, resp., and the second letter (S, M, L) for small, medium or large. In this way we can cover the entire basic set with overlapping fuzzy sets (see Fig. 11.9 for a different representation of linguistic terms and the corresponding variables). Only the upward directed vertical position has been left out since this should be treated by the second controller that is responsible for the stabilization.

While the basic set for the angle is essentially given as the entire circle, it is more difficult for the angular velocity to determine what, for example, actually designates a “large angular velocity”, i.e., the range of values where the terms are defined. A small experiment, however, quickly helps us out. We simply release the pendulum from an almost upward vertical position and measure the rotational velocity for various angles. With the same velocity (by modulus) in the opposite direction one returns back to exactly the initial position. From this it is easy to estimate the angular velocities that are required for given angles. The linguistic terms for the angular velocities must now be chosen so that slightly too small as well as slightly too large velocities are included as well. The rules will then be designed in a way that attempts to obtain the previously measured optimal rotational velocity in dependence of the angle. In the following, we assume that there are five linguistic terms for each the rotational velocity as well as for the force that is to be applied and we use as their identifiers “NL”, “NS”, “Zero”, “PS” and “PL”. For example, the rule base given in Table 11.5 can now be set up in which the first row contains the linguistic terms for the angle and the first column contains those for the rotational velocity. The other cells contain the conclusions of the fuzzy rules, i.e., the force that is to be applied. For better clarity, we use the numbers –2 to 2 instead of the linguistic terms, where, e.g., –1 stand for “NS”: We choose not to discuss more on the individual values but instead we will point out those values that are missing. The two empty columns correspond to the horizontal angle positions. We wanted to use these positions to keep the cart in the middle. To this end, we must of course look at the position and not at the angular velocity as the second variable. In fact, we would need a four-dimensional table in order to cover every combination produced from the linguistic terms of the four variables. Normally, however, this is neither possible nor necessary. Instead, one sets up rules whose premise contains 2, 3 or 4 terms as it becomes necessary. The fewer terms contained in the premise, the larger the part of the state space covered by this rule since a premise with fewer terms can be satisfied for many more states.

Table 11.6 Rule base for the stabilizing controller; the top row contains the terms for the angle and the left column contains the terms for the angular velocity

	NL	NS	Zero	PS	PL
NL	–	–	2	1	0
NS	–	2	1	0	-1
Zero	2	1	0	-1	-2
PS	1	0	-1	-2	–
PL	0	-1	-2	–	–

11.5.3 Stabilizing the Pendulum

Similar to the upswinging controller, one first needs to define the linguistic variables and the associated terms for the stabilizing controller. The variables obviously remain the same since they correspond to the state quantities. Only the linguistic terms and their corresponding values change. For the angle, one requires a much smaller region around the unstable vertical equilibrium, and as a consequence the values for the velocity should also be smaller since the pendulum is slowed down on its way up.

As before, we do not want to discuss the design of the rule base in detail, but in closing only mention briefly that for a good specification of the linguistic terms it may no longer be absolutely necessary to individually set up every rule. The rule base in Table 11.6 reveals an extremely systematic structure that is essentially based on two basic ideas. On one hand, the force becomes larger with increased deviation of the pendulum from the equilibrium while on the other hand the force becomes larger the faster the pendulum moves away from the equilibrium (or the force becomes smaller the faster the pendulum moves towards to equilibrium).

This almost sounds too easy to be true. Indeed, the rule base given in Table 11.6 has to be adjusted to the respective system. It requires a very careful choice of linguistic terms and also some trial-and-error experiments are necessary, but in the end, this simple rule base is indeed sufficient. Moreover, this was exactly the objective we wanted to obtain using fuzzy control—to save ourselves the exact modeling of the system and the challenging classical control yet still be able to control the system. What admittedly is no longer possible is a simulation of the pendulum since we no longer have an exact mathematical model.

11.6 Outlook

Control engineering is a very large and at many junctures also a very challenging subject area. A more detailed introduction into the system theoretical basics is offered in [44] while a somewhat stronger focus on applications and computer implementations is provided in [15]. Furthermore, there exists an abundance of literature also available for particular aspects of control engineering. While it was not the objective to cover this field in its full breadth and depth here, we primarily

intended to first convey a certain basic understanding regarding the elementary notion of control in order to then be able to take a closer look at modeling: first employing methods of technical mechanics and then with the help of fuzzy logic. Once again, additional literature is recommended for those interested in developing a deeper understanding. Besides the references mentioned above, [37] also covers a wide range of application examples. In [16], the theory of fuzzy sets is treated in somewhat less detail, but it does consider many aspects of fuzzy control, for example non-linear and adaptive fuzzy control and the stability of fuzzy controllers.

Chapter 12

Chaos Theory

With the mention of the word *chaos theory*, many possibly think first of the fascinating pictures of fractals as in the “Mandelbrot set”, for example. But these pictures show only a portion of what chaos theory is about. Chaos theory essentially deals with the analysis of *nonlinear dynamical systems*, i.e., of systems whose dynamics are described, e.g., through nonlinear differential equations. Almost every system occurring in reality displays nonlinear behavior and it is for this reason that people from the most diverse branches of science are interested in such systems. A prominent example is of course the weather which can never be predicted for arbitrarily long periods into the future, no matter what computer power is available, since just now and then a bag of rice might be dropped in China. In some situations, an apparently fortuitous, so-called *chaotic* behavior can often be observed in the systems being analyzed.

To the point: What does chaos have to do with modeling and simulation? At first, the objective is to understand chaotic behavior, to discover structures in such a behavior—the classical task of chaos theory. To model and simulate systems with chaotic behavior—despite its inaccessibility—is of importance for many scientific fields. In mathematical finance, given chaotic behavior one tries to draw conclusions on the development of money markets; in neurology, chaos theory is used to predict epileptic seizures. The reason that chaos theory has established itself as an independent research field rests on the fact that such seemingly curious behavior can be found in the most diverse applications as well as the fact that similar structures can frequently be discovered. One is almost inevitably dependent on computer simulation for the analysis of nonlinear systems since only a small number of nonlinear equations can be solved analytically.

Chaos appears in *discrete* (e.g., discrete mappings) as well as in *continuous* (e.g., differential equations) systems. To provide a better illustration of chaos, we will explain some of its essential characteristics such as *bifurcations* and *strange attractors* by means of simple discrete mappings. But the same characteristics can also be observed for continuous systems. In the last part of the chapter, we will briefly model a mechanical system—a driven pendulum—and then use the

simulation results to look at the chaotic behavior of the system. The sections on analysis as well as the numerical treatment of ordinary differential equations found in Chap. 2 are helpful and facilitate the understanding of this chapter.

12.1 Introduction

Lacking is a consistent definition for chaos theory or chaos itself. The term chaos is frequently used incorrectly, in particular in connection with random behavior. When we speak of chaos, we are in fact always concerned with the behavior of deterministic systems in the absence of any stochastical influence. For this reason, we introduce brief, informal definitions to convey an intuition what this chapter is actually about.

Chaos

A deterministic system is chaotic in its actions if its response is very sensitive to slight changes in the initial conditions. If one simulates the same system twice for lengthy time spans but with minimal difference between the initial states, then comparatively, the two end states no longer suggest that both systems were started with almost identical initial states. In terms of cause and effect, while exactly identical causes lead to the same effects for a chaotic system as for every other deterministic system, nearly identical causes may lead to completely different effects. One can also say that such a system is unpredictable. This is nicely expressed by the famous question whether the fluttering of a butterfly's wing can cause a hurricane at the other end of the world, however, it has nothing to do with *randomness* but rather with the special properties of the underlying nonlinear equations.

Chaos Theory

Chaos theory is the theory of dynamical nonlinear systems. Chaos theory does not only deal directly with chaotic behavior, but also with any behavior of nonlinear systems. This also includes *fixed points* as well as their *stability*, and *cycles*, which we will also discuss briefly.

There exist discrete as well as continuous systems that display chaotic behavior. For discrete systems, which are also called iterated functions, chaos can occur with only a single state variable while for continuous systems at least three state variables are necessary. For a discrete system (also called mapping), the state is only computed at discrete time instances t_n . Here, the state at time t_{n+1} is computed from the state at time t_n . We will mainly deal with a selected mapping, namely with the logistic mapping. The advantage of this mapping is that, while it is very

simple, all the important properties of chaotic systems can be explained. Next, we will briefly discuss a discrete system with two states since some things can be visualized and therefore illustrated better because of the two dimensions. Toward the end of the chapter we will introduce a relatively simple mechanical system—a driven pendulum—to show on one hand that the effects seen for discrete systems also appear in continuous systems while on the other hand to have a real system that everyone can also imagine visually.

12.2 From Order to Chaos

As already mentioned, chaos can be observed for discrete as well as for continuous systems. Both resemble each other with respect to the methods for their analyses as well as the observed effects. One-dimensional discrete systems are easy to comprehend in view of their low complexity and they are also particularly easy to simulate: We will therefore focus mainly on such a one-dimensional discrete system which we will exploit in order to briefly explain the basic phenomena.

12.2.1 Logistic Mapping and Its Fixed Points

Let an *iterated function* $\Phi(x)$ (also called a *discrete mapping*) be defined as follows:

$$x_{n+1} = \Phi(x_n), \quad (12.1)$$

where x_n and x_{n+1} denote the values at times n and $n + 1$, respectively. For example, the chapter on population dynamics made use of such iterated functions in order to analyze the development of a population over a certain time span. The discrete instances of time could be years or even involve the different time units that are related to the reproduction cycle of the species. In the analysis of iterated functions one is often interested in *fixed points*. Fixed points are points that are mapped to themselves, i.e., it holds that

$$\Phi(x) = x. \quad (12.2)$$

Provided that a fixed point is *attractive* and thus also *stable*, a sequence that started with the value x_0 in a certain neighborhood containing the fixed point will always converge toward the fixed point over the course of time. For the example of population dynamics, this would be the number of animals for which in a given year the number of deaths equals the number of births. Whether a fixed point is attractive or repulsive depends on the first derivative of the iterated function. If its absolute value is less than one, then the fixed point is attractive; if it is larger than

one, then the fixed point is *repulsive*. We do not go into detail for the special case that the absolute value is exactly one. For attractive fixed points, the behavior of the system can be divided into two parts. Starting with an arbitrary initial value, the fixed point is typically not reached within a single iteration step but there is a certain transient behavior, called the *transient dynamic*. As soon as the fixed point is reached, the transient dynamic approaches zero and the system has a *stationary behavior (asymptotic dynamic)*.

We consider the following iterated function that is also called a *logistic mapping* that is used for the discrete modeling of population growth:

$$\Phi(x) = rx(1 - x). \quad (12.3)$$

For our investigations we restrict ourselves to initial values $x_0 \in [0, 1]$. The behavior of this logistic mapping is very sensitive to the choice of the parameter $r \in [0, 4]$. The restriction of r to values between zero and four ensures that the values for $\Phi(x)$ remain within the interval $[0, 1]$. We first determine the fixed points, i.e., points that, according to (12.2), satisfy

$$x = rx(1 - x).$$

Immediately, we find that the first fixed point is $x_a = 0$. To determine additional fixed points we divide both sides by x and then solve for x yielding a second fixed point,

$$x_b = \frac{r - 1}{r}.$$

In order to analyze the stability of the two fixed points, we find the derivative,

$$\Phi'(x) = r - 2rx < 3$$

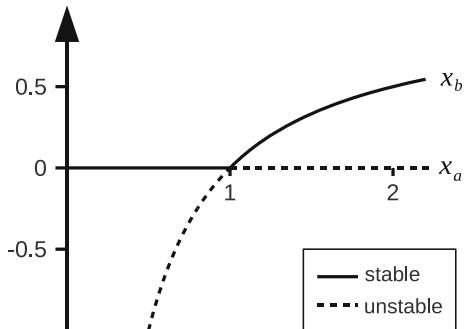
of (12.3). For the first fixed point $x_a = 0$ it holds that,

$$|\Phi'(0)| = |r| \begin{cases} < 1 \text{ for } 0 \leq r < 1, \\ > 1 \text{ for } 1 < r \leq 4. \end{cases}$$

For values $r < 1$, x_a is hence an attractive fixed point. This also becomes apparent if one takes a look at the three factors r , x and $(1 - x)$ in the iterated function (12.3). For $r < 1$ and $x_0 \in [0, 1]$, all three factors are less than one, and consequently, the absolute value of x must become smaller and smaller by iterating continuously and thus converge towards zero. For the second fixed point, $x_b = \frac{r-1}{r}$, it holds that

$$\left| \Phi' \left(\frac{r-1}{r} \right) \right| = |2 - r| \begin{cases} > 1 \text{ for } 0 \leq r < 1, \\ < 1 \text{ for } 1 < r \leq 3, \\ > 1 \text{ for } 3 < r \leq 4. \end{cases}$$

Fig. 12.1 Transcritical bifurcation for the logistic mapping



For values of r between zero and one, the second fixed point x_b is unstable, but between one and three it is stable. Figure 12.1 shows the stability of the two fixed points with respect to r . Here one sees that $r = 1$ serves as the point in which the two fixed points reverse their respective stability behavior. For such a situation, one speaks of a *transcritical bifurcation*. In general, a *bifurcation* denotes a qualitative change of the dynamic behavior of a system when a system parameter is changed. The transcritical bifurcation is only one example how such a qualitative change may look.

12.2.2 Numerical Analysis and Bifurcations

We have performed the previous analysis of the logistic mapping in a purely analytic way and in doing so we have determined the fixed points and their stability for values of the parameter $r \in [0, 3]$. Here, the stable fixed points correspond to the asymptotic dynamic of the logistic mapping. The behavior of the logistic mapping becomes interesting when $r \in [3, 4]$. For this interval, there do not exist any stable fixed points. However, in view of the structure of the mapping we do know that for these parameter values an initial value $x_0 \in [0, 1]$ is again mapped to the interval $[0, 1]$. Hence, the question comes up about the resulting asymptotic behavior. In the following we will no longer use analytical methods for its analysis since these are either not at all applicable or only produce results with very large effort. Rather, we will rely on numerical methods which means that we use the iterated function repeatedly until the transient dynamic has decayed and only the asymptotic dynamic is visible.

This is represented in Fig. 12.2 (left) for the logistic mapping with parameter $r = 1.5$ and initial value $x_0 = 0.1$. One sees how the current value x_i approaches the fixed point x_b monotonically as the iteration progresses. The same iteration can also be constructed graphically (right in Fig. 12.2) by starting with a point on the diagonal at $x_0 = 0.1$ and first drawing a vertical line to the graph of the iteration function $\Phi(x)$. One then draws a horizontal line from the

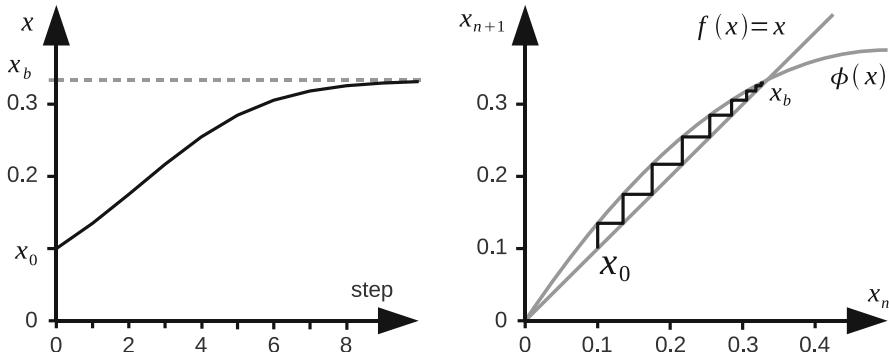


Fig. 12.2 Iterative approximation to $x_b = \frac{1}{3}$ ($r = 1.5$, $x_0 = 0.1$) (left) and graphical construction of the iteration (right)

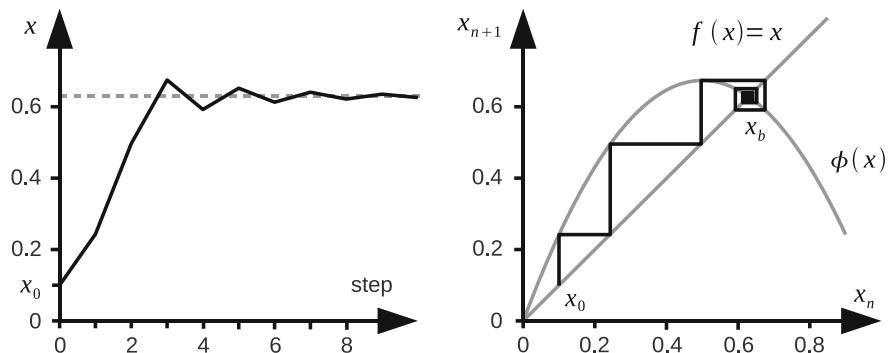


Fig. 12.3 Iterative approximation to $x_b = 0.63$ ($r = 2.7$, $x_0 = 0.1$) (left) and the graphical construction of the iteration (right)

obtained point of intersection back to the diagonal. Repeated application yields the displayed *graphical iteration*. The fixed point obviously corresponds exactly to the intersection of $\Phi(x)$ with the diagonal since only then $\Phi(x) = x$ holds. The advantages of the graphical iteration will be made clear later. But we first want to consider the transient dynamic for a second example in which the parameter is $r = 2.7$. We already know that there exists a stable fixed point for this value of the parameter, so the asymptotic dynamic at least does not differ qualitatively.

Nonetheless, one sees from Fig. 12.3 that the approximation to the fixed point is no longer monotonic. Although the values oscillate around the fixed point, the sequence still converges toward the same one. The graphical iteration shows that the source for this oscillation around the fixed point is the negative derivative of the iteration function $\Phi(x)$ at x_b . If now the parameter r is further increased to a value greater than three, the absolute value of the derivative at the fixed point then becomes greater than one which tells us there no longer exists a stable fixed point.

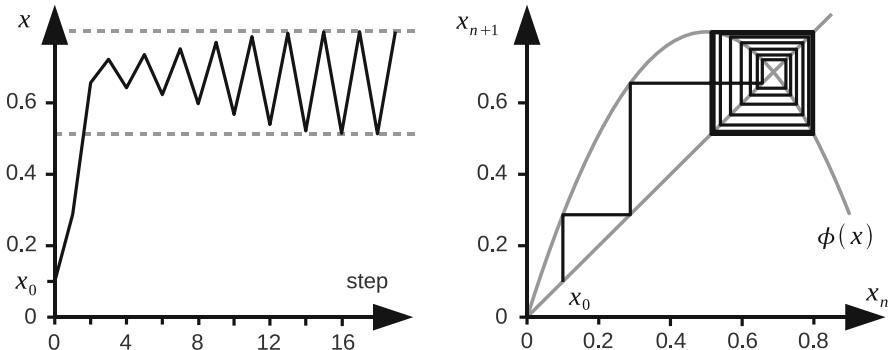


Fig. 12.4 Iteration for a stable 2-cycle ($r = 3.2$, $x_0 = 0.1$)

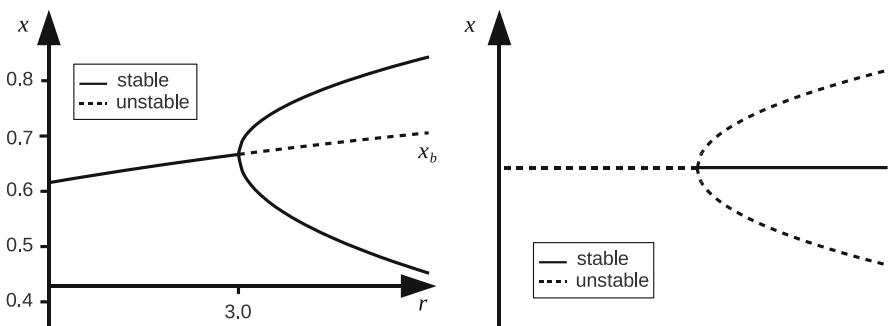


Fig. 12.5 Supercritical pitchfork bifurcation for the logistic mapping (left), schematic representation of a subcritical pitchfork bifurcation (right)

For unstable fixed points one would normally expect the value to grow exponentially. However, we have already noticed that for $r < 4$ and $0 \leq x_0 \leq 4$, the logistic mapping does not leave the interval $[0, 1]$. For $r = 3.2$ we find that, after a certain transition phase, the iterated function jumps back and forth between two values (see Fig. 12.4). Thus, there exists a cycle of period 2. This is qualitatively a clearly different behavior than for the parametric values $r < 3$. Thus we have found another place of bifurcation.

This type of bifurcation is called *supercritical pitchfork bifurcation* and is represented in Fig. 12.5 (left). One sees that the fixed point x_b becomes unstable beginning when $r = 3$ and that simultaneously a stable cycle is formed. This bifurcation type is “harmless” since even though the qualitative behavior changes given a slight change in the parameter, the quantitative behavior only changes slightly as well since the emerging cycle lies in a small neighborhood around the unstable fixed point. This looks entirely different for a *subcritical pitchfork bifurcation* that is represented schematically in Fig. 12.5 (right): The course of the curves is similar to the one for the supercritical pitchfork bifurcation, but the stability is inverted. This has serious consequences for a real system. With respect

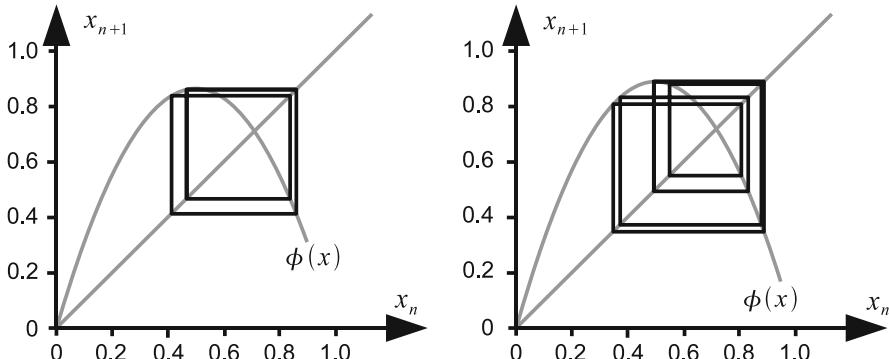


Fig. 12.6 Cycle with period 4 for $r = 3.46$ (left) and with period 8 for $r = 3.56$ (right)

to the parameter values, while there exists a stable fixed point on one side of the bifurcation place, there are neither stable fixed points nor stable cycles on the other side. In the worst case, the system “explodes” resulting in x increasing exponentially. If we think of x as the amplitude of the vibration of an airplane wing and if the parameter is the flight velocity, then it becomes apparent that such a bifurcation can have catastrophic effects in technical applications. Here it becomes apparent why the analysis of nonlinear systems is so important.

Although there still exist more types of bifurcation which will not be further discussed here, we instead choose to focus on continuing our analysis of the logistic mapping. When continuing to increase the parameter r further, the behavior likewise becomes more complex. In order to still maintain a clear overview we will only consider the asymptotic behavior. To this end, we simply start with an arbitrary initial value x_0 (it should, however, not be one of the fixed points) and observe the simulation for the first few hundreds of time steps until the transient dynamic has decayed. We thereafter continue to compute a few more additional steps and visualize them.

Figure 12.6 shows the graphical iteration of the asymptotic behavior given parameter values $r = 3.46$ and $r = 3.56$. There occur additional bifurcations which double the period of the cycle but the distance between the occurrence of additional bifurcations keeps getting shorter.

12.2.3 Transition into Chaos

If we continue to increase the value of the parameter r using relatively small steps of 0.01, we then obtain Fig. 12.7.

Here, the simulation was allowed to continue for 10^3 steps before the next 10^4 steps were used for the visualization. For the figure, the frequency that the iteration was near a point x is depicted proportionally to its dark shade. As one can see

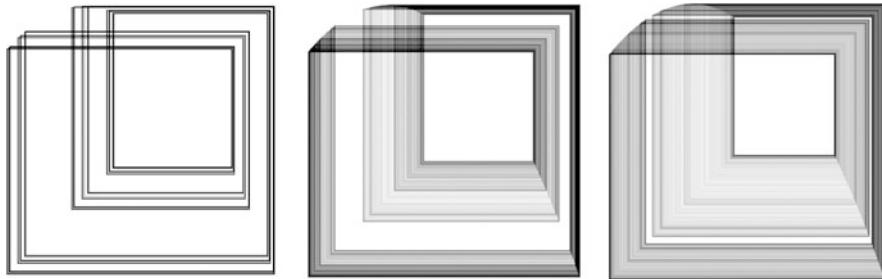


Fig. 12.7 Asymptotic dynamic of the logistic mapping (graphical iteration) for $r = 3.57$, $r = 3.58$ and $r = 3.59$

from the middle figure, when r is increased beyond $r = 3.58$, there are no stable cycles formed. However, the distribution of the points is also not random since there are certainly areas that are traversed by the iteration more frequently than others. Even though there no longer exists a classical attractor (fixed point, stable cycle) in which one eventually lands regardless of the initial state, there does exist a so-called *strange attractor*. A strange or chaotic attractor is characterized, among other things, by the fact that it restricts the domain in which the state of the system moves and as well, even from within the restricted domain some areas are traversed more frequently than others. Before we revisit strange attractors in the context of a two-dimensional system we return to our logistic mapping. At the present we realize that we cannot grasp the entire behavior spectrum of the logistic mapping through individual experiments with different parameter values. Instead, we use a so-called *bifurcation diagram*, as it has already been rudimentarily used in Fig. 12.1 and in Fig. 12.5. In the horizontal direction, we vary the parameter r and we display the corresponding asymptotic behavior (the cycle or chaotic attractor) in the vertical direction. As was done before in Fig. 12.7, those points appear darker in proportion to the frequency in which the iteration occurs in its proximity.

We therefore obtain the bifurcation diagram of the logistic mapping depicted in Fig. 12.8. At the left boundary one sees the 4-cycle for $r = 3.5$ which transitions into an 8-cycle at approximately $r = 3.55$. This in turn transitions into a 16-cycle at about $r = 3.565$. We thus observe a cascade of *period doublings* in which the distance between the doublings continues to decrease. If we would successively increase the area between $r = 3.56$ and $r = 3.57$, we would then find out that this doubling process continues and in the limit reaches a cycle with infinite period. Since the distance between doublings always decreases by the same factor (approximately 4.67), this limit value is reached for approximately 3.57. At this value, a transition to “chaotic” behavior happens. While this behavior is highly complex, one immediately sees that this behavior is not random. If one looks at the entire picture it is discovered that there is cyclic behavior at some places. Between these, there appear to be continuous spectra with chaotic behavior. A closer look, however, proves this conclusion to be wrong.

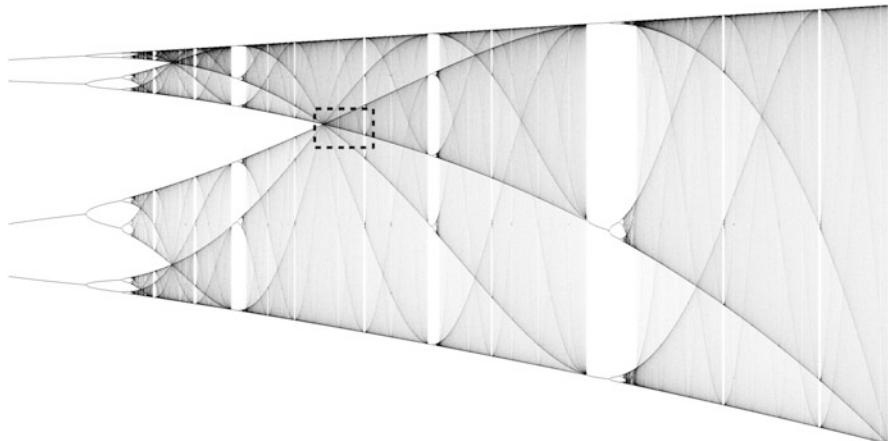


Fig. 12.8 Bifurcation diagram of the logistic mapping with $r \in [3.5, 4.0]$

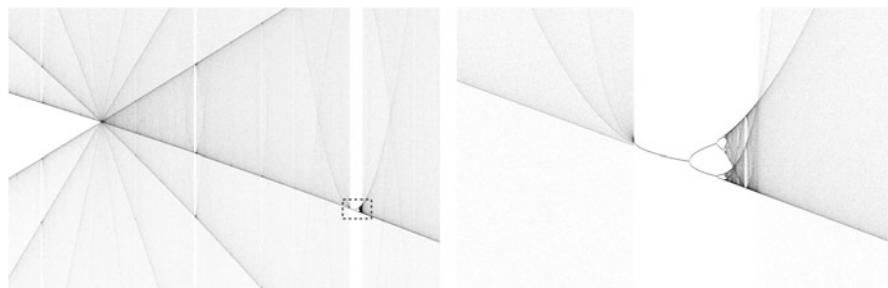


Fig. 12.9 Bifurcation diagram of the logistic mapping with $r \in [3.67, 3.71]$, $x \in [0.68, 0.76]$ (left), $r \in [3.7, 3.704]$, $x \in [0.7, 0.708]$ (right)

The framed partial area in Fig. 12.8 is seen enlarged in Fig. 12.9 (left). In this enlargement, one again sees a few vertical white lines in which there is no chaotic behavior. Further enlargement of the framed partial area in Fig. 12.9 (right) shows a very interesting effect that corresponds exactly to the original logistic mapping but reflected with respect to the horizontal axis. What looks like a fixed point in this example is of course not one, but part of a cycle with a much larger period since the picture is only a small extract of the overall illustration. Nonetheless, we have encountered here a remarkable property of chaotic systems, namely self-similarity. No matter how much we enlarge the picture, we will always find the same structures again, we find periodic regions, bifurcation cascades and strange attractors everywhere. Here, it does not at all matter which partial area we choose from the chaotic spectrum for enlargement.

12.3 Strange Attractors

We have already mentioned briefly the concept of the strange attractor. Before we go into more detail, however, we must explain the terms a little better.

Attractor

By means of the logistic mapping we have already analyzed attracting fixed points and cycles; both are particular types of attractors. The prerequisite of a fixed point was that it is mapped to itself. Additionally, in order to be attractive, a neighborhood around the fixed point must be mapped to the fixed point for $t \rightarrow \infty$. Although these two conditions apply in the same way to general attractors that include a set of arbitrarily many points, a further condition is added: There must not be a subset that as well satisfies the first two conditions—otherwise, the set of all real numbers would be an attractor for any arbitrary function. We see this holds for a fixed point since there cannot exist a subset (there is no non-empty set with less than one point). For a cycle with period k , exactly k points belong to the attractor.

For certain parameter regions, the system typically transitions from a cyclic behavior via a *bifurcation cascade* into a chaotic state. The states that such a system traverses during continued iteration jointly form the strange (chaotic or *fractal*) attractor.

Strange Attractor

In addition to the properties of attractors, the strange attractor has an additional property, namely significant sensitivity with respect to changes in the initial condition. The distance between two points originally close to one another grows significantly with time. A frequent effect of this sensitivity is the *fractal dimension* (see Sect. 12.3.1) of the attractor as well as its self-similarity.

Since the logistic mapping is a one-dimensional system, the strange attractor corresponds to the varied accumulation of points on a line. This property sufficed to recognize that the point distribution is not random. Much more cannot be seen, however, and it does not become quite clear why this behavior is called a strange attractor as well as identifying the special properties. But these strange attractors appear in all chaotic systems, and as a consequence they certainly appear in systems with several states as well. In this section, we will consider the strange attractors of a discrete mapping with two states. This was chosen because, on one hand, two-dimensional discrete mappings are still relatively simple while on the other hand, they are especially suited for graphic visualization since the two states can correspond to the two coordinates of a picture. Before we turn to the two-dimensional attractors we will define a few terms and briefly discuss fractals which play a major role for the comprehension of strange attractors.

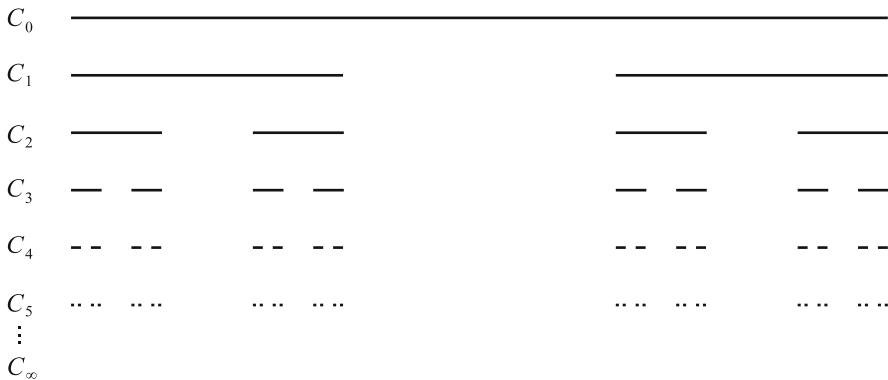


Fig. 12.10 Construction method for the Cantor set

12.3.1 Self-Similarity and Fractal Dimension

It was previously mentioned that strange attractors have a fractal dimension and, as well, are frequently self-similar. In order to obtain a visual meaning of these terms, we want to introduce a simple fractal and self-similar set.

Cantor Set

The *Cantor set* is a set that can be constructed through a recursive method. We begin with a set C_0 which consists of a line that covers the interval $[0, 1]$. From this line, the middle third is removed in order to obtain the set C_1 . From the two remaining parts, again the respective middle thirds are removed in order to obtain the set C_2 , etc.

Figure 12.10 illustrates this procedure. The actual Cantor set is C_∞ . By means of this construction principle, the term *self-similarity* becomes clear. If one takes an arbitrary segment of the line given in Fig. 12.10, this segment is then subdivided in the same way as the original line. This pattern repeats itself on arbitrarily small scales. In this case, the patterns are exactly identical since each piece of the line is refined in exactly the same way. For more complex systems, although it can also happen that the patterns occur in changed form, one still speaks of self-similarity.

A second term that can be illustrated by means of the Cantor set is the *fractal dimension*. With regards to the common understanding of dimension a line is one-dimensional, whereas a point (or even a set of points) has dimension zero. The intermediate steps for the construction of the Cantor set always consist of finitely many line segments. But the actual Cantor set no longer has line segments but instead infinitely many points. In a generalized sense, its dimension should therefore be somewhere between zero and one. Before we provide a formula for the computation of the dimension, we perform a few thought experiments. The method

Table 12.1 Relationship between the scaling factor of an object, the number of small copies that are required in order to cover the original objects, and the dimension of the object

Object	Scaling factor	Number of copies	Dimension
Line	2	2	1
Line	3	3	1
Square	2	4	2
Square	3	9	2
Cube	2	8	3
Cube	3	27	3

for constructing the Cantor set requires that at every step the set is replaced by two smaller copies each which are scaled down by a factor of 3. If instead we would scale them down by a factor of 2, we would have the original line and therefore something one-dimensional for sure. If we scale down a surface (i.e., a two-dimensional object) by the factor 2, then we need four scaled down copies in order to cover the original surface. For a cube (three-dimensional), eight copies that have been scaled down by the factor of two are required to reconstruct the original cube.

Table 12.1 gives an overview of the scaling factors and the number of required copies for the different dimensions. Here one sees that the values in the table can be easily reproduced using the following formula:

$$\text{Number of copies} = \text{scaling factor}^{\text{dimension}} .$$

This can be used to derive a formula for the general notion of dimension that no longer is restricted to integers:

$$\text{dimension} = \frac{\ln(\text{number of copies})}{\ln(\text{scaling factor})} .$$

If we apply this formula to the Cantor set, then we obtain its fractal dimension,

$$d_{\text{Cantor}} = \frac{\ln 2}{\ln 3} = 0.63093 .$$

Thus, the Cantor set has some properties that are shared with strange attractors. We next consider a simple dynamical system having a strange attractor that almost corresponds to the Cantor set. Analogous to the one-dimensional iterated function (12.1), one can also set up an iteration function for the two-dimensional case:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \Phi \left(\begin{pmatrix} x_n \\ y_n \end{pmatrix} \right) .$$

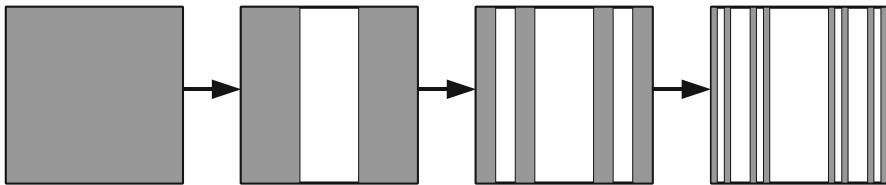


Fig. 12.11 Construction of a “two-dimensional” Cantor set

We choose the function so that the x -coordinate corresponds to the method of construction used for the Cantor set:

$$(x_{n+1}, y_{n+1})^T = \begin{cases} \left(\frac{1}{3}x_n, 2y_n\right)^T & \text{for } 0 \leq y_n < \frac{1}{2}, \\ \left(\frac{1}{3}x_n + \frac{2}{3}, 2y_n - 1\right)^T & \text{for } \frac{1}{2} \leq y_n \leq 1. \end{cases} \quad (12.4)$$

The mapping in (12.4) is defined on the unit square which is mapped to itself. Visually, one can imagine this mapping as being the unit square that is compressed in the x -direction by the factor three and stretched in the y -direction by the factor 2. The upper part that extends outside the unit square due to the stretching is cut off and placed into the open right part of the unit square.

Figure 12.11 shows the effect of this process after a few successive steps. One immediately sees that this two-dimensional mapping essentially uses the construction scheme of the Cantor set, the only difference being that a y -component has been added. An arbitrary initial point in the unit square gives a successively better approximation to the “two-dimensional” (it is not actually two-dimensional, but fractal) Cantor set. This set is hence an attractor of the mapping (12.4). In addition, it is also a strange attractor since two nearby points of the original unit square lie in completely different parts of the attractor due to the repeated stretching.

12.3.2 Hénon Mapping

Note that the mapping (12.4) is only nonlinear at $y = 0.5$ and otherwise describes a linear mapping. In contrast, we consider now a mapping that can be represented in closed form and is nonlinear on the entire domain, the *Hénon mapping*

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 1 + y_n - ax_n^2 \\ bx_n \end{pmatrix}. \quad (12.5)$$

This mapping has no particular physical meaning, but is very suitable to demonstrate the formation of strange attractors. The most frequent parameter choice for the Hénon mapping is $a = 1.4$ and $b = 0.3$ which in the following we will also use.



Fig. 12.12 Applying the Hénon mapping four times on a rectangular domain

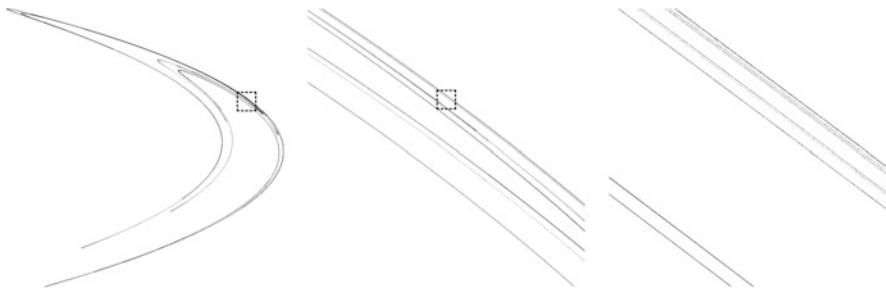


Fig. 12.13 Hénon attractor with two enlargements (factor 15 and factor 250, resp.) of the respectively framed partial areas

Despite the simplicity of the iteration rule it is no longer quite as simple to imagine what it produces. For some points, the value will explode very quickly due to the quadratic term. But there also exists a large region within which the iterates remain even after long periods of iteration.

In order to better understand what happens in the Hénon mapping, we analyze its effect on a rectangular domain. Figure 12.12 shows the initial domain and its transformation by applying the Hénon mapping four times. In each of the five pictures, the width of the entire domain is 2.6, the height is 0.8; hence the representation is somewhat distorted. Through repeated stretching and folding, the bow that is forming obtains more and more layers. This again suggests that one obtains an attractor in the limit case.

Instead of mapping the entire domain several times, we now want to see if an attractor is also formed from successive iteration of a single starting point. If one chooses the origin as the starting point and iterates the Hénon mapping several times, the same attractor is indeed formed. This one is represented in the left part of Fig. 12.13. In the framed part which is enlarged in the middle by a factor of 15 and in the right by a factor of 250, one can once more recognize the fractal structure. For this case, this is once more the result from the continuous distortion and folding of the domain by the iterated function.

12.3.3 General Two-Dimensional Quadratic Mapping

The logistic mapping is a one-dimensional quadratic mapping, while the Hénon mapping represents a two-dimensional quadratic mapping. Indeed, there exist a

multitude of other two-dimensional quadratic mappings. The general form of such a mapping is given by

$$\begin{aligned}x_{n+1} &= a_1 + a_2 x_n + a_3 y_n + a_4 x_n^2 + a_5 y_n^2 + a_6 x_n y_n , \\y_{n+1} &= b_1 + b_2 x_n + b_3 y_n + b_4 x_n^2 + b_5 y_n^2 + b_6 x_n y_n .\end{aligned}$$

Here, $a_i, i \in 1, \dots, 6$, and $b_i, i \in 1, \dots, 6$, are arbitrarily chosen parameters. In comparison, the logistic mapping had only one such parameter. By sampling this parameter, we generated bifurcation diagrams that gave a relatively good insight into the behavior of the system. But for twelve such parameters and two states, this is obviously no longer possible. To produce a picture, one can either show a variable parameter together with a state, or both states for a constant set of parameters. We do the later, since at the moment we are concerned with finding out more about strange attractors. As mentioned before, a strange attractor describes a subspace of the space spanned by the two state variables which is traversed by the iterated function after the decay of the transient dynamic. So far, however, we have yet to identify any parameter combination in which the system behaves chaotically. But we can be sure that a parameter combination exists for which chaotic behavior occurs, since both the logistic equation as well as the Hénon mapping are included as special cases in the general form of the two-dimensional quadratic function.

One obtains the logistic function by setting a_2 to r , a_4 to $-r$ and all the remaining parameters to zero. We assume that there exist additional parameter combinations in which chaotic behavior appears as well as others in which periodic behavior of fixed points appear. If, for example, one chooses all parameters to be very small, then there is again a stable fixed point at zero. For many parameter combinations the states will also go toward infinity, in particular if the parameters are greater than one. Yet there still remain sufficient parameter combinations for which chaotic behavior occurs. To this end, we will take a look at the following example:

$$\begin{aligned}x_{n+1} &= 0.6 - 0.6x_n - 0.4y_n + 0.4x_n^2 - 0.4y_n^2 + 0.3x_n y_n , \\y_{n+1} &= -1.0 + 0.5x_n - 0.2y_n + 0.2x_n^2 - 0.9y_n^2 + 0.3x_n y_n .\end{aligned}\tag{12.6}$$

As is common practice we first allow the simulation to run for a few thousand steps to ensure that the transient dynamic has decayed so that we can visualize the subsequent steps. For the logistic equation a few thousand steps had been sufficient. Here, however, we have a two-dimensional equation and obviously want to visualize it in two dimensions. For this, depending on the choice of parameters and the resolution, several million steps might be necessary.

System (12.6) then results in the strange attractor of Fig. 12.14. As before for the logistic mapping, there again exist regions that are iterated upon more frequently than others by the two-dimensional iterated function. When looking at the picture one could get the impression that curves trace through the region. This impression, however, is deceptive since after all this is a discrete system and as such cannot generate any continuous curves. In most cases, two consecutive

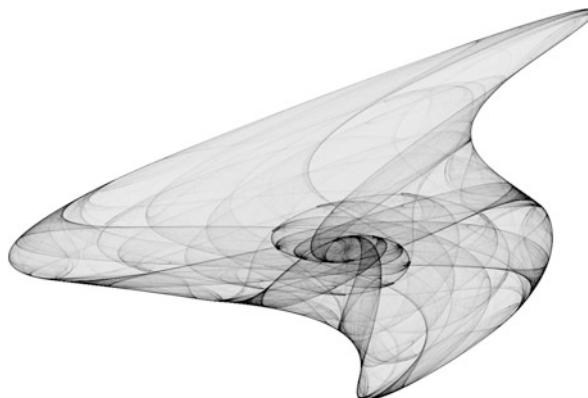


Fig. 12.14 A strange attractor of the two-dimensional quadratic mapping

function values will lie in completely different regions of the state space. If we think back to the bifurcation diagram of the logistic mapping in Fig. 12.8, in which a column of the picture corresponds to just the attractor given a particular parameter combination, we then find out that the attractors not always consist of connected domains. For example, directly after the transition into chaos the strange attractor is subdivided into many small regions. Likewise, there exist parameter combinations for the general two-dimensional quadratic mapping for which the strange attractor is divided into many regions.

In Fig. 12.15, the corresponding strange attractors are represented for six additional parameter combinations. The attractor at the bottom right exhibits just this effect in which the attractor consists of many small regions. The iterated function values continuously jump back and forth between these regions. The other five graphs show an entirely different, but yet fascinating behavior. Since a total of 12 parameters are available that can be freely chosen, there exists an enormous number of possible combinations and as a consequence many different attractors.

At the present it is hardly meaningful to set up bifurcation diagrams to represent the behavior with respect to changes in the parameters since we can only vary one parameter and would have to fix the other eleven. In this example, we had not assigned any explicit physical meaning to the parameters and we therefore could not make any meaningful choice for the parameters. Thus, in the next section we will turn to a mechanical system in which all parameters have a concrete physical foundation.

12.4 Chaotic Behavior of a Driven Pendulum

In the previous sections we became acquainted with the different properties of nonlinear mappings as well as the methods employed for the analysis of these properties by means of iterated functions. We now turn to a continuous example and



Fig. 12.15 Additional strange attractors of the two-dimensional quadratic mapping

work to find out how well the findings from the discrete examples can be transferred. We consider a simple mechanical system, namely a *damped pendulum*, that is driven by a force. In the next section we will see that this system can be modeled by three state variables. It has already been mentioned that only continuous systems with at least three state variables display chaotic behavior. In this as well as in many other books that deal with chaos, only systems that include friction are considered. Frictionless chaotic systems behave differently and for more details regarding this the reader is referred to [46].

12.4.1 Model of the Pendulum

Our pendulum is a rod in which one end is fixed to a freely rotating joint around which the rod can rotate freely—i.e., the full 360° . The damping of the pendulum is proportional to the rotational velocity. Furthermore, the gravitational force acts on the pendulum. Let the torque that is applied to the pendulum be proportional to a sinusoidal function. The system consists of three state variables, the phase, ψ , of the torque, the angle, ϕ , of the pendulum and the angular velocity, ω , of the pendulum. These three quantities are sufficient to describe the state of the system at any given time. We do not go into details of the derivation of the model here. The derivation for a similar example can be found in Sect. 11.2, and for further information on the modeling of dynamical systems (also chaotic ones), we refer

the reader to [7]. After the modeling process, one obtains the following ordinary differential equations (already in dimensionless form), which describe the system:

$$\frac{d\psi}{dt} = \omega_M , \quad (12.7)$$

$$\frac{d\phi}{dt} = \omega , \quad (12.8)$$

$$\frac{d\omega}{dt} = -D\omega - \sin\phi + A \cos\psi . \quad (12.9)$$

Equation (12.7) is very simple; the change of the phase, ψ , corresponds to the frequency, ω_M , of the driving sinusoidal moment. Likewise, the change of the angle, ϕ , obviously corresponds to the angular velocity, ω (see (12.8)). The last differential equation (12.9) describes the change of the angular velocity ω . The first term on the right hand side represents the damping that is dependent on the angular velocity. The faster the pendulum rotates, the more it is slowed down. The second term corresponds to the influence of the gravitational force while the last term is the driving moment. Thus the system has three free parameters, the frequency, ω_M , of the driving moment, a damping constant, D , and the amplitude, A , of the moment. When considering the equations, it stands out that the state variable, ψ , depends only on the frequency, ω_M , and not on the other two variables. This of course results from the moment always acting on the pendulum from the exterior which occurs independently of the current pendulum's state. We will see later that this greatly simplifies the analyses for us since we can concentrate on the two remaining states, ϕ and ω . For all the pictures, we will always only use these two states.

12.4.2 Discretization

A continuous system—or the respective model of such a system—must always be *discretized* for a simulation. The discretization method chosen obviously depends on the system itself as well as the desired accuracy and the available computing capacities. Since we are only concerned with an exemplarily demonstration of chaotic behavior by means of a continuous system, we do not devote much of our attention to the discretization. Given the differential equations (12.7) through (12.9), we have three continuous equations that describe the system. Since these are first order ordinary differential equations, their discretization is relatively simple anyhow. With *Euler's method*, we discretize the three equations separately. For this method, the derivative is approximated by the difference quotient,

$$\frac{dx}{dt} \doteq \frac{x_{n+1} - x_n}{\delta t} .$$

If we apply this method to (12.7) through (12.9), we obtain the discretized equations given by,

$$\begin{aligned}\psi_{n+1} &= \psi_n + \delta t \cdot \omega_M , \\ \phi_{n+1} &= \phi_n + \delta t \cdot \omega , \\ \omega_{n+1} &= \omega_n + \delta t \cdot (-D \cdot \omega_n - \sin \phi_n + A \cdot \cos \psi_n) .\end{aligned}$$

This discretization introduced an additional parameter, namely the time step width δt . Intuitively, we should obviously choose this value to be as small as possible in order to keep the discretization error small, in particular since with Euler's method we have chosen a method known for its relatively large discretization error. Additionally, chaotic systems are very sensitive to small changes in the initial conditions and thus in turn to errors introduced during the simulation. However, opposing this intuition is the need to compute many time steps. For the different strange attractors of the two-dimensional quadratic mapping, the simulations were required to already run for several millions of iterations. As we will see later, we require many multiples of this for continuous systems rendering the computing time to be no longer negligible if time steps are chosen too fine. Here, for all analyses we will define the time step length to be $\delta t = 10^{-3}$. By using better discretization techniques such as, e.g., Heun or Runge–Kutta, one could further increase the step length and therefore save additional computing time.

12.4.3 Cycles and Attractors

The model of the driven pendulum contains three parameters. If one wants to analyze the system for a realistic application, then one naturally has to place certain restrictions on the parameters, or it could be that some of the parameters may also be fixed by the configuration of the system. For the first experiments, we specify a fixed damping and frequency of the driving torque and choose $D = 0.7$ and $\omega_M = 0.8$. We want to analyze the effect of different amplitude magnitudes on the behavior of the pendulum. To this end, we run the simulation of the system with different values for A for a sufficient number of iterations to ensure that the transient dynamic has decayed and then plot the remaining asymptotic dynamic. To this end, we project the trajectory onto the ϕ/ω plane.

Figure 12.16 displays the dynamic behavior for $A = 1.8$, $A = 1.87$ and $A = 1.89$. When $A = 1.8$, the result is a cycle while for $A = 1.87$, this cycle is doubled and when $A = 1.89$ it is quadrupled. This behavior is reminiscent of the period doubling cascade for the logistic mapping with the difference that the logistic mapping jumped back and forth between 2^i points. But here we have a continuous system which makes the comparability somewhat difficult. For the discrete system, it was easy to determine the period of a cycle by simply counting the points. For the continuous system, however, the period is no longer the number of iterations but

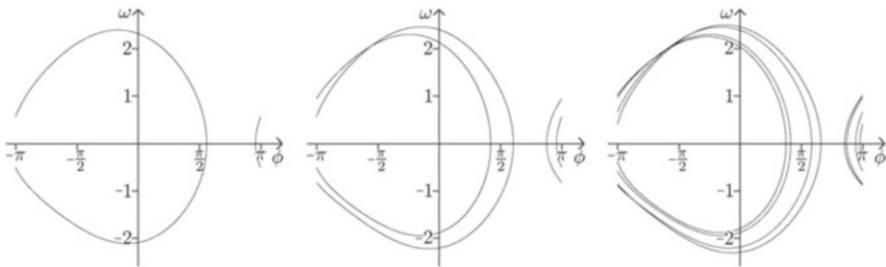


Fig. 12.16 Asymptotic dynamic of the driven pendulum for $A = 1.8$ (left), $A = 1.87$ (middle) and $A = 1.89$ (right)

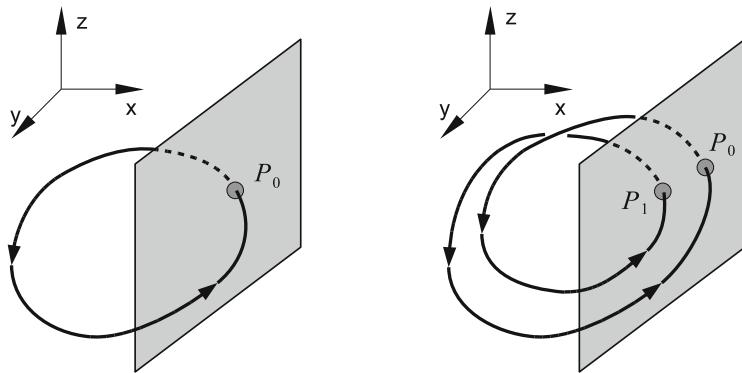


Fig. 12.17 Poincaré section of a cycle with one and two “rounds”

actually the duration of a cycle. Graphically, one could of course say that the period corresponds to the number of “rounds”, i.e., in the above three cases, 1, 2 and 4. But this is hard to grasp mathematically, especially for more complicated courses of trajectories. Also, the direct construction of bifurcation diagrams for the analysis of the behavior for an entire parameter interval is not possible for the continuous system. Both problems, however, can be solved through a reduction of dimension as will be described in the following.

Poincaré Section

The *Poincaré section* is a method to reduce an n -dimensional continuous system to an $(n-1)$ -dimensional discrete system. For three-dimensional problems, a plane is typically chosen that is preferably perpendicular to the trajectory and thus is intersected by it.

Provided the trajectory is cyclic, the plane will be crossed from the one side at least once. This is illustrated in the left part of Fig. 12.17 in which it can be seen that there exists a closed cycle that successively crosses through the point P_0 on the

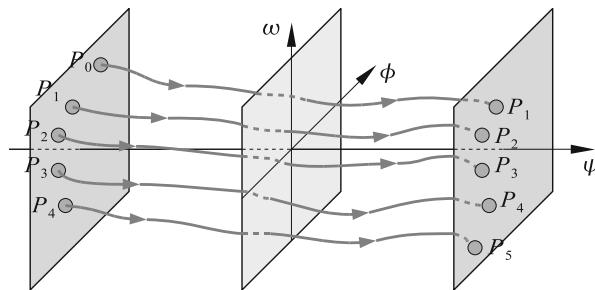


Fig. 12.18 Schematic representation of a Poincaré section for the driven pendulum; due to the periodicity of the driving mechanism, each of the displayed planes is crossed by the trajectory again and again; as soon as the trajectory leaves the region on the right, it re-enters at the same place on the left

plane over and over again. Strictly speaking, there should be yet a second point in this figure in which the plane is crossed from the other side, but we only consider the points in which the plane is crossed from the front. If the trajectory now requires two “rounds” before the cycle repeats itself, then this plane is crossed at two points provided it has been placed correctly.

For our pendulum, the values for ψ and ϕ are both restricted to the interval $[-\pi, \pi]$. As soon as one of the states leaves this interval on one side, it is re-entered on the other side. This becomes immediately apparent if one imagines the motion of the pendulum. For $\phi = 0$, the pendulum hangs down vertically. Both for $\phi = \pi$ as well as for $\phi = -\pi$, the pendulum stands vertically upward, and therefore the system is in the same state.

For the construction of a Poincaré section of our system we must specify the exact position of the plane. Reviewing equation (12.7), one sees that ψ grows linearly as well. Since ψ is bounded and since the value is always reset when leaving the interval, the same value for ψ is reached over and over again in exactly the same time periods. For the plane we therefore chose it to be arbitrary given a constant ψ , or, in other words, a plane that is spanned by ϕ and ω . This also implies that the plane is always crossed from the same side, as can be seen in Fig. 12.18.

For the trajectories in Fig. 12.16, such a Poincaré section is of course unspectacular since the plane is hit only one to four times. For higher values of A , we once more expect chaotic behavior due to the expected bifurcation cascade. This actually happens. The Poincaré section for $A = 2.04$ can be seen in Fig. 12.19. Similar to the discrete Hénon attractor given in Fig. 12.13, there exists a strange attractor here as well.

Instead of the analysis of individual parameter combinations, we have used bifurcation diagrams for the logistic mapping in order to analyze the system behavior over entire parameter intervals. But the logistic mapping is one-dimensional and as such the diagram was very easily constructed. By the Poincaré section, we have already reduced the dimension from three down to two, but we must lose another

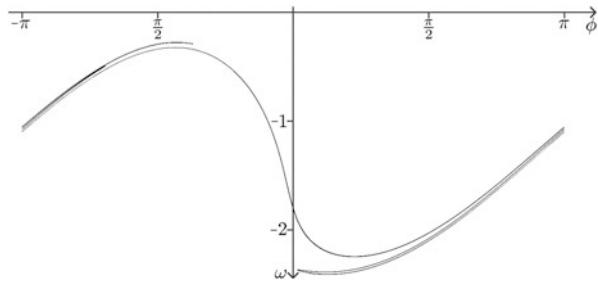


Fig. 12.19 Poincaré section of the strange attractor for $A = 2.04$

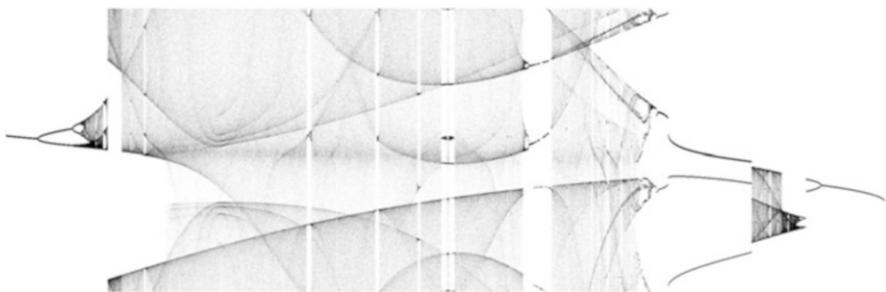


Fig. 12.20 Bifurcation diagram for $A \in [1.8, 2.8]$, $D = 0.7$ and $\omega_M = 0.8$

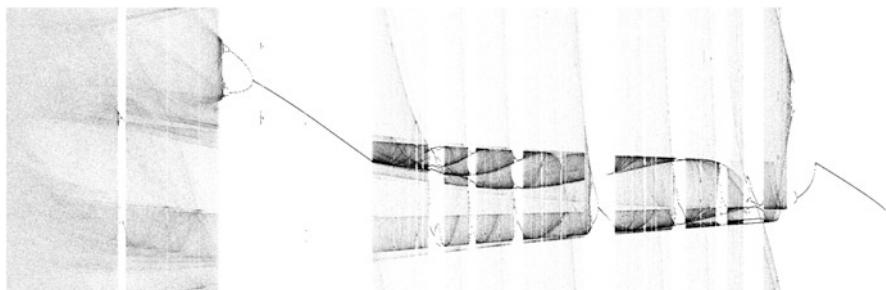


Fig. 12.21 Bifurcation diagram for $A = 1.0$, $D \in [0.0, 0.7]$ and $\omega_M = 0.5$

dimension for the bifurcation diagram. The easiest possibility to do this is the projection to one of the two remaining coordinate axes. Since one is normally interested only in qualitative rather than quantitative statements, this projection typically does not restrict the informative value by much. A cycle with period k corresponds to just k points in the plane in the Poincaré section. Provided that these points are not arranged vertically or horizontally with respect to each other, there still remain k points after a projection onto one of the two axes.

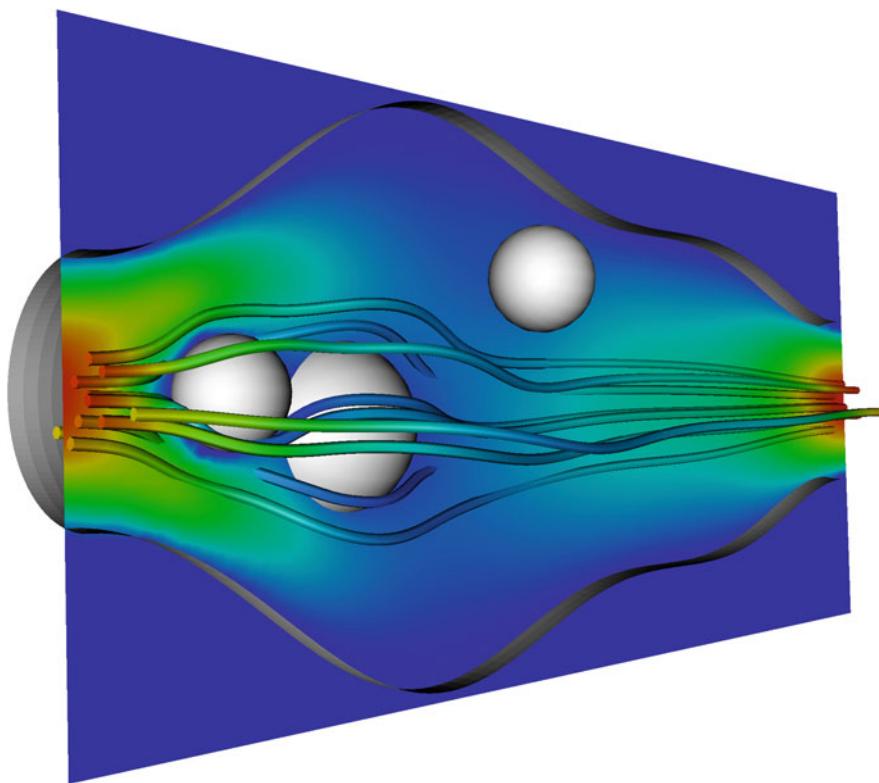
For the previous experiments, we have specified the damping and the frequency of the driving torque as $D = 0.7$ and $\omega_M = 0.8$. We keep this and construct a

bifurcation diagram with variable amplitude between $A = 1.8$ and $A = 2.8$. For every “column” of the diagram we construct a Poincaré section and project it onto the ϕ -axis. This axis has the advantage that the values are restricted to the interval $[-\pi, \pi]$ and we therefore do not have to spend time considering the interval that is to be represented. We thus obtain the bifurcation diagram in Fig. 12.20.

Needless to say, there already exist many different combination possibilities for three parameters in which the system responds with an entirely different behavior. Figure 12.21, for example, shows another bifurcation diagram for a fixed frequency and amplitude of the driving force but a variable damping. In order to draw meaningful insight from the bifurcation diagrams, one has to think quite a bit more beforehand about the parameter combinations to be analyzed than what we have done here. In this chapter, we were mainly concerned with illustrating what one understands under chaos and to show the characteristic properties of such behavior—in particular in the context of modeling and for the simulation of simple technical systems such as the pendulum that was considered.

Part IV

Physics in the Computer: Take-Off Toward Number Crunching



The last part of our book is devoted to the applications of modeling and simulation having strong physical backgrounds. Here, the subtitle “Take-off toward number crunching” conveys the message “PC farewell!”. Even though we have also dealt with challenging simulation methods in the previous chapters, the

computational effort mostly remained manageable. This now changes in a fundamental way since we enter the domain of *high performance computing (HPC)*. In the years 2000, 2005 and 2007, for example, the renowned *Gordon–Bell-Award*, the Noble prize for simulation so to speak, has been awarded for computations in molecular dynamics. Hence, as our first concern we will take a look at *molecular dynamics*: a particle approach that, even though it is still based on ordinary differential equations (where there exist surprising analogies between the interdependencies between molecules and those of planets), it yet brings the resolution of the space into the scene. We then continue with *heat transfer*, a classic in numerous books on mathematical modeling. Though the continuum mechanical model, now fully resolved in space, is relatively simple, computations in thermodynamics may require much more time.

Next—of course—*fluid dynamics*. This field has also been highly successful with respect to the accumulation of Gordon–Bell prizes, e.g. in 1987, 1995, 1996, 1999 and 2002. In particular when turbulence enters the game, then numerical fluid dynamics specialists are popular or dreaded customers at the computing center (depending on available capacities . . .).

What surely at first sight must appear a little unusual is our last application, which originates from computer graphics. But truly only at first sight since one only needs to think of such diverse and pertinent Hollywood productions as Jurassic Park, Toy Story, Cars, or of the flood of computer games: Wobbling dinosaur bellies are modeled and computed, the elegant motion of curtains in the wind is modeled and computed, terrain models are modeled, the flight across them is simulated.

Not to mention illumination—without illumination models, it would in fact remain pitch-black on the monitor. And finally, we want to concern ourselves in more detail with the global illumination of scenes that must be represented. With this an obviously important target group of this book should be served, those who so far feel that they got the shorter end of the stick with all the mathematical foundations and technical-scientific applications: the students of computer science. It can hardly be disputed that this type of physics in the computer is computer science in its purest form.

Chapter 13

Molecular Dynamics

Molecular dynamics deals with the simulation of substances on the *molecular* or *atomic* level. This means that at least every molecule, if not even every atom, is considered separately in the simulation domain. Thus it is immediately apparent that the domains in question must be very small. One mole of a substance contains approximately $6 \cdot 10^{23}$ particles. For an ideal gas, one mole corresponds to 22.4 liters while for solids the volume of this quantity is obviously very much smaller. Furthermore, because the simulation over meaningful time periods requires a large number of simulation time steps, the simulation over large domains is ruled out from the beginning. In this respect, one will probably never (at least not within a time frame from which the authors of this book will live to see it) simulate a wind channel completely on the molecular level—and this would probably be overkill, even if it was technically possible. Yet, even under the given constraints, there exist a multitude of application fields in which a molecular examination is not only meaningful, but even necessary. For example, it is necessary in *biological* or *medical* applications in order to analyze the functioning of proteins or other *macro molecules*, or in nanotechnology. A molecular dynamics simulation can also be sometimes useful in fields where simulations on the continuous level are normally employed, for example, in fluid flow simulations. Here, some phenomena occur which cannot be resolved on the continuous level, for example the precise behavior on the border between two different substances. Furthermore, molecular dynamics is increasingly gaining traction in *material sciences* and *process engineering*. In the latter, the focus lies on the interplay between the different *aggregate states*, i.e., evaporation, vaporization and distillation processes. Of the many different application areas this one is the area that we will concentrate on in this chapter. An important characteristic is that typically only very small molecules—in comparison, for example, to proteins—will be considered, but of those there are plenty.

Based on the laws of physics, we will derive models for the *interaction* between atoms. We will convert these models into a differential equation which we then discretize for the simulation. To this end, previous knowledge from the areas of analysis and the numerical treatment of ordinary differential equations (see Chap. 2)

will prove helpful. After the discretization, we will deal with the construction of a simulation domain and the associated necessary parameters and boundary conditions. Finally, we will go into methods for the efficient implementation and parallelization, i.e., the distribution of computational effort to many processors.

13.1 Modeling of Molecules and Interactions

We have already made clear how molecular dynamics essentially deals with the simulation of the motion of a multitude of molecules. The state of a molecule is completely determined by the position of the molecule in space and its velocity. Since the velocity is the first derivative of the position the change of position can therefore be computed with the help of the velocity. In turn, the change of the velocity, requires its first derivative and this gives the acceleration of the molecule. By Newton's second law, the acceleration a of a body with mass m is proportional to the force F acting on the body so that it holds that

$$F = m \cdot a . \quad (13.1)$$

The essential challenge in molecular dynamics is to compute the acting force for every molecule. We will see in the following that this force depends on the surrounding molecules. Because molecules apply forces to each other we therefore speak of *interactions* between the molecules. In the following, we will for simplicity only consider atoms and no longer arbitrary molecules. Much of this carries over very easily to some small molecules, but in general—and in particular for macro molecules—this is not the case.

However, we first must decide how close to reality our models must be before atoms and the interactions between them can be modeled. For the highest accuracy possible we would have to use the laws of quantum mechanics when setting up the models. But by doing this, however, just the simulation of a single atom would already be so complex that the simulation of millions of atoms would remain pure utopia. Probably the simplest model would be to picture individual atoms as spheres that only interact when they collide (i.e., they would act as billiard balls; such a model is called a *hard-sphere model*). One realizes, however, that one can replicate only very few and very particular scenarios with such a simple model. We therefore want to deal more closely with the physical interactions between atoms in the following sections.

13.1.1 Fundamental Physical Forces

In physics there exist four fundamental forces. All forces that occur anywhere can be traced back to these four forces:

- *gravitational force*: While this force is very weak, its effect decreases very slowly for increasing distance which is why it is also called a *long-range* force. It always

acts attractively and is responsible for the orbits of the celestial bodies and also for the gravitational force that we are exposed to on Earth (Earth and humans mutually attract each other). Strictly speaking, this force acts between all bodies, i.e., also between any arbitrary pair of atoms in the universe. For short range (in the magnitude order of molecules), this force is negligible compared to other forces and hence does not play a role in the simulations that we will consider.

- *electromagnetic force*: The electromagnetic force, for example, acts between electrically charged particles (particles could be loaded as positive or negative or even be neutral). Equal charges repel each other and opposite charges attract each other. This force is significantly stronger than the gravitational force and is also long-range. However, for most bodies, negative and positive charges cancel each other. Thus, the attracting and repelling electromagnetic forces that such bodies apply to each other are mostly cancelled out—at least for large distances. Most of the forces that can be observed in everyday life are based on an electromagnetic force. It is also responsible for the force between atoms. We will look into this in more detail in the course of this chapter.
- *strong nuclear force*: The strong nuclear force is the most powerful among the four forces. Its range, however, is so short that it essentially only acts within the atom's nucleus. For the simulation considered within the content of this chapter, we are interested only in forces between atoms and as such we will not go into further detail with respect to the strong nuclear force.
- *weak nuclear force*: also a force of very short range, which among other things is responsible for nuclear decay processes within the atom's nucleus. It also does not play any role in the following.

13.1.2 Potentials for Uncharged Atoms

We first have to briefly explain the term *potential*. Potentials in the context of particle simulations describe the “capability” of the particles to apply forces to each other. In our case we will only use *pair potentials* which are potentials that depend only on the distance between two particles. However, for some substances more complex potentials are necessary which, for example, depend on more than two particles and possibly not only on the distance.

In order to determine the force \mathbf{F} that two particles i and j apply to each other from the potential between these two particles, we compute the *negative gradient* of the potential U ,

$$\mathbf{F}_{ij} = -\nabla U(r_{ij}),$$

where r_{ij} denotes the distance between the particles i and j . From the gradient we obtain a force vector, i.e., both the direction as well as the magnitude of the force. The direction always corresponds to the distance vector between the two molecules. What we are concerned with in this section is therefore only the

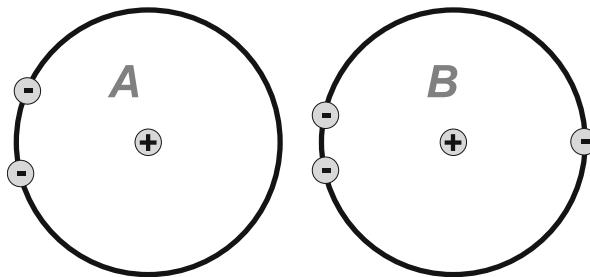


Fig. 13.1 Schematic representation for the origination of the van-der-Waals force through temporary dipoles

magnitude of the force. In the following we thus consider the force as a scalar quantity (F) which corresponds to the negative derivative of the potential.

In the previous section we have established that only the electromagnetic force is relevant to us. Because of this we will neglect the other three forces in the subsequent modeling. Furthermore, only uncharged atoms will be considered in the following. These are—at least from a sufficiently simplified perspective—built out of a nucleus which is positively charged due to the protons and a negatively charged electron shell. The numbers of protons and electrons are equal, the electrons are on average uniformly distributed around the nucleus and as a result the atom is overall neutral. For a large distance between atoms, hardly any forces are active.

Attractive Forces

As soon as the atoms come closer to each other, this is no longer the case. The so-called *van-der-Waals force* is active by which atoms *attract* each other. Through the use of Fig. 13.1, we attempt a brief explanation regarding the origin of this force and how it is based on the *electromagnetic* force. The electrons move relatively freely in the electron shell of an atom. As a result, the negative charge is not distributed uniformly in the shell and varies continuously. Let the atom **B** in Fig. 13.1 now be an atom in which the charge is concentrated slightly stronger on the “left side” so that a so-called *temporary dipole* develops. If in this situation atom **B** is in the proximity of atom **A**, then it has an effect on its electrons. Since negative charges repel each other, the electrons in atom **A** are pushed to the left. Now both atoms are temporary dipoles, whereby the positive pole of atom **A** is facing the negative pole of atom **B**. This leads to an overall attraction between the atoms that becomes increasingly stronger as their proximity lessens. Mathematically, the underlying *van-der-Waals potential* can be described as follows:

$$U_1(r_{ij}) = -4\epsilon \left(\frac{\sigma}{r_{ij}} \right)^6 .$$

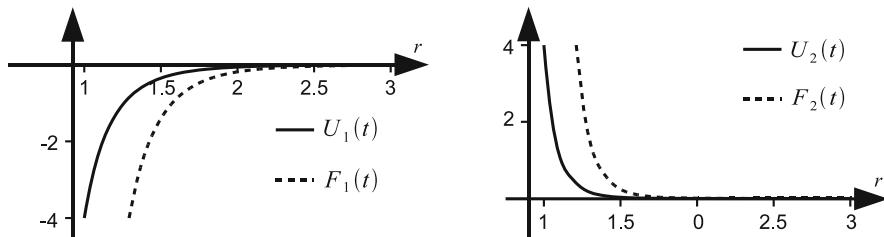


Fig. 13.2 Attractive van-der-Waals force (left) and Pauli repulsion (right); respective force and corresponding potential

The corresponding force

$$F_1(r_{ij}) = 24\epsilon\sigma^6 \left(\frac{1}{r_{ij}}\right)^7$$

is the negative derivative of this potential. The parameters ϵ and σ describe the energy and size of the atom, respectively. One sees in Fig. 13.2 (left), that for large distances r_{ij} , the force tends toward zero due to the large exponent present in the denominator.

Repulsive Forces

This attractive force is complemented by a *repulsive* force since otherwise all atoms would collapse into each other. This repulsive force grows with decreasing proximity between one another. Visually, one can picture the cause as the overlapping of electron clouds. The electrons of the two atoms are then so close that they repel each other and thus the two atoms as well; this effect is called a *Pauli repulsion*. The corresponding potential can be modeled in various ways. Important for the repulsive force here is that it acts opposite to the attractive force and that it is stronger in absolute value for smaller distances and weaker for larger distances compared to the attractive force.

The potential

$$U_2(r_{ij}) = 4\epsilon \left(\frac{\sigma}{r_{ij}}\right)^{12}$$

for example, satisfies these requirements. This potential represents a good approximation to the actual physical effect and has the advantage that it can be computed relatively quickly. In Fig. 13.2 (right) one can see that it decays faster than the attractive potential.

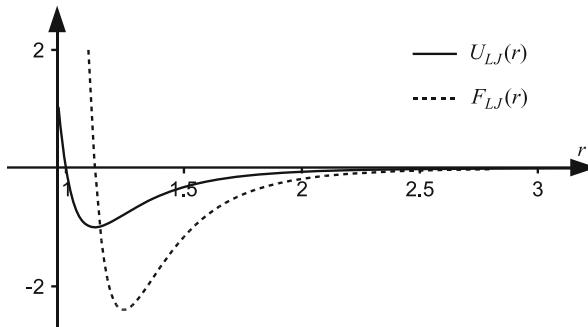


Fig. 13.3 Combination of the attractive and repulsive potentials for the Lennard–Jones potential

Lennard–Jones Potential

Combining the attractive and repulsive potentials introduced above we obtain the so-called *Lennard–Jones potential*,

$$U_{\text{LJ}}(r_{ij}) = 4\epsilon \left(\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^6 \right). \quad (13.2)$$

From the equation one should clearly see the advantage of the selected repulsive potential—it can be computed from the attractive part of the potential by squaring it once. In Fig. 13.3, the potential and the resulting force are shown with their dependence on the distance between the two atoms. The acting (attractive) forces are very weak when the distance between atoms is very large. These forces first become stronger as the atoms get closer. However, at some specific distance apart the attractive and the repulsive forces counterbalance each other. This distance is exactly the minimal point of the potential or the root of the force.

For the interaction between atoms we will exclusively use the Lennard–Jones potential. Strictly speaking, we therefore do not compute the interactions between the atoms but interactions between atom models. This means we will no longer speak of atoms but instead of *Lennard–Jones centers*. Such a Lennard–Jones center can either represent an individual atom or in some cases also a small group of atoms. It is parametrized by ϵ (energy) and σ (size).

Mixing Rules

The Lennard–Jones potential in (13.2) requires the two parameters ϵ and σ of the Lennard–Jones center. This does not cause a problem yet if only equal atoms appear in a simulation. However, if the objective is to compute the force between two atoms of different types, then the question arises regarding the choice of parameters to be used for the evaluation of the equation. In this case, an averaged value is

computed for every possible combination of atoms (and therefore of Lennard–Jones centers). In general, it cannot be determined how accurately this averaging has to be computed, and it strongly depends on the involved substances, thus we do not go into further detail here.

Cut-off Radius

All the potentials considered previously decay at a rate of at least r^{-6} where r denotes the distance between two particles. This means that the force between two particles quickly declines with increasing distance. In this case, one speaks of a *short-range* potential. In fact, in order to compute the force that acts on an individual particle i , one would have to compute the force between this particle and every other one in the simulation (or in the entire universe). Consequently, for a simulation with N particles $\mathcal{O}(N)$ operations must be performed in order to compute the force that acts on a single particle and $\mathcal{O}(N^2)$ operations are therefore required to compute the force on all particles.

Since the force quickly declines with increasing distance, it is sufficient to consider only those particles that are near in proximity to particle i . The distance defining which particles are considered near is called *cut-off radius* (r_c). Typically, this distance corresponds to several σ , where σ is the size parameter of the atoms in (13.2). Hence, the number of particles that are within the cut-off radius and for which a force computation has to be performed is bounded by a constant. Hence, the force on one particle can be computed with $\mathcal{O}(1)$ operations and thus the force on all particles with $\mathcal{O}(N)$ operations. By using the cut-off radius, the cost for the force computation in one time step of the simulation can be reduced from $\mathcal{O}(N^2)$ down to $\mathcal{O}(N)$. Remaining now is the question how large the cut-off radius can in fact be chosen. The smaller it is, the less the computational effort, but the results also become less accurate. In the end, the precise choice of the cut-off radius depends on how accurate the results need to be for the particular application. Normally, the value chosen lies between 2.5σ and 5σ . It is also common to estimate the influence of those particles that have been ignored through the cut-off and to add this as a correction term to the force.

13.1.3 Computation of the Force Acting on an Atom

In the previous section we were concerned with the way to describe the interaction between uncharged atoms. Except for the simulation of noble gases, hardly any additional scenarios are realizable given the current considerations. For this, one would need to take a much closer look at the modeling. For example, one would still need a modeling of *dipoles* as well as including a consideration of molecules instead of only looking at atoms, etc. However, the subsequent procedure in the remainder of this chapter is hardly affected by model extensions. It is only a little

more descriptive with this simple model. This is why the present model is sufficient for our purposes. The equations for the potential and the corresponding force (again as a vector quantity) from which we will subsequently build upon are given by,

$$U_{\text{LJ},r_c}(r_{ij}) = \begin{cases} 4(r_{ij}^{-12} - r_{ij}^{-6}) & \text{for } r_{ij} \leq r_c, \\ 0 & \text{for } r_{ij} > r_c, \end{cases} \quad (13.3a)$$

$$\mathbf{F}_{ij,r_c}(\mathbf{r}_{ij}) = \begin{cases} 24(2r_{ij}^{-12} - r_{ij}^{-6}) \frac{\mathbf{r}_{ij}}{r_{ij}^2} & \text{for } r_{ij} \leq r_c, \\ 0 & \text{for } r_{ij} > r_c. \end{cases} \quad (13.3b)$$

With (13.3b), the force between two particles i and j can be computed. In order to compute the entire force \mathbf{F}_i that acts on a particle P_i , all pairwise forces for which P_i is one of the involved particles have to be summed:

$$\mathbf{F}_i = \sum_j \mathbf{F}_{ij}.$$

13.2 Equations of Motion and Their Solutions

We now have obtained the knowledge regarding the interaction of forces that act on molecules and how they can be computed with the help of a physical model. In this section we discuss the manner in which the motion of the molecules that result from these forces can be computed. To this end, one first sets up a differential equation for the position of the particles and solves it with various numerical methods. The methods differ on the one hand by properties that are independent of the particular application such as, for example, *accuracy* and *time reversibility*. But on the other hand the methods differ in the properties that are important in particular in molecular dynamics simulations, for example *energy conservation* and *symplectic properties*. We will briefly mention only the advantages and disadvantages of the methods without going into further detail regarding the respective properties.

13.2.1 Equations of Motion

Our objective is the simulation of the temporal development of a substance on the molecular level. To this end, an initial state must be prescribed in which the position and velocity are specified for all molecules. From this, with the help of the material considered in the previous section, the force on every individual molecule can be computed. Given the forces, the acceleration of every molecule can in turn be computed with (13.1) and transferred to vector representation,

$$\mathbf{a} = \frac{\mathbf{F}}{m}.$$

We can thus compute the acceleration for all molecules at a given time instance. Furthermore, we know that the acceleration is the second temporal derivative of the position:

$$\mathbf{a} = \ddot{\mathbf{r}}.$$

Hence, for every molecule we obtain an ordinary differential equation of second order so that altogether we have a system of N differential equations of second order whose solution describes the course of the molecules. However, when there are three or more particles this system of differential equations can no longer be solved analytically. We therefore must use a numerical method for its solution. In principle, the procedure looks as follows:

- Given are the positions and velocities of all particles at a certain time t .
- From this, the forces and hence accelerations of all particles at time t are calculated.
- From these three quantities, the positions and velocities are determined for a later time $t + \delta t$.
- The loop now begins again at time $t + \delta t$.

We therefore need a method that can compute the new positions and velocities. In the following, we will derive two different methods and discuss their advantages and disadvantages.

13.2.2 Euler's Method

A very simple method which has already appeared multiple times in this book is *Euler's method* (see Sect. 2.4.5). It can be derived with the help of a Taylor expansion by neglecting (i.e., truncating) terms of higher order. For the position \mathbf{r} at time $(t + \delta t)$, it therefore follows that

$$\mathbf{r}(t + \delta t) \doteq \mathbf{r}(t) + \delta t \mathbf{v}(t).$$

Likewise, a formula for the velocity \mathbf{v} at time $(t + \delta t)$ can be computed:

$$\mathbf{v}(t + \delta t) \doteq \mathbf{v}(t) + \delta t \mathbf{a}(t).$$

This method is very easy to derive as well as to implement, but it also has some disadvantages. Besides the low level of accuracy it has other properties which make its use in the field of molecular dynamics simulation difficult. It is therefore suitable only for small illustrative examples, but not for realistic simulations.

13.2.3 Velocity-Störmer-Verlet

There exist quite a number of better discretization techniques with entirely different properties. Here, we want to take a closer look at one of the most frequently used methods, the so-called *Velocity-Störmer-Verlet method*. Once again, it originates from a Taylor expansion, but this time it includes terms to the second order:

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t \mathbf{v}(t) + \frac{\delta t^2}{2} \mathbf{a}(t). \quad (13.4)$$

This formula requires the position, velocity and acceleration at time t . The first two are always known while the computation of the acceleration at time t requires only the position at the same time instance. This also implies that from the now obtained new position at time $(t + \delta t)$, the acceleration at time $(t + \delta t)$ can be immediately computed as well. All these values can therefore be used for the computation of the velocity at time $(t + \delta t)$.

The derivation of the formula for the velocity is slightly more complex. First, the velocity at time $(t + \frac{\delta t}{2})$ is computed by an explicit Euler step using the velocity and acceleration at time t . Thereafter, an implicit Euler step is used to compute the velocity at time $(t + \delta t)$ using the velocity at time $(t + \frac{\delta t}{2})$ and the acceleration at time $(t + \delta t)$:

$$\begin{aligned} \mathbf{v}(t + \frac{\delta t}{2}) &= \mathbf{v}(t) + \frac{\delta t}{2} \mathbf{a}(t), \\ \mathbf{v}(t + \delta t) &= \mathbf{v}(t + \frac{\delta t}{2}) + \frac{\delta t}{2} \mathbf{a}(t + \delta t). \end{aligned}$$

By substituting these two formulas into each other one obtains the desired formula for the velocity at time $(t + \delta t)$,

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \frac{\delta t}{2} (\mathbf{a}(t) + \mathbf{a}(t + \delta t)), \quad (13.5)$$

which only depends on values that have already been computed. The two formulas (13.4) and (13.5) together form the Velocity-Störmer-Verlet method. The procedure of this method is illustrated in Fig. 13.4 in four steps. The dark-gray colored blocks denote the data that is required for the respective computation while the light-gray blocks denote the values to be computed. From this one immediately sees the disadvantage of the method in its current form. The computation of $\mathbf{v}(t + \delta t)$ requires the acceleration at two different time instances. It is obvious that both acceleration values have to be stored as well. But it would be more efficient if for each molecule only one acceleration and thus one force had to be stored. This can be achieved through a clever choice for ordering the computations. Before the third step in Fig. 13.4, the computation of (13.5) lacks only $\mathbf{a}(t + \delta t)$. The other values are required only for this computation. One can therefore partially evaluate the equation

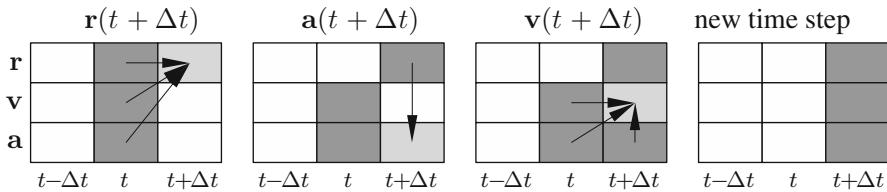


Fig. 13.4 Individual steps and corresponding occupied storage locations for the Velocity-Störmer-Verlet method

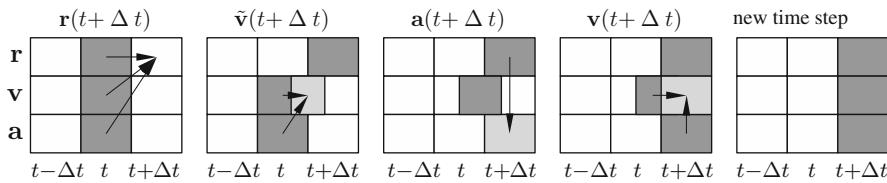


Fig. 13.5 Velocity-Störmer-Verlet method with efficient (wrt. storage use) ordering of computations

already by first adding $\frac{\delta t}{2} \mathbf{a}(t)$ to $\mathbf{v}(t)$. This procedure is illustrated in the second step of Fig. 13.5. After this intermediate step, the acceleration at time step t is no longer required and the corresponding storage location can be overwritten in the third step. In the fourth step, the remainder of (13.5) is evaluated, and therefore all values for time $(t + \delta t)$ are known.

13.2.4 Remarks

There exist a number of additional methods for the discretization of the equations of motion in molecular dynamics. The choice for the one used certainly depends on what exactly is simulated, for example, whether innermolecular forces have to be taken into consideration as well as what insight is to be gained from the simulation. While for many other technological problems the accuracy of the discretization method is very important, it typically plays only a subordinate role in molecular dynamics simulations. One is not interested in the paths taken by individual particles but in statements about the entire system, i.e., pressure, temperature, ... Even if one would want to determine the individual paths, this would not be possible over longer time periods even with the most accurate methods. This is due to the fact that systems based on molecular dynamics are *chaotic* (see Chap. 12). That is, a very small error, for example a round-off error, has a very large effect over time, and the simulated trajectory departs exponentially from the actual one. Since the systems considered in this chapter are composed of atoms with short-range forces, the Velocity-Störmer-Verlet method is a good choice in any case.

13.3 Simulation Domain

In the previous section, we dealt exclusively with the modeling of physical interactions between molecules. Here, we concentrated on few substances to be simulated and attempted to represent the interactions between these substances well. We have therefore considered the physical basics for the simulation, but have not devoted any thoughts on the purpose of our simulation. The force acting on every single molecule is computed, yielding the trajectories of every molecule in the simulation domain. As previously mentioned, we are not interested in the trajectories of individual molecules. Much more important is the development of macroscopic quantities which do not make statements on individual molecules but rather on the entity of the simulated molecules. Examples of such quantities are the *temperature*, the *pressure* or the *potential energy* of the simulated domain. In order to obtain usable results for these quantities, we must first specify the exact framework for the conditions of the simulation. Here, different parameters can be taken into consideration, for example the size of the simulation domain, the number of particles, the initial configuration of the particles, the density and the pressure in the simulation domain, ... Some of the parameters mentioned depend on each other. For example, the pressure is not independent of the density. In order to obtain realistic results for a simulation, some parameters have to be fixed at the beginning of the simulation and must not be permitted to change during the course of the simulation. There are several different possibilities for specifying the framework conditions. One of those we will consider more closely: the so-called *NVT-Ensemble*.

13.3.1 NVT Ensemble

The three letters stand for the number of particles (N), the volume of the simulation domain (V) and the temperature of the simulated substance (T). For the simulation of an NVT ensemble, these three quantities must remain constant over the entire simulation. As a simulation domain, we specify at the beginning a cubic domain which naturally determines the volume. Furthermore we have to specify how many particles are initially in this cube and give their exact locations. This number depends on the density of the substance to be simulated. The initial position is for example chosen so that all particles are distributed uniformly. Since the particles move while the simulation domain remains the same, some particles will go beyond the border of the domain in the course of the simulation. In return, obviously particles must also enter into the domain from the outside. In Sect. 13.3.2, we will see how this problem is solved, but first of all let us simply mention here that within the domain the number of particles remains constant.

Our objective is to perform the simulation of a substance for a given temperature. It is also apparent that the temperature must be constant during the simulation. To this end, we must contemplate how to actually measure or set the temperature at a

molecular dynamics simulation. The temperature of a substance T depends on the velocity v_i of all atoms. The higher the velocity, the warmer the substance. Each atom has a *kinetic energy* which is determined by its velocity. By adding up these energies one obtains the overall kinetic energy

$$E_{\text{kin}} = \frac{1}{2} \sum_i m_i v_i^2$$

of the simulated domain. From the kinetic energy, one can compute the temperature with the help of the Boltzmann constant k_B ($1.38 \cdot 10^{-23} \text{ J/K}$) as

$$T = \frac{2}{3Nk_B} E_{\text{kin}}.$$

Here, N denotes the number of atoms in the simulation domain. Thus, the temperature is easily determined directly from the velocity of the atoms with the help of the two formulas given above.

Normally, the actual temperature T_{act} fluctuates during a simulation. However, we have set our goal to simulate a substance for a given temperature. In order to reach this goal, a *thermostat* is used: This is the convention used to name methods that bring a substance to a target temperature T_{tar} . This can, for example, be reached in a very simple way through the multiplication of all velocities by $\sqrt{T_{\text{tar}}/T_{\text{act}}}$.

We have now reached the point where we can set up the initial configuration of a simulation having a domain with N particles, volume V and temperature T . The volume remains constant automatically, the number of particles remains constant as well through a suitable boundary treatment and the temperature is kept at the specified temperature through a thermostat.

13.3.2 Boundary Conditions

The selected simulation domain is only a part of the real world. Normally, there would be particles leaving the domain as well as entering into the domain. Furthermore, in reality there are also particles outside of the simulation domain which interact with those within the domain. We thus simulate a system that is not absolutely closed and we must therefore think about what to do at the interface between the simulation domain and the surrounding domain. Here, the most diverse approaches are conceivable. For *reflecting boundary conditions*, the particles that reach the boundary are bounced back, whereas for *periodic boundary conditions* they are removed from the domain in order to be re-entered on the opposite side. As already indicated by the name, particles can flow out of and into the domain for outflow or inflow boundary conditions.

Since we employ an NVT ensemble, the boundary condition must ensure that the number of particles remains constant, i.e., only periodic or reflecting boundaries

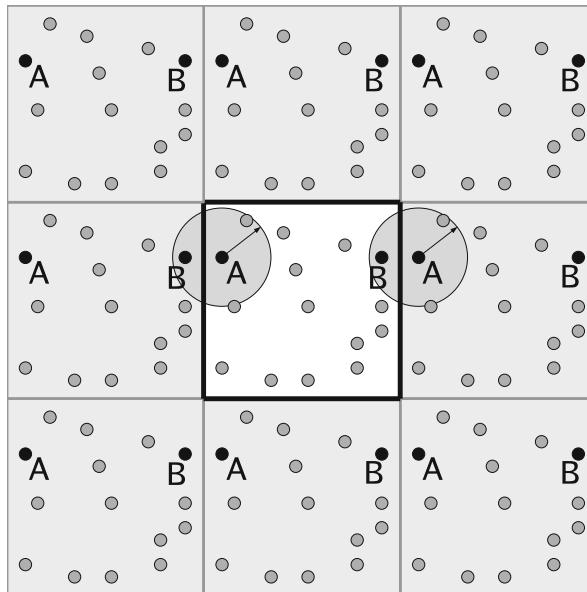


Fig. 13.6 Periodic boundary conditions for a square simulation domain

come into consideration. For the reflecting boundary conditions, the boundary acts like a wall that cannot be penetrated. But our simulation domain is only a small part of a larger domain, i.e., we assume that the same conditions apply outside of the domain as they do inside. The selected boundary condition should exploit this knowledge. This is possible with the periodic boundary conditions. The middle square in Fig. 13.6 corresponds to our simulation domain, all the other squares are virtual copies of it. If the force onto particle **A** in the middle domain is now to be computed, then among other things particle **B** in the left domain needs to be considered since it lies within the cut-off radius. Since the left domain is just a virtual copy of the actual simulation domain, we simply go back and use particle **B** in the middle domain. Alternatively, one could establish a boundary layer in which actual copies of the particles are stored. Particles that leave the domain on one side are inserted again on the other side. This way, the simulation is in a way made to believe that the same substance continues indefinitely outside the actual domain.

13.4 Implementation

In the previous sections, the different aspects pertaining to the set up of the model have been considered. Starting with the physical model (Sect. 13.1), we have concerned ourselves with a mathematical model and the corresponding algorithms

in Sect. 13.2. Finally, in the previous section we have discussed general framework conditions such as, for example, the size of the domain and the boundary conditions. Now we can address the implementation aspects which are necessary for an efficient simulation.

13.4.1 Linked-cells Data Structure

As already described in Sect. 13.1, we only consider particles which have a distance of at most r_c in light of the quickly decaying forces as the distance increases. It is therefore a challenge to find these neighboring particles. One could simply measure the distance to all neighbors and then compute the forces for those with small distance. But this way, one would have a constant complexity for the force computation per particle while the distance computations would have a linear (with respect to the overall number of particles) complexity per particle and therefore require an overall quadratic complexity. In order to prevent this, only $\mathcal{O}(1)$ particles must be considered for finding the neighborhood of a particle.

Domain Decomposition

A simple possibility for this is the so-called *linked-cells method*. Here, a regular grid is used to subdivide the simulation domain into small cells, the resulting cell grid has n rows and m columns. In the simplest case these cells are square, and the edge length corresponds exactly to the cut-off radius r_c . As an example, the cells are stored in a continuing array in which the rows of the subdivided domain follow one after another. A domain of width $6r_c$ and height $5r_c$ yields the subdivision into 5×6 cells with their respective indexing as represented in Fig. 13.7.

For each time step of the simulation all the particles are first sorted into their corresponding cells. To this end, the indices of the corresponding cells have to be computed for the positions of all particles. This is possible with a constant number of operations per particle for the regular grid chosen for our subdivision. After the sorting, all cells and the particles contained within are traversed. Since the length of a cell corresponds to the cut-off radius, the neighboring particles can only be in the eight directly neighboring cells. The problem here is that cells at the boundary do not have eight neighboring cells. However, as explained in Sect. 13.3.2, we want to simulate periodic boundary conditions. This means that every particle, and therefore every cell as well, is surrounded by neighbors. At the boundary of the domain the particles at the opposite side are chosen as neighbors according to the periodic boundary conditions. In order to still have easy access to the neighboring cells at the boundary one simply puts another layer of cells around the domain which point to the respective opposite cells. This obviously changes the indexing of the original cells.

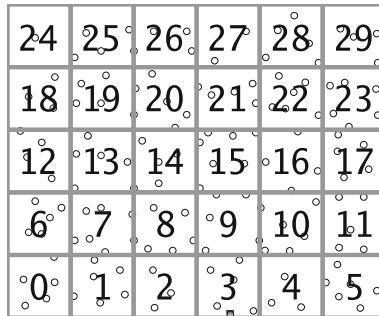


Fig. 13.7 Indexing of the cells in the linked-cells method

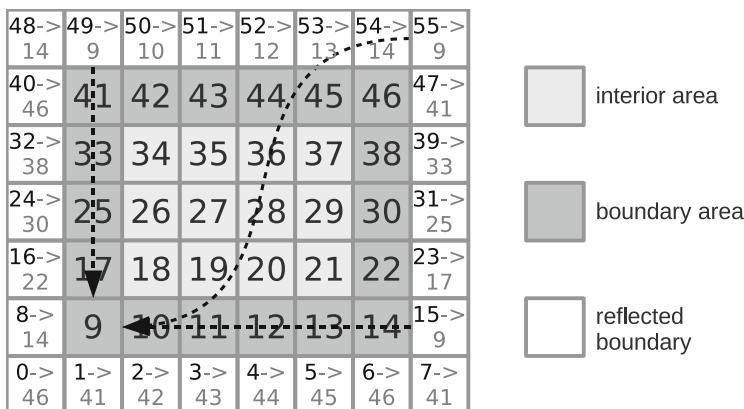


Fig. 13.8 Indexing of the cells in the linked-cells-method when using periodic boundary conditions

Cell 0 in Fig. 13.7 obtains the index 9 in Fig. 13.8. If one wants to determine the neighboring cell to the left of this cell, then this is initially cell 8, but this points to cell 14 on the opposite side of the domain. For the corners, the respective cell that lies diagonally across has to be used. Thus, the cells 15, 49 and 55 all point to cell 9. For the computation of the forces that act on the particles, all cells of the actual domain (boundary and interior area in Fig. 13.8) are first processed in an outer loop. For every cell, an additional loop runs through all corresponding particles. And for each of these particles, the distance to the particles in the neighboring cells is measured. If this is not larger than r_c , the force between these two particles is then computed and stored.

Access to Neighbors

We now turn to the actual purpose of the linked-cells method that is the identification of neighboring particles, i.e., of two particles each which have at most a distance r_c .

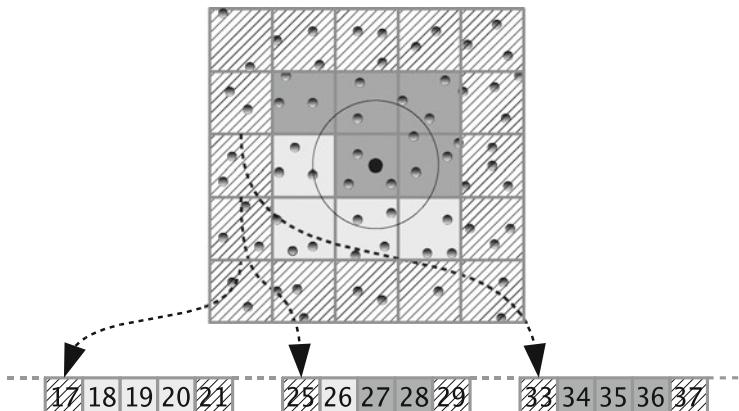


Fig. 13.9 Linked-cells data structure with corresponding array

As already mentioned, the neighboring particles of a certain particle can only be in its own cell and in the surrounding eight cells. Due to Newton's third law one does not even need these eight cells since every pair of particles, and therefore every pair of cells, has to be considered only once. Thus, when determining the relevant neighbors of a cell, we will take only those cells that come after the current cell, i.e., cells whose indices are higher.

Figure 13.9 illustrates the process for finding the neighbors of cell 27 that is given in Fig. 13.8. The middle cell, number 27, is the one whose particles will have their forces computed, the light and dark gray cells are the neighboring cells with lower and higher indices, respectively. Ruled cells cannot contain neighboring particles. Since, as explained above, only cells with a higher index have to be searched for neighbors—and obviously cell 27 itself—only the five dark gray cells are relevant. Due to the chosen storage, their indices are very easily computed. For the right neighbor, one is simply added. The index difference to the neighbor above corresponds to the number of cells in a row, in this case eight. For the cells diagonally above, from this number one has to in turn add or subtract one.

13.5 Parallelization

So far, we have considered the essential steps needed to perform a rudimentary simulation of molecules. If, however, we want to perform it over a longer time and for large particle numbers, we will most likely fail or at least wait for the result for a very long time. For example, in order to simulate millions of particles over 10^5 time steps—which is a reasonable order of magnitude for many applications—the work must be done by many processors in parallel. In order to facilitate the development of parallel programs, there exist libraries such as *MPI* (Message Passing Interface),

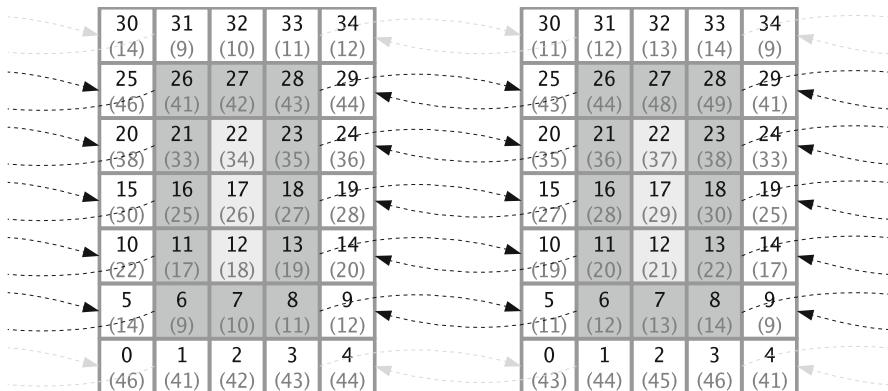


Fig. 13.10 Division of the domain from Fig. 13.8 onto two processes. The upper number is the respective new local cell index, the number in parentheses is the index of the corresponding cell in the original domain

for example, which facilitate the communication between processes. The core work consists of thinking about which processes need to exchange what information and when. We will roughly analyse this for our scenario without going any further into the actual implementation using MPI.

We must first come up with a strategy for distributing the computing work to different processors. A widely used strategy is the spatial division of the domain into equal sized parts which are assigned to the individual processes. The division of the domain from Fig. 13.8 into two parts of equal size is illustrated in Fig. 13.10. Through this subdivision, a new boundary layer results at the interfaces. The molecules outside of the intersecting edge (white cells) must be retrieved from the respective other process from its boundary layer (dark gray cells). At the left boundary of the left domain and at the right boundary of the right domain there still hold periodic boundary conditions. The data to be exchanged are on the respective other process. As one can recognize easily from the diagram, both processes therefore have a left and a right neighbor. This holds in the same way for the subdivision into more subdomains. For the individual process it is irrelevant whether it is located at the boundary of the original domain or not, every process exchanges particles at the boundary with its neighbor. In Fig. 13.10 one can see this for the two-dimensional case.

It is obvious that every process obtains an upper and a lower neighbor in addition to the left and right neighbors. As can be seen in the enlargement (see Fig. 13.11), there must also be an exchange of molecules diagonally. This can either be reached through direct communication between the respective processes, or, for example, by first pushing the particles from the right lower corner of a domain to the right neighbor so that this sends it further down. Since in general it makes sense to communicate with as few of the other processes as possible, the latter method is more reasonable since it reduces the number of neighboring processes from eight down to four.

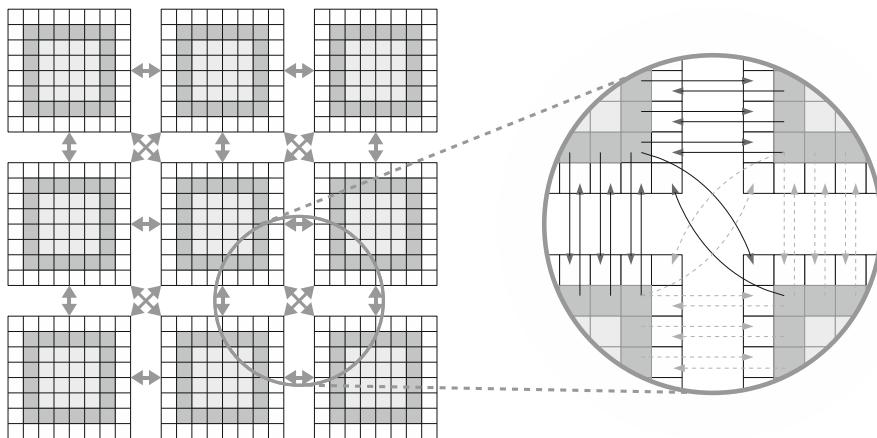


Fig. 13.11 Subdivision of a domain in both spatial directions onto several processes. Data exchange takes place with the direct neighbors as well as diagonally

13.6 Outlook

Especially in the last few years, very diverse applications based on molecular dynamics have become more and more important, in particular in high performance computing. This can, for example, also be seen from the fact that numerous such applications were represented among the finalists of the Gordon–Bell prize, one of the most important prizes in the area of high performance computing. In continuum-based simulations such as fluid flow simulation, for example, in principle even large domains can be simulated with low computational effort by a corresponding coarse choice of the discretization grid. Although this obviously has an effect on the accuracy and significance of the simulation, the simulation is nevertheless possible. For some applications of particle based simulations, however, one is not presented for the possibility to save computing time through a coarser resolution. A protein, for example, consists of a certain number of atoms which all have to be taken into consideration in the simulation. This is one reason for the fact that particle based simulations become interesting for ever new applications through the increasing computing power of modern computers. It is also the reason that we have placed the little excursion into parallelization into this chapter.

It was the attempt of this chapter to cover a large breadth, starting from physical considerations and the mathematical modeling via numerical methods up to the implementation and parallelization aspects. There is definitely a lot more to say about more complex potentials, small and very large molecules, long-range forces and much more. A good overview is given in [3, 22, 33]. The basic theory with respect to modeling can be found in [29, 48] contains many code examples. Last, a basic introduction with special focus on the numerical treatment is found in [31].

Chapter 14

Heat Transfer

The molecular dynamics simulations discussed in Chap. 13 required the use of *ordinary differential equations (ODE)* when describing particle trajectories. For these, only one independent variable, in this case the time, was needed. But there also exist a very large range of physical problem settings in which the modeling process is assisted by *partial differential equations (PDE)* in a way that is somewhere in the area between obvious, suitable or necessary. An example is *structural mechanics*, which among other things considers the deformation of structures under the influence of forces. Such analyses are relevant in entirely different scenarios—from the construction of bridges to the construction of micro-electromechanical sensors and actuators (MEMS). Another important example is *fluid dynamics* which we will be concerned with in the next chapter. From the perspective of the simulation, the problems which can be modeled with PDEs are interesting and challenging, in particular since the most modern methods and computers are typically indispensable for their efficient numerical solution.

But first things first: A problem setting that is relatively easy to derive is described by the *heat equation* and is be considered in this chapter as a prototype of a phenomena that is based on *balancing processes*. In *thermodynamics* as well as in many other application areas it is important to be able to make statements pertaining to the propagation of heat. Frequently, the objective is to either divert heat as quickly as possible (e.g., at air conditioning systems and cooling devices) or to deliver it as quickly and loss-free as possible (e.g., at cooktops). However, other applications can be concerned with analyzing the distribution of heat in a body in order to recognize places that, for example, are threatened with the possibility of overheating.

The following will first briefly introduce the physical basics of heat transfer. The choice of dimension is also considered since ultimately the complexity of the simulation depends very much on the number of independent variables. In this case, we only consider the temperature, therefore the complexity depends on the number of dimensions considered in the model (space and time). From here, we naturally consider the simulation of the (steady) heat equation. This includes on the one hand

the discretization while on the other the solution of the resulting linear system of equations.

Given the tools discussed in Chap. 2 this requires not only analysis but the comprehensive parts from the section on numerical analysis. For obvious reasons, our treatment is once again relatively terse. A more comprehensive reference for the physical description as well as for the discretization is found in [57].

14.1 Derivation of the Heat Equation

The derivation of partial differential equations for the description of physical processes can be very complicated. Before modeling, one should first restrict the application area of interest as much as possible: The more scenarios and effects that are to be included into the model, the more complex the model obviously becomes. The point of departure are the basic physical laws, often in form of *conservation laws*. In this chapter, these are considerations concerning the amount and transfer of heat. Typically, such laws include differential operators (as a simple example from classical mechanics one can think of the relationship between the distance traveled and velocity or acceleration) in which the mathematical treatment leads to systems of differential equations that often are connected with algebraic equations that, e.g., describe constraints (maximal displacement, etc.).

For the diffusion of heat in an object, hereforth called the *body*, the desired quantity is the *temperature* $T(\mathbf{x}; t)$ that is dependent upon the location \mathbf{x} and the time t . The time and space coordinates must be treated separately in the following derivation: For the operators such as ∇ (*gradient*), div (*divergence*) and $\Delta = \operatorname{div} \cdot \nabla$ (*Laplace operator*), the partial derivatives only apply to the spatial coordinates.

For the derivation of the heat equation we consider the amount of heat stored in the respective body from which we will obtain a conservation equation. We assume that this amount of heat per unit volume is proportional to the temperature whereby the proportionality factor is the product of the *density* ϱ of the body and the *specific heat capacity* c (a material property of the substance which describes how much energy is required in order to increase the temperature by a certain amount in a certain amount of substance). One then obtains the amount of heat Q that is stored per unit volume V (reference volume) of the body by integration over the reference volume:

$$Q = \int_V c \varrho T(\mathbf{x}; t) \, d\mathbf{x}. \quad (14.1)$$

In the following we assume that the body is *homogeneous* so that ϱ and c are (positive real) constants; hence, they will not play any particular role in the following.

The temperature distribution—and thus the amount of heat—changes over time. We thus derive an equation that describes the heat transfer across the surface

∂V of our reference volume. The heat transport is driven through the balancing processes (here due to temperature differences within the body) that we assume to be proportional to the *normal derivative* $\frac{\partial T}{\partial n}$ at the surface, i.e., the scalar product of the gradient ∇T and the outward directed normal vector \mathbf{n} (a positive normal derivative therefore describes the case of heat transfer into the volume V). Here, the proportionality factor is provided by the so-called *thermal conductivity* $k > 0$ (in a substance that conducts heat well, e.g., in a metal, k is large, whereas k is only slightly greater than zero for a substance that insulates heat). We assume the thermal conductivity to be constant within the entire body and in addition *isotropic*, i.e., it is not dependant on direction. We now obtain the inflow of heat into the volume V across its surface and therefore the temporal change in the amount of heat (here, other impacts such as chemical reactions within V , for example, are excluded) as a *surface integral* over the temperature gradient (k is only a constant), which can be transformed into a *volume integral* through use of the *divergence theorem*:

$$\frac{dQ}{dt} = \int_{\partial V} k \nabla T(\mathbf{x}; t) \cdot \vec{dS} = \int_V k \Delta T(\mathbf{x}; t) \, d\mathbf{x}. \quad (14.2)$$

Differentiation of (14.1) with respect to t and substitution into (14.2) yields

$$\int_V c\varrho \frac{\partial T}{\partial t}(\mathbf{x}; t) \, d\mathbf{x} = \int_V k \Delta T(\mathbf{x}; t) \, d\mathbf{x}.$$

Since this equation is satisfied for every arbitrary reference volume V , the two integrands must coincide:

$$c\varrho \frac{\partial T(\mathbf{x}; t)}{\partial t} = k \Delta T(\mathbf{x}; t).$$

With $\kappa = k/(c\varrho)$ one immediately obtains the *heat equation*

$$\frac{\partial T(\mathbf{x}; t)}{\partial t} = \kappa \Delta T(\mathbf{x}; t), \quad (14.3)$$

that, for the case of three spatial coordinates x , y and z , can be written explicitly as

$$\frac{\partial T(x, y, z; t)}{\partial t} = \kappa \left(\frac{\partial^2 T(x, y, z; t)}{\partial x^2} + \frac{\partial^2 T(x, y, z; t)}{\partial y^2} + \frac{\partial^2 T(x, y, z; t)}{\partial z^2} \right).$$

This is a PDE of *parabolic type* (see Sect. 2.4.6), for which *boundary-initial value problems* are typical: In this case, the temperature distribution $T(\mathbf{x}; t_0)$ is given at the start t_0 of the considered time interval as well as conditions for the surface of the overall body for all $t > t_0$ (e. g., *Dirichlet boundary conditions* that specify the temperature, or *Neumann boundary conditions* that specify the heat flow). The differential equation itself states how the temperature distribution changes under these conditions.

Without restricting boundary conditions solutions for this equation can sometimes be determined without the need for numerical calculations, i.e., analytically or “with paper and pencil”. For example, one easily verifies that for arbitrary $v_x, v_y, v_z \in \mathbb{N}$, the functions

$$T(x, y, z; t) = e^{-\kappa(v_x^2 + v_y^2 + v_z^2)t} \sin(v_x x) \sin(v_y y) \sin(v_z z)$$

satisfy the heat equation.

The previous representation of the heat equation depends on three spatial variables and one temporal variable. In the following, however, we will only consider one or two spatial dimensions. The procedure does not differ from the three-dimensional case in principle. However, the two-dimensional case is much easier to conceive.

In addition, one is often not interested in the temporal development of the temperature distribution but only in a distribution reached in the limit *steady state* as $t \rightarrow \infty$. Such a distribution is characterized (in analogy to the equilibrium points for ODEs) by the fact that no more temporal change takes place. It therefore holds that $T_t = 0$ so that the solution satisfies the *steady heat equation*

$$\kappa \Delta T(\mathbf{x}) = 0 . \quad (14.4)$$

This is a differential equation of *elliptic type*; there no longer are initial values but only boundary conditions. One therefore solves a classical *boundary value problem*. In order to keep the effort of the subsequent discretization low, we will restrict ourselves here to the steady state problem.

14.1.1 Number of Dimensions

The number of spatial dimensions that have to be considered in a model and its subsequent simulation depends essentially on four things: on the shape of the object, on the boundary conditions, on the desired accuracy, and on the available computing capacities. Even though all items of daily use are three-dimensional objects, sometimes a simplified representation is entirely sufficient. The heat diffusion in a rod can be represented with only one spatial dimension while a piece of sheet metal requires two. For this, it is important that the sheet is insulated relatively well on the top and bottom surfaces since otherwise heat is exchanged in this third, not modeled dimension.

The required accuracy cannot be determined through the simulated object alone. As an example we consider the simulation of the heat diffusion in a cooking pot as it is illustrated schematically in Fig. 14.1

We make the simplifying assumption that the heat diffusion in the pot behaves according to the formula derived above. The main heat input comes from below via

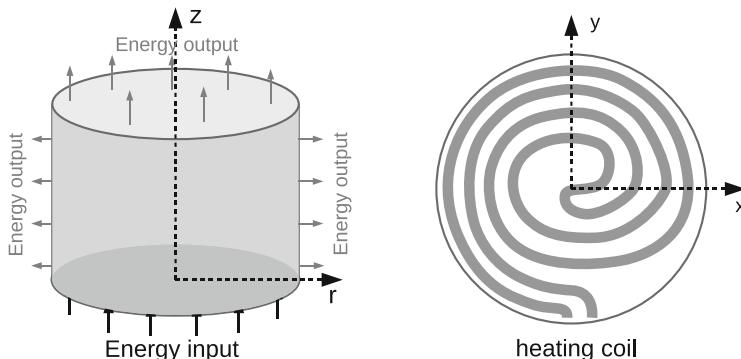


Fig. 14.1 Schematic representation of a cooking pot (left) and cooktop (right)

the cooktop and diffuses in the z -direction. For the amateur cook who needs only a rough idea of the temperature at a given height, it suffices to consider this dimension alone. Admittedly, the pot loses heat through the exterior sides. For the manufacturer of the pot it would make sense to consider this in the modeling regardless. Thus, the radius is used as an additional coordinate. As illustrated schematically on the right side in Fig. 14.1, the heat distribution is also inhomogeneous on the cooktop itself. For a manufacturer of cooktops it would be important to take such effects into consideration in the modeling of heat diffusion in a pot as well via the angle as a third spatial coordinate.

14.2 Discretization

In this section we assume that the model equations are given so that their discretization can now be discussed. Here, we will use *finite differences* which have already appeared in several places of this book. We will review the discretization since we are also interested in the efficient solution of the resulting linear systems of equations.

In the derivation of the heat equation we considered the case involving the *steady state* (14.4). The solution of this steady state equation requires numerical techniques different from those for the solution of the transient heat equation. In the *transient* case, both the space as well as the time must be discretized whereas in the steady state case only the spatial coordinates are discretized. For example, if one wants to analyse the distribution of heat in a material that is connected to a heat source on one side, one may consider this either as a steady state problem or as a transient problem. One could simulate the material for a certain period of time in order to determine the heat distribution at the end of this time period. In this case, the time has to be discretized and the problem becomes transient. Since one is not interested in the development over time, but rather in the final result, one may instead solve a

steady state problem in which one is looking for the heat distribution which satisfies the heat equation subject to the given initial conditions.

These two approaches impose quite different requirements to the respective numerical methods. Without numerics we cannot proceed in either case since the equations, together with their boundary and initial conditions, hardly ever have analytic solutions. We restrict our attention in the following to the steady state heat equation. In Sect. 2.4.6, we have introduced several discretization techniques for PDEs and for our problem at hand we select the method of finite differences. We briefly review the discretization through the construction of both the *3-point-stencil* (one spatial dimension) and *5-point-stencil* (two spatial dimensions), resp.

14.2.1 3-Point-Stencil

The underlying principle of finite differences is based on the use of difference quotients to approximate derivatives. As an example, for the first derivative one can approximate a tangent line using a secant line as illustrated in Fig. 14.2.

When approximating first derivatives, one distinguishes between *forward difference quotients*,

$$T'(x_0) \doteq \frac{T(x_0 + h) - T(x_0)}{h} , \quad (14.5)$$

in which the current and following grid points are used, and the *backward difference quotients*,

$$T'(x_0) \doteq \frac{T(x_0) - T(x_0 - h)}{h} , \quad (14.6)$$

in which the current and previous grid points are used instead. In order to discretize the second derivatives appearing in the heat equation, one can once more construct difference quotients from (14.5) and (14.6). This procedure leads to

$$\begin{aligned} T''(x_0) &\doteq \frac{\frac{T(x_0+h)-T(x_0)}{h} - \frac{T(x_0)-T(x_0-h)}{h}}{h} \\ &= \frac{T(x_0 + h) - 2T(x_0) + T(x_0 - h)}{h^2} . \end{aligned} \quad (14.7)$$

To perform numerical computations we need to determine a suitable value for the *grid width* h . On the one hand, this value will depend on the required computational accuracy (it is hoped that the closer the grid points are to each other, the higher the accuracy—at least asymptotically). On the other hand, however, we have to keep in mind the computing resources available. In the case of the one-dimensional

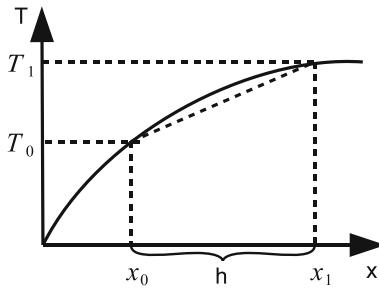


Fig. 14.2 Approximation of the derivative by the secant

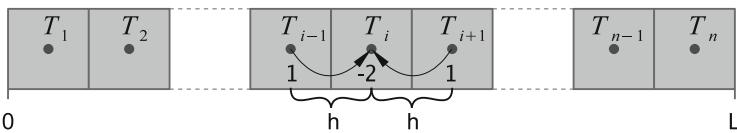


Fig. 14.3 Discretization of a bar

heat equation (for example, in the computation of the heat distribution in a rod), one needs to decide the number of discrete *grid points* at which to determine the temperature. The greater this number, the higher the expected accuracy as well as the computational effort. Figure 14.3 shows a section of a rod that has been divided into n intervals of equal length. In this example, we approximate the temperature, T_i , $i = 1, \dots, n$, at the midpoints of the intervals. These midpoints therefore have a distance $h = \frac{L}{n}$ to their neighboring midpoints.

We can now apply the discretization to the one-dimensional heat equation. For a given gridpoint i , using (14.4) in (14.7) implies

$$\kappa \cdot \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2} = 0,$$

which simplifies to

$$T_{i+1} - 2T_i + T_{i-1} = 0.$$

The equation for gridpoint i requires the temperature T_i at point i as well as the values T_{i-1} and T_{i+1} at the neighboring grid points. Such a discretization of the second derivative is called a *3-point-stencil* since three neighboring points are involved in the computation at point i . The discrete equation has to be satisfied over the entire (discretized) domain, in this case for all i from 1 to n . However, the two boundary points do not have any exterior neighbors. We will treat the boundary conditions in more detail in Sect. 14.2.3. Here, we only note that we will have fixed,

known values T_0 and T_{n+1} , at the boundary. Therefore, we obtain the following *linear system of equations* with n equations in n unknowns T_1, \dots, T_n :

$$\begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & -2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_{n-1} \\ T_n \end{pmatrix} = \begin{pmatrix} -T_0 \\ 0 \\ \vdots \\ 0 \\ -T_{n+1} \end{pmatrix}.$$

14.2.2 5-Point-Stencil

In view of the Laplace operator, the solution of (14.4) requires the two second order partial derivatives $\frac{\partial^2 T}{\partial x^2}$ and $\frac{\partial^2 T}{\partial y^2}$. Analogous to (14.7), we approximate these derivatives at (x, y) using a uniform grid width h for both the x - and y -directions by,

$$\begin{aligned} \frac{\partial^2 T}{\partial x^2} &\doteq \frac{T(x+h, y) - 2T(x, y) + T(x-h, y)}{h^2}, \\ \frac{\partial^2 T}{\partial y^2} &\doteq \frac{T(x, y+h) - 2T(x, y) + T(x, y-h)}{h^2}. \end{aligned}$$

This in turn yields the approximation

$$\kappa \left(\frac{T(x+h, y) + T(x, y+h) - 4T(x, y) + T(x-h, y) + T(x, y-h)}{h^2} \right) = 0$$

of (14.4), and after simplification yields

$$T(x+h, y) + T(x, y+h) - 4T(x, y) + T(x-h, y) + T(x, y-h) = 0.$$

Analogous to the one-dimensional case, we can subdivide the two-dimensional simulation domain into a grid of discrete cells and denote the temperature at the midpoint of cell (i, j) by $T_{i,j}$. In doing so we obtain the equation,

$$T_{i+1,j} + T_{i,j+1} - 4T_{i,j} + T_{i-1,j} + T_{i,j-1} = 0.$$

This discretisation scheme is called a *5-point-stencil*.

In the one-dimensional case, the left and right neighbors were needed for the computation of a cell. The individual temperature values were ordered in a vector which is important for setting up the system of equations as well as storing the values in the implementation. In order to set up a linear system of equations in the

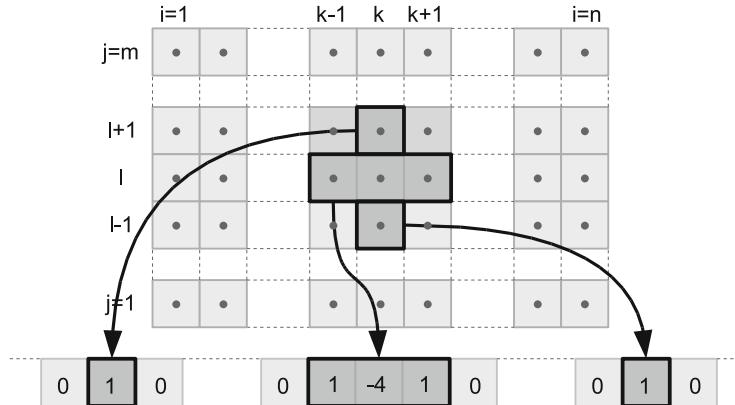


Fig. 14.4 Access to the required elements of the *5-point-stencil*

two-dimensional case, we need to determine an ordering in which the temperature values will be ordered in a vector. We choose a row-wise ordering: Neighbors which lie in the same row, defined in a geometric sense of the two-dimensionally ordered cells, will also be neighbors in the vector. Those neighbors which lie in the same column are separated from each other in the vector. Whereas this procedure appears straightforward and enjoys great popularity, it turns out to hinder an efficient implementation—at least if we would like to or even have to exploit the full capacity of the available computer hardware. (The so-called *spatial locality* is important for *cache-hierarchies* of modern processors: Items that are spatially close to each other should be treated promptly to ensure the availability of required quantities in the cache and to avoid expensive accesses to the main memory). Figure 14.4 illustrates the *5-point-stencil* for the computation of the element (k, l) as well as the ordering of the respective cells in the vector.

With these considerations, we now assemble a system of equations for the two-dimensional case. For elements at the boundary we proceed in a similar manner as in the one-dimensional case. For a very small test domain with only three cells per direction (without boundary), one obtains the following system of equations:

$$\left(\begin{array}{ccc|cc} -4 & 1 & & 1 & \\ 1 & -4 & 1 & & 1 \\ & 1 & -4 & & 1 \\ \hline 1 & & -4 & 1 & 1 \\ 1 & & 1 & -4 & 1 \\ & 1 & & 1 & -4 \end{array} \right) \cdot \begin{pmatrix} T_{1,1} \\ T_{1,2} \\ T_{1,3} \\ T_{2,1} \\ T_{2,2} \\ T_{2,3} \\ T_{3,1} \\ T_{3,2} \\ T_{3,3} \end{pmatrix} = \begin{pmatrix} -T_{0,1} - T_{1,0} \\ -T_{0,2} \\ -T_{0,3} - T_{1,4} \\ -T_{2,0} \\ 0 \\ -T_{2,4} \\ -T_{3,0} - T_{4,1} \\ -T_{4,2} \\ -T_{4,0} - T_{3,4} \end{pmatrix}. \quad (14.8)$$

The matrix for this system is divided into nine blocks. The number of blocks always equals the square of the number of rows in the discretized domain, and the number of entries (including zero entries) per block equals the square of the number of columns. Solving the linear system of equations yields the (numerical, or discrete) solution of the two-dimensional steady-state heat equation.

14.2.3 Boundary Treatment

In Chap. 2, we have learned that for PDEs there exist different possibilities to define conditions at the boundary of the domain in order to guarantee the existence of a unique solution to the problem. The intuitive choices in the previous two sections were *Dirichlet boundary conditions*. Here, the temperature values are given at all boundary cells but we still need to consider a few additional details. For example, one has to determine what constitutes the boundary. In our case, we divided the domain into cells and computed temperatures at the cell midpoints (see Fig. 14.4). This opens up two possibilities: Either one fixes the temperature for the outermost layer of cells within the domain, or one adds another layer of cells around the domain in which one fixes the temperature. Here, we will pursue the second option, resulting in indices 0 and $n + 1$. Instead of computing the temperatures at the cell midpoints, which appears reasonable for illustrative purposes, one could also have chosen the vertices of the cells (or intersections of gridlines, resp.). Using vertices then implies that the outermost values would lie directly on the boundary of the domain which would subsequently simplify the boundary treatment. This would also be more meaningful in the sense that a given temperature usually holds directly at the boundary and not half a cell width away from it.

These considerations illustrate the amount of work which is required for the transition from the assignment of Dirichlet boundary conditions in the model to a numerical solution technique and its implementation. The question of boundary conditions will again reappear in Chap. 15, Computational Fluid Dynamics.

14.3 Numerical Solution of the PDE

The discretization for either the one- as well as the two-dimensional case produces a large *linear system of equations* with a corresponding *sparse* coefficient matrix—in the k 'th row, there can be nonzero entries only in positions that are connected to the grid point of this row via the discretization stencil, and there are at most three or five such positions, resp. In principle, this structure remains the same for three dimensions; there will be two additional off-diagonal entries since now the local grid point also has direct neighbors in the z -direction. The special structure of these systems of equations suggests the use of *iterative methods* for their solution as introduced in Sect. 2.4.4. Direct solvers such as Gaussian elimination are usually

not recommended in light of the typically very large number of grid points and subsequent unknowns in the system. In particular, the *relaxation methods* are easy to apply and have proven sufficient for the not too large two-dimensional examples.

14.3.1 Simple Relaxation Methods

We now seek the solution of (14.8) with quite a few more discrete grid points. For many iterative methods the system of equations and its corresponding matrix do not have to be assembled explicitly; it is sufficient to know how to *apply* the matrix, i.e., to multiply it by an arbitrary vector. If we store all interior cells as well as another layer of cells for the given boundary in a two-dimensional array and then loop over all interior cells, we no longer need to distinguish between the interior and boundary cells. For each cell, the corresponding row of the system of equations is used to compute an approximate solution. The elements that are required according to the *5-point-stencil* are hereby extracted from the two-dimensional array. The remaining question concerns the number of iteration steps that should be performed.

Figure 14.5 shows the simulation result after 20 (left) and 500 (right) steps of the *Jacobi iteration* for a domain with 40×20 cells. In this example, the lower boundary has a prescribed fixed warm temperature (100°) and the other boundaries have a constant cooler temperature (0°).

In general, one cannot, or does not, want to simply prescribe the number of iterations. Instead, one iterates until a certain quality threshold is reached. A natural measurement for this quality would be the *error*, e , between the current approximation and the exact solution which, in general, is unknown. Therefore, one uses the *residual* which is defined by

$$r := b - Ax,$$

where A denotes the matrix and b the righthand side of the system of equations. In this form, the residual is a vector whose elements may serve as an indicator of the error at each discrete position in view of the relationship $r = -Ae$. It is desirable to have a scalar value as the stopping criterion for the iteration. Such a value, for example, may be computed as

$$\bar{r} = \sqrt{\frac{\sum_1^n r_i}{n}}.$$

The smaller this value \bar{r} , the more accurate—at least hopefully—the current solution. However, we cannot be completely sure since A could be of the type in which a small residual occurs side by side with a large error. In the above simulation with starting values of zero throughout the domain, it takes 500 iteration steps to reduce \bar{r} from 22.36 to 0.0273. In general, there is no guarantee that this

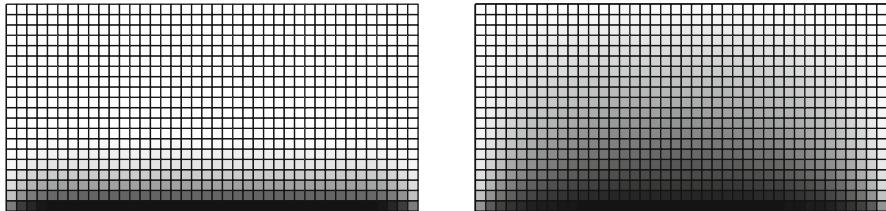


Fig. 14.5 Simulation on a domain of 40×20 cells with Dirichlet boundary and high temperature at the lower boundary: heat distribution after 20 (*left*) and 500 (*right*) steps

is accurate enough. Performing additional iterations, however, will produce only small changes since the Jacobi iteration uses the current residual to compute the next iterate. Furthermore, the computed heat distribution satisfies our expectations given the prescribed boundary conditions.

In this context, we will briefly mention two additional relaxation methods which have already been introduced in Chap. 2 in the section on numerics. If we use the *Gauß–Seidel method* instead of the Jacobi method for our example, we obtain the same accuracy in only 247 steps instead of 500. For the *SOR-method*, the so-called *overrelaxation* method, the performance depends on the choice of the relaxation parameter. It is well-known that it has to be smaller than 2. If it is larger, the method no longer converges. The optimal factor depends on the given problem, and in our example is found to be 1.77. The effect is that now only 31 iteration steps are required to reach the same accuracy which is a significant improvement over the Jacobi method. This becomes even more important when we consider very fine grids to resolve our domain since the number of iteration steps in all relaxation methods depends on the number of (discrete) unknowns and thus on the size of the matrix. Hence, the number of steps increases much faster for the Jacobi method than the SOR method. This undesirable behavior will be addressed in the following with a much more involved way to solve linear systems of equations—the *multigrid methods*.

14.3.2 Multigrid Methods

In the previous section, it was criticized that the computational work increases more than linearly with respect to the number of discretization points. As a consequence for the numerical simulation, an increase in resolution (i.e., a decrease in grid width and thus an increase in accuracy) does not only imply an increase in computational work per iteration step (in view of the larger matrix) but also requires a greater number of iteration steps to reduce the initial error by a prescribed factor. Currently, this fact is one of the main reasons why some large problems cannot be simulated due to the excessive simulation time.

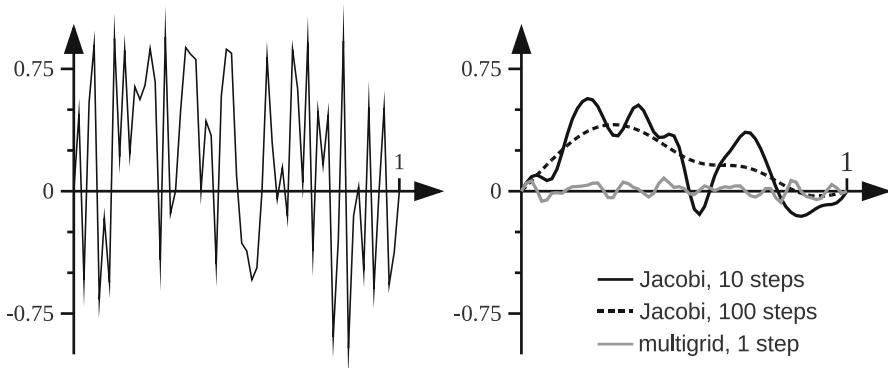


Fig. 14.6 Random initial values (left) and error of an approximate solution of a one-dimensional heat equation with different iteration methods (right)

Without doubt, a linear computational effort would be optimal for a linear system of equations since each discretization point has to be visited at least once—if only to figure out the starting value. In order to study the source for deteriorating convergence behavior of relaxation methods, one can test the solver against problems where the solution is known. In the case of the heat equation, one can set all boundary conditions to the same value and choose the initial condition randomly. The exact solution will be a constant temperature throughout the entire domain, namely the one prescribed on the boundary.

Figure 14.6 documents the result of such an experiment for the one-dimensional case. A constant reference temperature is applied to both ends of a rod, whereas the initial temperature values at the interior points are chosen randomly and thus vary significantly (Fig. 14.6, left). The right part of Fig. 14.6 shows the error—after 10 Jacobi steps, after 100 Jacobi steps, and after a single *multigrid step*. Even if you have no idea how multigrid works, you hopefully are impressed!

What should we learn from Fig. 14.6? After a few iterations with the Jacobi iteration, the approximation becomes visibly *smoother*. This does not come as a surprise: The heat equation (14.4) is solved exactly when the second derivatives disappear everywhere. Although the not-so-smooth components of the error are apparently smoothed quickly, the other components are not reduced. To simplify, we view the error as a composition of different *frequency bands*—the high-frequency components are eliminated quickly, whereas the low-frequency parts are hardly reduced at all. In the above example, even after 100 Jacobi iterations we observe an alarmingly large error (especially when consideration is given to the fact that 100 iteration steps constitute quite a large number of iterations for such a trivial, one-dimensional problem!).

One can debate whether a certain frequency should be considered as *high* or not, which leads next to the underlying principle of multigrid methods. In fact, whether an error frequency is or is not viewed as high depends on the resolution of the

discretization grid. According to the Nyquist–Shannon sampling theorem, a signal has to be tested, i.e., discretized with a frequency which is at least twice as high as the maximal frequency occurring in the signal in order to be reconstructed correctly. If a smaller number of points is used the function changes more rapidly between the grid points, meaning that the frequency becomes *too* high for this grid—and this constitutes a discretization error. Those frequencies which can be represented on this grid could once more be divided into high and low frequencies. The high frequencies are the ones for which this grid is suitable; the low frequencies (for which we observed difficulties when using relaxation methods) do not need such a fine grid for their representation—a coarser grid would suffice.

The reasoning given above suggests that one begins by *smoothing* the initial guess (i.e., to remove the high-frequency components) and then transfers the problem to a coarser grid. On the coarse grid we need to convert the low frequencies (with respect to the fine grid) into high frequencies (with respect to the coarse grid) to be tackled next. This second part of the method is called *coarse grid correction*—one exploits a coarser grid in order to correct, i.e., improve the current approximation on the fine grid. This suggests transferring the residual to the coarser grid where we solve a so-called *correction equation* where the residual is the right hand side vector and its exact solution is the error—the ideal *correction*.

Such a *two-grid approach* may now be continued recursively since the error on the coarse grid once more has low frequency components. We will therefore visit a *hierarchy* of different grids Ω_l , $l = 1, \dots, L$, with respective grid widths $h_l = 2^{-l}$. The following is the resulting prototype of a *multigrid algorithm*, starting on a fine grid on level l .

```

multigrid(l, b, x) {
    x = jacobi(l, b, x)                      // pre-smoothing
    if(l>0) {                                 // stopping criterion
        r = residual(l, b, x)                  // compute residual
        b_c = restrict(l, r)                  // restriction
        e_c = zero_vector(l-1)
        e_c = multigrid(l-1, b_c, e_c) // recursion
        x_delta = interpolate(l, e_c) // prolongation
        x = x + x_delta                   // correction
    }
    x = jacobi(l, b, x)                      // post-smoothing
    return x
}

```

This procedure is called the *V-cycle* since the recursion, as illustrated in Fig. 14.7 (left), descends to the coarsest grid and then ascends directly back to the finest grid. There exist further variants (e.g., the *W-cycle* in Fig. 14.7, right) which may be superior with respect to their convergence behaviors. These will not be treated further here, however. In the following, aspects concerning some important algorithmic details on the individual steps of the V-cycle are provided.

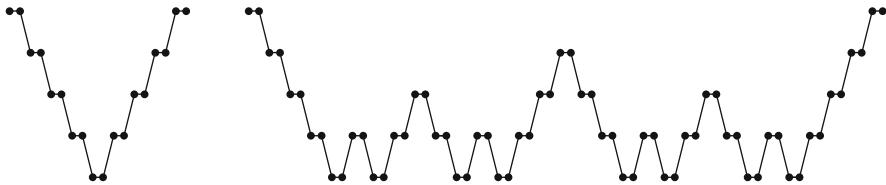


Fig. 14.7 Schematic illustration of the course of recursion in the multigrid method; V-cycle (left) and W-cycle (right)

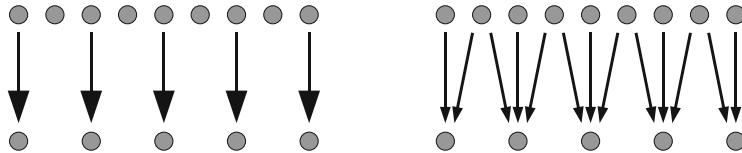


Fig. 14.8 Methods for the restriction; Injection (left) and full weighting (right)

Pre-smoothing

If we assume nothing is known about the solution of the PDE, then we also must assume that the initial error e_1 contains various high and low frequencies. With respect to the fine grid Ω_1 , we need to remove the high frequencies before we can transition to a coarser grid to tackle the low frequencies—otherwise information would be lost in the transition to the coarse grid Ω_{l-1} . Such *presmoothing* can be accomplished with only a few steps of a relaxation method, e.g. the Jacobi or Gauß–Seidel method. Thereafter, and in the ideal case, the residual r_l of the approximation x_l would contain only low frequency components (with respect to Ω_l).

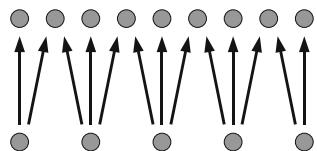
Restriction

Once the residual of the pre-smoothed approximation is computed, it needs to be transferred to the next coarser grid. There exist several possibilities, of which the easiest is to choose only every other point and to ignore the others (the so-called *injection*). This procedure is illustrated in Fig. 14.8 (left). Instead of ignoring the intermediate points, one could average the coarse grid point by using the fine grid point ($\frac{1}{2}$) and the two neighbors ($\frac{1}{4}$ each). This approach is called *full weighting* and is illustrated in Fig. 14.8 (right). In higher dimensions, one would have to consider additional points for averaging. The hope is that the increase in effort will yield an increase in accuracy when compared to the simple injection.

Coarse Grid Solution

The coarse grid has been introduced in order to find a correction for the current approximation that will remove the low error frequencies. Such a correction may be

Fig. 14.9 Transfer of coarse grid values to the fine grid using linear interpolation



obtained by solving $Ae = r$, which in turn could be accomplished using a general solver such as, for example, the SOR method, yielding a *two-grid method*. However, it turns out better if one proceeds with the solution of the coarse grid problem in the same way as for the fine grid problem, i.e., to first smooth and then coarsen again. This procedure implies a recursive call to the multigrid method. At some point this recursion obviously has to end—this typically occurs when the system of equations is small enough to be solved efficiently by a direct solver.

Prolongation

The goal of the coarse grid correction is to remove the low error frequencies. When the multigrid method returns from the recursion, i.e., when it moves from the coarse back to the fine grid, the correction, which has been computed on the coarse grid, needs to be *interpolated*, or *prolongated*, to the fine grid. The simplest method is linear interpolation as illustrated in Fig. 14.9. Here, intermediate fine grid points are computed as the average of the two neighboring coarse grid points. In this way, we obtain a correction value for each fine grid point which can then be added to the previous approximation x_l .

Post-smoothing

Theoretically, we have now removed both the high as well as the low error frequencies. However, in reality the prolongation may introduce new high frequency error components that can be reduced through a few additional relaxation iterations.

Cost and Effect

In comparison to typical relaxation methods, the multigrid method (which we introduced in very brief form) appears to be a more complicated algorithm in which a single iteration step—a V-cycle—requires more work than, for example, a Jacobi step. In fact, a V-cycle already includes several steps (typically two) of a relaxation method for pre- and post-smoothing. The cost for the computation of the residual, the restriction, prolongation and correction are smaller than those for the smoother and are therefore neglected here. Since the deciding factor for assessing

the multigrid method is the overall computational cost, we need to analyze the cost that occurs for the recursion.

Let n denote the number of fine grid points, and assume that cn denotes the cost on the finest grid for a constant c . Then the cost on the second finest grid is $c\frac{n}{4}$ (in two dimensions) due to the smaller number of grid points. The cost for further coarsenings can be represented as a geometrical series:

$$cn \sum_{i=0}^{\infty} \frac{1}{4}^n = cn(1 + \frac{1}{4} + \frac{1}{16} + \dots) = cn \frac{1}{1 - \frac{1}{4}} = \frac{4}{3}cn.$$

The combined cost for all the coarser grids is only a third—in three dimensions only a seventh—of the cost on the fine grid and may thus be neglected.

The total cost of a multigrid iteration is thus $\mathcal{O}(n)$, just as the cost of the relaxation methods, but with a larger constant factor. The number of iterations required for a specified accuracy, however, is $\mathcal{O}(1)$, which implies that the method is linear with respect to the number of grid points. Figure 14.6 (right) shows an example taking a random initial error and after a single multigrid step the error was reduced significantly better than had 100 Jacobi iterations been applied.

Concluding Remarks

The brief introduction of multigrid methods should only be interpreted as an “appetizer”. It highlights the algorithmic considerations that can be involved in the key task of efficiently solving large, sparse systems of linear equations. There exist numerous variants, refinements, adjustments, etc., which lie beyond the scope of this book.

Chapter 15

Fluid Dynamics

Fluid flows have always been among the most modeled and (numerically) simulated phenomena, appearing in entirely different contexts and disciplines. Astrophysics, plasmaphysics, geophysics, climate research and weather forecasting, process engineering and aerodynamics as well as medicine—fluid flows are studied everywhere even though the materials, or *fluids*, are completely different. As we have seen in Chap. 7 with macroscopic traffic simulation, flow processes are also taken as a model for applications in which no substance flows in the traditional sense. Flows are also a prime example of very computing intense simulations—in particular if turbulence plays a role—and are thus often a case for *high performance computing (HPC)* and *high performance computers*. It is therefore natural that this book includes a chapter on *computational fluid dynamics (CFD)*. With respect to simulation we must again deal with a numerical simulation approach while our concerns with modeling mostly involve a PDE-based model. Accordingly, our greatest needs are met from the sections on analysis and numerical analysis presented as the tools in Chap. 2.

Choosing from the multitude of relevant scenarios, in the following we select the case of *viscous*, i.e., *with friction*, laminar flows of *incompressible* fluids. The necessary terms will be introduced and discussed in the following section but as a rule of thumb for this chapter one is better off thinking of honey rather than air. For simplicity, we additionally remain within a two-dimensional geometry. The flow around an obstacle shall serve as an example of an application. For a more detailed discussion of this topic we refer the reader to the books “Numerical Simulation in Fluid Dynamics: A Practical Introduction” by Michael Griebel, Thomas Dornseifer and Tilman Neunhoeffer [30] and “Computational Methods for Fluid Dynamics” by Joel H. Ferziger and Milovan Peric [19].

15.1 Fluids and Flows

In general, fluids are understood to be materials in a state in which they cannot resist shear forces. Essential for the (observable) movements of fluids are forces—exterior (for example gravitation when a river flows downstream) or internal (among fluid particles themselves or between fluid particles and the boundary of the domain, for example an obstacle). For the latter, it is the *viscosity* or *toughness* of a fluid that is responsible and from which *friction* is produced. Friction, or viscosity, cause the (macroscopic) motion of fluids that is not induced by exterior forces to eventually stop. Gases, on the other hand, are in general idealized and considered as *inviscid* fluids; here, the inertia dominates the viscosity. The proportion of these two properties is specified through the so-called *Reynolds number* that also accounts for reference values for velocity and domain size. The Reynolds number therefore characterizes a fluid flow scenario or, in short, a flow and not only a fluid.

Basically, one distinguishes between *laminar* and *turbulent* flows. The former are based on the idea that neighboring layers of the fluid hardly mix in the course of the flow—friction or viscosity are too large; one thinks for example of honey. This is entirely different for turbulence, colloquially known from travel in aircraft: Here, a heavy mixing occurs and vortices of different size and intensity appear. Even if in principle we want to be satisfied with this somewhat imprecise characterization of this phenomenon, let us nonetheless point to an important physical detail. The vortices of different sizes—and by all means the spectrum can extend over several orders of magnitude, for example in the wake of an airplane at take-off—coexist not simply side by side but rather through strong interaction between one another as given, for example, through a strong energy transport. For the simulation used in practice this implies that one may not neglect very small vortices, even if one is only interested in a very rough scale. Kolmogorov has quantified this relationship and formulated it in a famous law. Thus, once again one has a *multiscale problem* that we have already discussed in the introductory Chap. 1.

One can very nicely illustrate the interplay of modeling and simulation with the example of turbulence. On the one hand, one can simply start with a (yet to be discussed) basic model of fluid dynamics, the *Navier–Stokes equations*, and abstain from any additional modeling effort. The price paid, however, is the immense and nowadays mostly unfeasible computational effort. The other extreme position to this *direct numerical simulation (DNS)* attempts to encompass the turbulence entirely through the modeling technique—by averaging and by including additional quantities and additional equations. In contrast to DNS, this now requires significantly more modeling effort. In fact, in the course of time a multitude of *turbulence models* have been developed, among them the famous *k- ε model*. The advantage is that one can achieve a reduced computational effort: The details are in the model and no longer have to be reproduced by the resolution of the discretization. A compromise is taken with the approach for the *large-eddy simulation (LES)* that directly resolves vortices up to a certain size, leaving all scales that are no longer included directly to be treated through the modeling technique. Each

approach has their advocates: Naturally, mathematical modelers and analysts favor a model-sided solution—the DNS approach is sometimes denounced dismissively as a “sledgehammer approach”. Conversely, numerical analysts seem eager to happily misjudge the value of high-quality models and point out the imminent modeling mistakes like a mantra. As so often found in this world, there just is no “right” and no “wrong”, but only different paths to the (unreachable) goal.

The illustrative example per se for laminar or turbulent flows is provided by the so-called *Kármán vortex street* found in the wake of a flow around obstacles. For viscous fluids or low velocities the fluid closely snuggles to the obstacle and returns to a normal state of order after a short distance from the obstacle. For decreasing viscosity or increasing velocity, vortices slowly begin to detach from the obstacle and a rather regular formation of vortices results. With an additional increase in velocity the regular pattern then gives way to a strongly irregular and finally, for turbulence, chaotic appearing form—the vortex street. Figure 15.1 shows different developments of the flow in the wake of an obstacle up to the Kármán vortex street.

Yet a third pair of terms needs to be introduced: *compressible* and *incompressible*. One speaks of a compressible fluid if it can be “pressed together”, i.e., if a certain mass need not automatically assume a certain volume, but where the latter depends on the pressure. Gases at high velocities are the standard example for compressible fluids while liquids on the other hand are incompressible. Gases at low velocities are typically also modeled as incompressible fluids—which at first sight is easier since here the density can be assumed to be constant.

In the following we once more traverse the pipeline: Starting with an established model, the *Navier–Stokes equations*, we next introduce a discretization technique that is based on the so-called *marker-and-cell approach (MAC)* and last we attend to the numerical solution of the discretized equations. Using the example of the flow around an obstacle that we have mentioned repeatedly, we demonstrate the capability of the chosen approach. Then, in a final outlook, we briefly discuss what other possibilities exist if one is not subject to a compact and only introductory treatment of the topic of flows.

15.2 Mathematical Model

We now turn to the model for the spatially and temporally resolved description of laminar, viscous flows of incompressible fluids that were established long ago, the *Navier–Stokes equations*. We first formulate the equations and then follow up with some remarks concerning their derivation.

15.2.1 Navier–Stokes Equations

In general, the flow of a fluid in a two-dimensional domain $\Omega \subset \mathbb{R}^2$ over a time interval $[0, T]$ is primarily characterized through three physical quantities: the

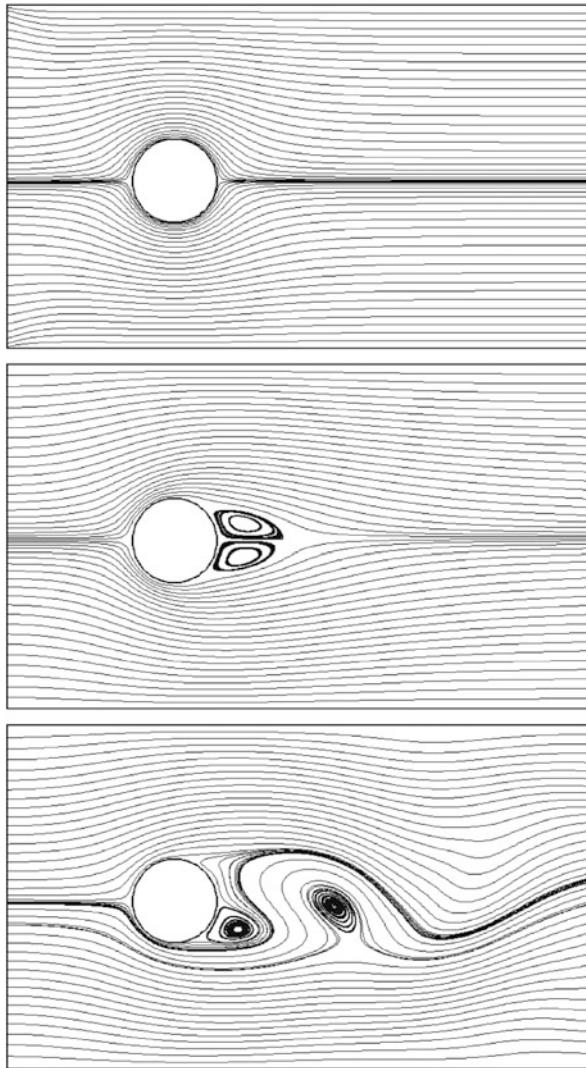


Fig. 15.1 Flow in the wake of an obstacle at different Reynolds numbers: from the tedium to the Kármán vortex street

velocity field $\mathbf{u}(x, y; t) = (u(x, y; t), v(x, y; t))^T$, the pressure $p(x, y; t)$ and lastly, the density $\varrho(x, y; t)$ that is assumed to be constant in the incompressible case, i.e., $\varrho(x, y; t) = \varrho_0$. In the so-called *dimensionless form*, the Navier–Stokes equations are given as

$$\begin{aligned}\frac{\partial}{\partial t} \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} &= -\nabla p + \frac{1}{\text{Re}} \Delta \mathbf{u} + \mathbf{g}, \\ \operatorname{div} \mathbf{u} &= 0.\end{aligned}\quad (15.1)$$

Here, $\text{Re} \in \mathbb{R}$ denotes the already mentioned dimensionsless *Reynolds number* and $\mathbf{g} = (g_x, g_y)^T$ denotes the sum of the external forces such as gravitation, for example. The first equation is called the *momentum equation* while the second is known as the *continuity equation*. If one uses the latter above in the term $(\mathbf{u} \cdot \nabla) \mathbf{u}$ to reformulate, one then obtains the equivalent formulation,

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{1}{\text{Re}} \Delta u - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} - \frac{\partial p}{\partial x} + g_x, \\ \frac{\partial v}{\partial t} &= \frac{1}{\text{Re}} \Delta v - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} - \frac{\partial p}{\partial y} + g_y, \\ 0 &= \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}.\end{aligned}\quad (15.2)$$

To fix the unknowns uniquely—we are already familiar with this issue—in addition to the equations, it is required to have *initial values* for the unknowns at the beginning (typically, the values are given for the velocity components that satisfy the continuity equation) as well as *boundary values* at the boundary of the domain for all times. To compactly formulate the boundary conditions independently of the respective course of the domain boundary, we introduce the normal component (perpendicular to the boundary) $\varphi_n := \mathbf{u}^T \cdot \mathbf{n}$ in which \mathbf{n} is the outward normal vector as well as the tangential component (tangential to the boundary) of the velocity, φ_τ . In the paraxial case, $\Omega = [x_l, x_r] \times [y_u, y_o]$, it therefore holds that, depending on the section of boundary, $\varphi_n = \pm u$ and $\varphi_\tau = \pm v$, or vice versa.

The following boundary conditions are particularly common, where $(x, y) \in \Gamma := \partial\Omega$ denotes a point on the domain boundary $\partial\Omega$:

- *no-slip condition*: the idea of a wall—no fluid can penetrate, and the adhesion prevents any tangential movement in the immediate proximity of the boundary, i.e.,

$$\varphi_n(x, y; t) = \varphi_\tau(x, y; t) = 0;$$

- *free slip condition*: the idea of a virtual boundary (e.g., given an axially symmetric canal, then for cost reasons only half of the domain is simulated)—no fluid can penetrate, but now a tangential movement is possible, i.e.,

$$\varphi_n(x, y; t) = 0, \quad \frac{\partial \varphi_\tau(x, y; t)}{\partial n} = 0;$$

- *inflow condition*: the idea of an inlet—classical Dirichlet boundary conditions for both velocity components, i.e.,

$$\varphi_n(x, y; t) = \varphi_n^0, \quad \varphi_\tau(x, y; t) = \varphi_\tau^0$$

with given functions φ_n^0 and φ_τ^0 ;

- *outflow condition*: the idea of an outlet—no change in both velocity components in the outlet direction, i.e.,

$$\frac{\partial \varphi_n(x, y; t)}{\partial n} = \frac{\partial \varphi_\tau(x, y; t)}{\partial n} = 0;$$

- *periodic boundary condition*: the idea of a periodically continued simulation domain—velocities and pressure must coincide at the inlet and outlet, i.e.,

$$u(x_l, y; t) = u(x_r, y; t), \quad v(x_l, y; t) = v(x_r, y; t), \quad p(x_l, y; t) = p(x_r, y; t)$$

for periodicity in the x -direction.

If for all boundaries the velocity components are specified directly, then the constraint for the velocity field to be divergence-free requires the additional condition that,

$$\int_{\Gamma} \varphi_n \, ds = 0$$

for the boundary values. This means simply that the same amount of fluid flows into the domain as it flows out.

Extensive statements regarding the existence and uniqueness of solutions could be proved for the two-dimensional case. In contrast, the existence and uniqueness of the solution for arbitrary time intervals could not be shown yet for three dimensions. As far as the computation of solutions in specific scenarios is concerned, the analysis in the two-dimensional setting also reaches its limits admittedly fast—here a numerical approach is typically indispensable.

15.2.2 Remarks Concerning the Derivation

This section can be made short since a comparable derivation was already presented in the previous chapter for heat transfer. In fact, the approach is closely related: Once more, the starting point are the *conservation laws*—in the case of flows it is now the *mass conservation*, which eventually leads to the continuity equation, as well as the *conservation of momentum*, which results in the momentum equation. One obtains the mass at every time instant through integration of the density over the current simulation domain (which may actually change as in the example for

the case of moving boundaries). The momentum results from the corresponding volume integral over the product of the density and velocity field. One can again set up the conservation or balance equations since on the one hand the derivatives of the respective integrals are indicators for change and have to disappear in view of the conservation principle while on the other hand they have different origins. Here, however, one considers portions of the fluid that move with the flow instead of spatially fixed domains. Finally, one discards the integral with the standard argument “must hold for all integration domains” and thus obtains partial differential equations. Depending on whether one considers viscous or inviscid fluids or flows, one has to model the so-called *stress tensor* differently. In doing so, one obtains either the previously introduced Navier–Stokes equations or the central model of *gas dynamics*, the *Euler equations*, which will not be discussed further.

These remarks should suffice here and we will now turn to the numerical treatment of the Navier–Stokes equations.

15.3 Discretization of the Navier–Stokes Equations

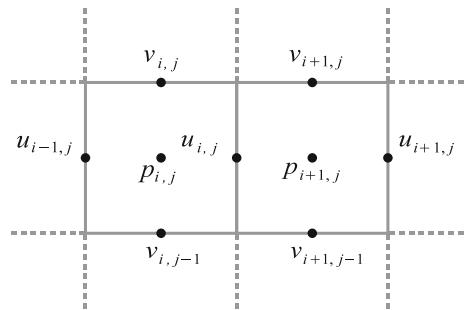
After a brief discussion of the selected discretization approach, the numerical treatment of the spatial and temporal derivatives is introduced. Remarks on the numerical treatment of the boundary conditions conclude this section.

15.3.1 Finite Differences

Yet another (and final!) time in this book we use finite differences as a discretization principle. The essential reason for this is—once more—their simplicity. The diversity in practice however knows hardly any bounds. For example, frequently employed methods include *finite volumes*, *finite elements*, *spectral elements* or so-called *meshfree* or *particle-based* methods such as *smooth particle hydrodynamics (SPH)*. With the increased computer capacities, one nowadays even uses *molecular dynamical* approaches to study flows. In these cases, one obviously does not study macroscopic scenarios but one is for example interested in a precise description of the behaviour of fluids at walls in order to derive improved (macroscopic) boundary conditions from the microscopic results.

The use of finite differences implies (or at least strongly suggests) the adoption of Cartesian (i.e., composed of rectangles in a structured way) grids, which is also assumed here in the following. As the domain we consider the rectangular flow canal $[x_l, x_r] \times [y_u, y_o]$ of length $l_x := x_r - x_l$ and width $l_y := y_o - y_u$. Let the interval $[0, T]$ be the time window to be considered. If $N_x + 1$ and $N_y + 1$ equidistant gridpoints in x - and y -direction are used, then this corresponds to N_x and N_y respective *cells* of *mesh width* $h_x := l_x/N_x$ and $h_y := l_y/N_y$ in x - and

Fig. 15.2 Basic scheme of a two-dimensional staggered grid



y -direction, resp. Analogously, we assume to have N_t time steps of constant step width $\delta t := T/N_t$.

One notes that the unknowns (i.e., the discrete values for velocity components and pressure) are not necessarily all placed at the gridpoints, but e.g. also on the cell boundaries—an idea that was borrowed from the finite volume world. One also speaks of *staggered grids*, and in the following we will also make use of staggered grids: The discrete velocity components in the x -direction (u -values) will be placed at the midpoints of the vertical cell boundaries, the discrete velocity components in the y -direction (v -values) at the midpoints of the horizontal cell boundaries. Only the discrete pressure values sit at the cell midpoints (see Fig. 15.2). One more remark concerning the notation: We denote the cell $[(i-1)h_x, ih_x] \times [(j-1)h_y, jh_y]$ as cell (i, j) , $1 \leq i \leq N_x$, $1 \leq j \leq N_y$. The corresponding index i and j have u , v and p at the right boundary, top boundary and at the midpoint of this cell, respectively (see Fig. 15.2).

15.3.2 Treatment of Spatial Derivatives

We now proceed to the discretization of the Navier–Stokes equations (15.2) according to the approach given above. For the continuity equation (third equation in (15.2)) we apply *central differences* with half the mesh width of the respective cell midpoint, i.e.,

$$\frac{\partial u}{\partial x} \approx \frac{u_{i,j} - u_{i-1,j}}{h_x}, \quad \frac{\partial v}{\partial y} \approx \frac{v_{i,j} - v_{i,j-1}}{h_y}. \quad (15.3)$$

In contrast, the momentum equation for u (first equation in (15.2)) is discretized at the midpoints of the vertical cell boundaries while the momentum equation for v (second equation in (15.2)) is discretized at the midpoints of the horizontal cell boundaries. For the second partial derivatives $\partial^2 u / \partial x^2$, $\partial^2 u / \partial y^2$, $\partial^2 v / \partial x^2$ as well as $\partial^2 v / \partial y^2$, the so-called *diffusion* terms, this proceeds without any problem using the techniques already reviewed. The spatial derivative for the pressure is also worked

without any trouble using central differences with half the mesh width as discussed above.

Slightly more complicated is the situation for the so-called *convective* terms $\partial(u^2)/\partial x$, $\partial(uv)/\partial x$, $\partial(uv)/\partial y$ as well as $\partial(v^2)/\partial y$ —after all, there are no points in which u and v are available in discrete form. However, a suitable averaging process provides a remedy so that, as an example, for $\partial(uv)/\partial y$, one obtains

$$\frac{\partial(uv)}{\partial y} \approx \frac{1}{h_y} \left(\frac{(v_{i,j} + v_{i+1,j})(u_{i,j} + u_{i,j+1})}{4} - \frac{(v_{i,j-1} + v_{i+1,j-1})(u_{i,j-1} + u_{i,j})}{4} \right). \quad (15.4)$$

Analogously, one proceeds with $\partial u^2/\partial x$ to obtain

$$\frac{\partial(u^2)}{\partial x} \approx \frac{1}{h_x} \left(\left(\frac{u_{i,j} + u_{i+1,j}}{2} \right)^2 - \left(\frac{u_{i-1,j} + u_{i,j}}{2} \right)^2 \right). \quad (15.5)$$

This simple procedure quickly reaches its limits, however. For larger Reynolds numbers, one may no longer employ purely central differences due to the now dominant convective parts of the momentum equations and their resultant stability problems. Rather, in this case one employs a mixture of these and the so-called *upwind discretization* or even alternatives such as the *Donor-Cell scheme*. At this juncture we are content with only referencing the imperfection of the discretization suggested in (15.4) as well as in (15.5) and will thus turn our attention to the temporal derivatives in (15.2).

15.3.3 Treatment of Temporal Derivatives

The spatial discretization discussed above must be performed at every discrete time instant $n\delta t$, $n = 1, \dots, N_t$. As a consequence, for each discrete time instant the values of the velocity components u and v as well as the pressure p must be considered, or in fact computed. In the following, we denote the time instant using an upper right index in parentheses, i.e., $u^{(n)}$ etc.

Considering (15.2), the only temporal derivatives are first derivatives of the velocity components u and v . For these, the *explicit Euler method* leads us to

$$\left(\frac{\partial u}{\partial t} \right)^{(n+1)} \approx \frac{u^{(n+1)} - u^{(n)}}{\delta t} \quad (15.6)$$

and an analogous result holds for $\partial v/\partial t$.

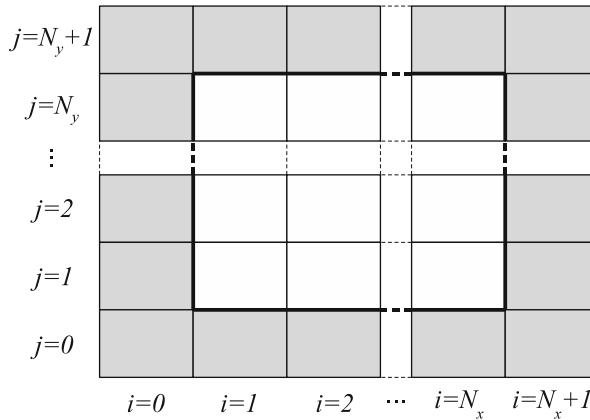


Fig. 15.3 Boundary layer for a staggered grid in order to satisfy the boundary conditions

15.3.4 Treatment of Boundary Conditions

For the staggered grid, near the boundary not all unknowns actually lie exactly on the boundary (see Fig. 15.2). For this reason, an additional boundary layer of grid cells is considered as well (see Fig. 15.3). In fact, all appearing boundary conditions can now be satisfied through averaging. For the discretization of the Navier–Stokes equations introduced, one now takes the u - and v -values lying on the boundary of the domain. Namely, these are $u_{0,j}, u_{N_x,j}$ for $j = 1, \dots, N_y$, as well as $v_{i,0}, v_{i,N_y}$ for $i = 1, \dots, N_x$. Furthermore, one takes values outside of Ω (i.e., in the boundary layer) into account, namely $u_{i,0}, u_{i,N_y+1}$ for $i = 1, \dots, N_x$, as well as $v_{0,j}, v_{N_x+1,j}$ for $j = 1, \dots, N_y$. The latter are to be specified separately depending on the boundary condition type. We will now take a closer look at four of the five types previously introduced:

- *no-slip condition:* In the case of a static boundary, all velocity components shall be zero. For the values lying directly on the boundary it therefore holds that,

$$u_{0,j} = u_{N_x,j} = v_{i,0} = v_{i,N_y} = 0, \quad i = 1, \dots, N_x; j = 1, \dots, N_y,$$

while for the others one specifies at the vertical boundaries,

$$v_{0,j} = -v_{1,j}, \quad v_{N_x+1,j} = -v_{N_x,j}, \quad j = 1, \dots, N_y,$$

and at the horizontal boundaries

$$u_{i,0} = -u_{i,1}, \quad u_{i,N_y+1} = -u_{i,N_y}, \quad i = 1, \dots, N_x.$$

In this way—for linear interpolation—there are velocity values of zero being generated on the actual boundary.

- *free slip condition*: Since in our discretization the velocity components that are perpendicular to the respective boundary lie directly on the boundary, it initially follows as before that,

$$u_{0,j} = u_{N_x,j} = v_{i,0} = v_{i,N_y} = 0, \quad i = 1, \dots, N_x; j = 1, \dots, N_y.$$

This is different, however, for the tangential velocity components: Here, the condition for constant values at the boundary is enforced through the conditions that,

$$v_{0,j} = v_{1,j}, \quad v_{N_x+1,j} = v_{N_x,j}, \quad j = 1, \dots, N_y,$$

as well as

$$u_{i,0} = u_{i,1}, \quad u_{i,N_y+1} = u_{i,N_y}, \quad i = 1, \dots, N_x.$$

- *outflow condition*: Here, the two velocity components perpendicular to the boundary shall not change. On the left and right boundary, this implies for u

$$u_{0,j} = u_{1,j}, \quad u_{N_x,j} = u_{N_x-1,j}, \quad i = 1, \dots, N_x,$$

while the specifications there for v and respectively for u and v at the bottom or top boundary are carried out analogously.

- *inflow condition*: In this case, explicit values are prescribed for u and v on the respective boundary. For the points that lie directly on the boundary this is trivial, in the other case one typically averages again (e.g., the v -values on the right boundary).

15.4 Numerical Solution of the Discretized Equations

The ingredients for a numerical solution are now available. Still missing are the coupling of the spatial and temporal discretization as well as the subsequent (iterative) solution method for the resulting discrete system of equations. This is the subject of the current section.

15.4.1 Time Step

The time loop typically represents the outer iteration. For the initial time instant $t = 0$, there exist initial values for the entire simulation domain, and subsequently, the (unknown) values at time t_{n+1} can be computed from the (known) values at time t_n —a classical *time step* known from the ODEs from previous chapters.

Given the temporal derivatives in (15.2)—presently in their continuous form with respect to space—this implies,

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + \delta t \left(\frac{1}{\text{Re}} \Delta u - \frac{\partial(u^2)}{\partial x} - \frac{\partial(uv)}{\partial y} + g_x - \frac{\partial p}{\partial x} \right) \\ &=: F - \delta t \frac{\partial p}{\partial x}, \\ v^{(n+1)} &= v^{(n)} + \delta t \left(\frac{1}{\text{Re}} \Delta v - \frac{\partial(uv)}{\partial x} - \frac{\partial(v^2)}{\partial y} + g_y - \frac{\partial p}{\partial y} \right) \\ &=: G - \delta t \frac{\partial p}{\partial y}, \end{aligned} \quad (15.7)$$

where obviously the auxiliary quantities F and G as well as the pressure p on the right hand sides have to be assigned to unique time instances (i.e., t_n or t_{n+1}) as well. Doing so now leads us to the so-called time-discrete momentum equations (discretized only in time),

$$\begin{aligned} u^{(n+1)} &= F^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial x}, \\ v^{(n+1)} &= G^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial y}. \end{aligned} \quad (15.8)$$

One therefore has an explicit time discretization with respect to velocity and an implicit time discretization with respect to pressure. In particular, the new velocity field can only be computed after the new pressure has been computed. This can be achieved via the continuity equation. To this end, if one uses (15.8) to derive expressions for the derivatives $\partial u^{(n+1)} / \partial x$ as well as $\partial v^{(n+1)} / \partial y$ and substitutes these into the continuity equation (third equation of (15.2)), we then obtain

$$\frac{\partial u^{(n+1)}}{\partial x} + \frac{\partial v^{(n+1)}}{\partial y} = \frac{\partial F^{(n)}}{\partial x} + \frac{\partial G^{(n)}}{\partial y} - \delta t \Delta p^{(n+1)} \stackrel{!}{=} 0$$

or a Poisson equation for the pressure

$$\Delta p^{(n+1)} = \frac{1}{\delta t} \left(\frac{\partial F^{(n)}}{\partial x} + \frac{\partial G^{(n)}}{\partial y} \right). \quad (15.9)$$

Putting this all together, the time step $t_n \rightarrow t_{n+1}$ thus consists of three substeps: (1) the determination of the auxiliary quantities $F^{(n)}$ and $G^{(n)}$ from the current velocity field $(u^{(n)}, v^{(n)})^T$, (2) the solution of the above Poisson equation for the pressure and (3) the determination of the new velocity field $(u^{(n+1)}, v^{(n+1)})^T$ through use of the now updated new pressure values $p^{(n+1)}$ according to (15.8). Boundary conditions that are suitable for the pressure updated in the second substep

are provided, for example, by the so-called *Chorin's projection method*—here, homogeneous Neumann boundary conditions ensure that the normal component is not changed due to the correction in (15.8).

15.4.2 Spatially Discrete Momentum Equations

Next in line for the completion of the discretization is the spatial discretization of the time-discrete momentum equations. Using the difference quotients for the individual derivative terms derived in Sect. 15.3.2, (15.8) becomes

$$\begin{aligned} u_{i,j}^{(n+1)} &= F_{i,j}^{(n)} - \frac{\delta t}{\delta x} \left(p_{i+1,j}^{(n+1)} - p_{i,j}^{(n+1)} \right), \\ v_{i,j}^{(n+1)} &= G_{i,j}^{(n)} - \frac{\delta t}{\delta y} \left(p_{i,j+1}^{(n+1)} - p_{i,j}^{(n+1)} \right), \end{aligned} \quad (15.10)$$

where for the first case, $i = 1, \dots, N_x - 1, j = 1, \dots, N_y$ and for the second, $i = 1, \dots, N_x, j = 1, \dots, N_y - 1$.

15.4.3 Spatially Discrete Poisson Equation for the Pressure

With this, the momentum equations are now available in a completely discretized form. The discretization of the continuity equation, or rather of its respective new form given in (15.9) as a Poisson equation for the pressure, must still be performed. This task, i.e., the discretization of a Poisson equation by means of finite differences, is at this junction no longer a feat and has been discussed comprehensively in the previous chapter on heat transfer. With the well-known *5-point stencil* and discretized Neumann boundary conditions one obtains a system of linear equations.

In consideration of the theme for this part of our book, “take-off toward number crunching”, naturally some comments regarding the resulting computational effort are presently in order. As one easily and quite plainly sees, the computational effort is primarily necessary (at every time step) for the solution of the pressure equation. Therefore, its efficient solution plays an essential role. Possible candidates for iterative solution methods have already been introduced in Sect. 2.4.4 in the chapter on the basics. In fact, the *SOR-method* is an example of a widely used solver. But one only reaches real efficiency if one employs the *multigrid principle* which has been briefly introduced in the previous chapter. Once more, the objective is of course obtaining a convergence speed that is independent of the fineness of the discretization which, in the practice of numerical fluid dynamics, is anything but an easy task; the mentioning of the key words, strong convection and complicated geometries, may suffice here as an ample explanation.

15.4.4 Regarding Stability

Beginning with Chap. 2, we have encountered the term numerical stability multiple times. In the case of the numerical simulation of time-dependent flow phenomena, the objective to obtain a stable algorithm requires the adherence to *stability conditions* that establish a strong relationship between the mesh widths h_x and h_y of the spatial grid and the time step width δt . This is quite plausible: After all, it would be quite surprising if it was reliable and not subject to problems if a particle that swims in the fluid is allowed to traverse several spatial cells in a single time step. However, such stability conditions are extremely annoying since one cannot simply increase the spatial resolution (for example, in order to capture small vortices), without simultaneously switching to shorter time steps as well.

Beyond question, the most famous stability conditions are the so-called *Courant-Friedrichs-Lowy conditions (CFL conditions)*,

$$|u_{\max}| \delta t < h_x , \quad |v_{\max}| \delta t < h_y , \quad (15.11)$$

which prevent a fluid particle from jumping across a cell, i.e., the case just mentioned above. Another popular condition reads,

$$\delta t < \frac{\text{Re}}{2} \left(\frac{1}{h_x^2} + \frac{1}{h_y^2} \right)^{-1} .$$

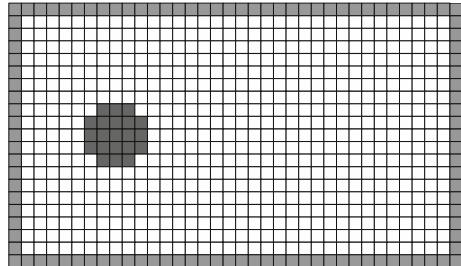
These restrictions governing the time step width brings about a certain level of attractiveness for the implicit methods which admittedly are more expensive at the individual time step but in general permit for larger time steps.

15.5 Application Example: Flow Around an Obstacle

A good number of prominent flow scenarios can already be simulated with a program that is based on the model- and simulation-technical foundations previously discussed. As examples, included are the so-called *driven cavity*—a container filled with a fluid where a tape is pulled with constant velocity along one of its sides—or the *flow over a backward facing step*—best to be illustrated with the help of flow across an embankment.

Special significance is certainly attributed to the flow around a circular or cylindrical obstacle since this scenario has been used as a *benchmark problem* [56] and as a consequence is often used for validation purposes. As mentioned initially (see Fig. 15.1), the resulting appearance of the flow depends on the respective fluid and its material properties as well as the inflow velocity. If the Reynolds number exceeds values of ca. 40, then the flow field becomes non-symmetric and unsteady.

Fig. 15.4 Approximate embedding of an obstacle geometry in a surrounding rectangular domain



In the wake of the obstacle, vortices detach themselves, alternating at the lower or upper boundary of the circle or cylinder.

How do we now map this scenario in the two-dimensional case? To this end, we must on one side prescribe boundary and initial conditions while on the other side we must find a way to describe the object (which is fixed in space)—so far, we have only considered rectangular domains that were entirely filled by the fluid.

The question regarding the boundary and initial conditions can be quickly answered. At the top and bottom boundary no-slip conditions hold while at the left boundary an inflow condition holds in which typically a *parabolic inflow profile* is chosen (zero at the top and bottom, maximal in the middle of the flow canal), and at the right boundary one sets an outflow condition. To start, the velocities in the interior are zero.

Concerning the representation of general *obstacle geometries*, one possibility is to embed the domain Ω of the actual flow into a surrounding rectangular domain which is, as previously described, subdivided into uniform cells. These cells are then divided into two disjoint subsets of *fluid cells* as well as *obstacle cells*. All cells that also lie in Ω , i.e., they contain a flow either entirely or at least partially, are then fluid cells while the others are obstacle cells in which we also want to count the artificial boundary layers (see Fig. 15.4). Besides the exterior boundary (the boundary of the rectangle) we now also have to deal with *interior boundaries*. To formulate the boundary conditions that are required there, one denotes as *boundary cells* all obstacle cells which are neighboring at least one fluid cell. What values are required and their placement obviously depends once more on the respective boundary condition (in our example of the flow around a circle these are typically no-slip conditions) as well as on which variable is placed where. A detailed consideration would go too far here, but we summarize that through use of this technique almost arbitrary obstacle geometries can be represented. However, the approximation accuracy is naturally restricted, in particular for curvilinear boundaries as they are obviously at hand for our example of a circular or cylindrical geometry. Nevertheless, there do exist suitable interpolation approaches as a remedy to this problem.

15.6 Outlook

As previously mentioned at the beginning of this chapter, the world of numerical fluid dynamics is far more comprehensive than it was possible to convey in the previous demonstrations. For this reason, we want to provide at least a somewhat more extended insight into the topic of fluid flow simulation. We again begin with the usual questions and the required model extensions and briefly address different discretization alternatives.

15.6.1 Problem Settings and Models

Previously mentioned was the *inviscid case*, typically hand in hand with *compressible fluids* (gases at high velocities). In general, a relative of the Navier–Stokes equations is here employed, the *Euler equations*. Airplanes capable of flying at supersonic speed are a prime example for a corresponding scenario.

A frequent extension of pure flow consists of the addition of *energy transport* (heat) or *mass transport* (concentration of substances dissolved in the fluid). In the case of heat transport, the model must include an additional conservation principle, the *conservation of energy* principle, that leads to the additional equation,

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T = c \Delta T + q \quad (15.12)$$

for the new independent variable T , the temperature. In order to avoid the need to take into consideration the effects of the temperature on all the other involved variables, one usually employs a *Boussinesq approximation* which limits the effect of the temperature. For example, effects of the temperature on the density are only allowed for the buoyancy force. This *transport equation* (15.12) is a particular example of the more general *convection-diffusion equation*, in which here the term involving the temperature gradient represents the convective part and the Laplace operator of the temperature the diffusive part. The parameter c determines the proportion between convection and diffusion, and the *source term* q represents an (external) heat supply or sink. With this model extension one can, for example, simulate the influence that a heated wall has on the flow and heat distribution in a water tank. The modeling of mass transport leads to an equation analogous to (15.12),

$$\frac{\partial C}{\partial t} + \mathbf{u} \cdot \nabla C = \lambda \Delta C + Q(t, \mathbf{x}, \mathbf{u}, C), \quad (15.13)$$

where the source term Q may now depend on location, time, velocity and concentration. This opens up the general approach for the treatment of interactions

of flows with chemical reactions, in particular, if one couples a transport equation to each of several dissolved substances. The characteristics of the chemical reactions of the substances with each other is then modeled via the different source terms.

Another model extension is provided by so-called *free boundary value problems*. In these, the shape of the flow domain can change in time. This class of problem setting includes on one hand problems with a *free surface* such as a water drop falling and plunging into water, a breaking dam, the flow across a step with a free surface, such as it is for example given at an embankment, or the technique of die casting. On the other side, one also speaks of free boundary value problems for problems with *phase transitions* such as melting or freezing processes. Also in this respect one can extend the introduced model by means of two essential ingredients: a method for the update of the simulation domain (employed methods are here for example *particle methods*, the *volume-of-fluid method* or *level-set methods*) as well as a possibly precise description of the activities at the free surface or at the phase transition (here, quantities such as the curvature play a decisive role).

To clarify what was already hinted implicitly: Multiphase flows also present a highly relevant extension of the pure flow model. Technically, one is concerned with the identification of several fluid phases and the precise modeling of the phase boundary—whether there is a phase transition as it can be found at the boundary between water and ice, or a mixing such as for differently colored water, or there are no effects across the boundary.

If one not only couples different fluid phases, but couples a fluid together with a structure (which surrounds it or which it flows around) as well, then one gets *fluid-structure interactions* or, generalizing to other physical effects, *coupled problems* or *multiphysics problems*. For these there exist two different approaches: The *monolithic* approach in which case the path of a model extension is actually chosen followed by the discretization of the extended model; the *partitioned* approach in which in contrast one uses existing models and solvers for the individual effects and then couples these in some suitable way.

This still leaves the phenomenon of *turbulence* for which the few remarks already made at the beginning of this chapter as well as in Chap. 1 should suffice.

15.6.2 Discretizations

Even the attempt to list the discretization schemes that are used in the context of CFD is doomed to failure—too large is its diversity. Thus we are content at this juncture with the hint suggesting that the method introduced in this chapter is only one among many others and as the others, it has its advantages (for example the simplicity of the approach—even if readers who engage themselves with this topic for a first time may not share this impression), but also its disadvantages (for example a lower theoretical foundation in comparison to finite-element approaches). Furthermore, let us point out that in some cases numerical stability requires the

transition to variants (for example, in the case of larger Reynolds numbers already mentioned).

A hot topic for numerical fluid dynamics is also the problem-adapted local refinement of the discretization, the *mesh adaption*—but more on this is reserved for the next section!

Appendix: Small Excursion into Grid Generation

The final section of this chapter shall deal with grid generation—a topic which in general is of importance for the discretization of PDEs and as such it must not remain unmentioned, especially in the CFD-context.

There exists a multitude of strategies for the approximate description and discretization of arbitrary computational domains and thus directly related, the *generation* as well as *refinement* of suitable *grids* or *meshes*. The respective choice naturally depends on the selected discretization scheme for the PDE as well as on the complexity of the domain. In real applications this step is often very time-consuming—of significance is that this step typically requires user interaction and as important, the expertise of the user.

The grid generation requires a couple of aspects to be taken into consideration: the *accuracy* (the grid must allow for the underlying physics to be represented with sufficient accuracy), the *boundary approximation* (the grid must reflect the geometry of the domain with sufficient accuracy), the *efficiency* (the overhead for the grid organization (storage, computational time) should be low and not get into the way of an efficient implementation on supercomputers) as well as the *numerical compatibility* (occurrences with negative effects such as too small or too large angles in triangles or extreme distortions should be avoided).

In general, one distinguishes between *structured* and *unstructured* grids. In the first form, neither geometrical nor topological information (i.e., coordinates or neighborhoods) have to be stored explicitly since they can be derived from the grid structure. Unstructured grids do not allow this, but they are more flexible with the positioning of the points, the shape of the cells as well as the approximation of the geometry making up the computational domain. In both cases, the generation of the initial grid represents only a first step since the optimal positioning of points is hard to predict and strongly depends on the solution that is to be computed, illustrating the necessity of *adaptive grid refinement*.

Finally, many parallelization strategies for PDEs are based on a decomposition of the computational domain (*Domain Decomposition*), which brings up additional questions: How can the computational load be distributed simply and effectively onto several processors and how can the communication effort between the resulting subgrids be kept small (small interfaces, strategies for graph partitioning)?

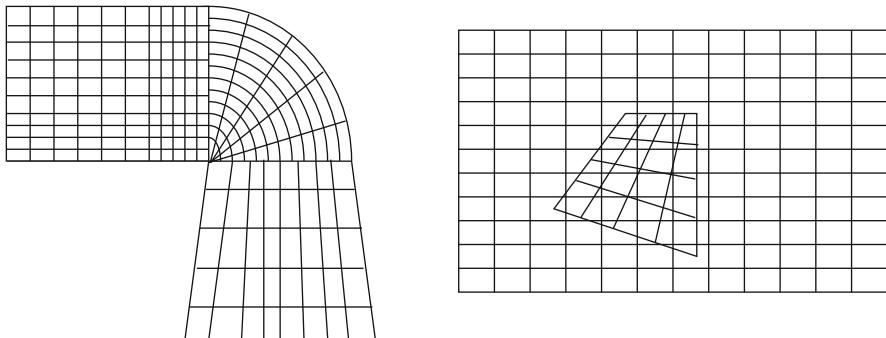


Fig. 15.5 Composite grids: Patches (*left*) and Chimära (*right*)

15.6.3 Structured Grids

Composite Grids

The *composite grid* concept is the key to the treatment of general three-dimensional geometries with structured grids. The idea is simple: Decompose the computational domain into several subdomains of simpler form such that a structured grid generation is possible in the respective subdomains without difficulty. This will hardly lead to the goal if a porous medium is given as a computational domain, but it will indeed work for many of the technical objects that have been designed by means of a CAD system. One distinguishes two essential variants (see Fig. 15.5): *patch grids*, in which different structured subgrids are joined together along interfaces (with or without continuous transition), as well as *Chimera grids* in which completely independent and overlapping subgrids overlay a basic grid in the background or even with other component grids. Here, interpolation becomes very important.

Block-Structured Grids

Block-structured grids are very popular, especially in numerical fluid dynamics. Here, the underlying domain is composed in an unstructured way using several *logically rectangular* subdomains. These result from a transformation of a reference rectangle and are supplied with a grid determined by applying the same transformation to a Cartesian grid within the original rectangle. Thus, the subdomains possibly have curvilinear grid lines but are nonetheless rooted to the topological structure of a rectangle. Continuity at the transition between subdomains is required here—in particular, the points must coincide there (see Fig. 15.6). The block-structured approach thus combines the advantages of boundary-adapted and strictly structured grids.

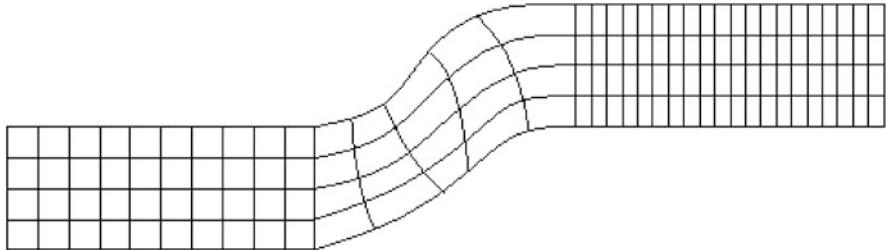


Fig. 15.6 A block-structured grid

Elliptic Generators

Numerous grid generation techniques are based on the idea of a mapping or *transformation* $\Psi : \Omega' \rightarrow \Omega$ from the unit square or unit cube Ω' to the computational domain Ω . It is obvious that the respective transformation must also be applied to the PDE. A famous representative of this class are *elliptic generators*, in which the grid lines or coordinates of the computational domain are obtained from the solutions of elliptic differential equations. This ensures that one obtains very smooth grid lines in the interior, even in the case of non-smooth boundaries. In contrast to this advantage, however, is the relatively high cost and the fact that often it is very difficult to control the grid points and grid lines.

We consider a simple two-dimensional example: A grid is to be generated on the (logically rectangular) domain Ω , which is bounded by the four curves $c_{0,y}$, $c_{1,y}$, $c_{x,0}$ and $c_{x,1}$ (see Fig. 15.7). Now define a system of Laplace equations for the components $\xi(x, y)$ and $\eta(x, y)$ of the mapping $\Psi(x, y)$,

$$\begin{aligned}\Delta \xi(x, y) &= 0 && \text{on } \Omega' =]0, 1[^2, \\ \Delta \eta(x, y) &= 0 && \text{on } \Omega',\end{aligned}\tag{15.14}$$

with Dirichlet boundary conditions,

$$\begin{aligned}\begin{pmatrix} \xi(x, 0) \\ \eta(x, 0) \end{pmatrix} &= c_{x,0}(x), & \begin{pmatrix} \xi(x, 1) \\ \eta(x, 1) \end{pmatrix} &= c_{x,1}(x), \\ \begin{pmatrix} \xi(0, y) \\ \eta(0, y) \end{pmatrix} &= c_{0,y}(y), & \begin{pmatrix} \xi(1, y) \\ \eta(1, y) \end{pmatrix} &= c_{1,y}(y).\end{aligned}\tag{15.15}$$

As a solution of the above (auxiliary) equations, one obtains suitable curvilinear grid lines on Ω .

One disadvantage of this approach is that in the non-convex case it is possible for the grid lines to leave the domain. Here, *inverse elliptic methods* help out in which the Laplace equations are defined on the curvilinearly bounded domain Ω and subsequently are mapped to the reference domain. The price to pay is that a more complicated and coupled system of PDEs must now be solved.

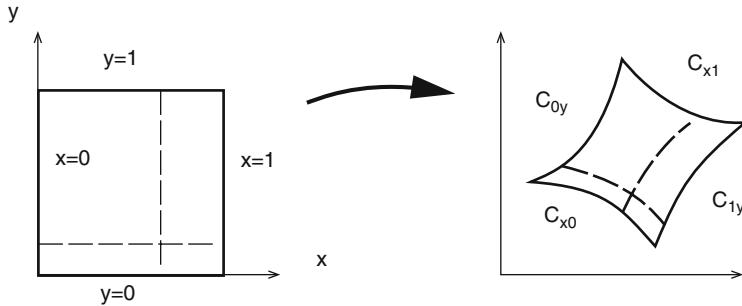


Fig. 15.7 Mapping and elliptic grid generation

Hyperbolic Generators

In place of elliptic systems, computational grids may also be constructed via the solution of suitable hyperbolic systems. However, because of the character of hyperbolic PDEs only one boundary side with respect to each coordinate can be specified—which in particular makes this technique appear attractive for (one-sided) unbounded domains. One often observes advantages with respect to speed in comparison to the elliptic case.

Algebraic Generators

Algebraic or *interpolation-based* generators get by without the solution of an additional system of PDEs, they are instead based on simple interpolation techniques. The most famous representative is the so-called *transfinite interpolation*, which was, for example, used in the *Coons patch*. We again restrict ourselves to the two-dimensional case and determine a grid for the domain Ω given in the right of Fig. 15.7 whose boundaries are prescribed through the four curves $c_{0,y}, \dots, c_{x,1}$. All four curves can be summarized by a single definition $c(x, y)$ where at first $c(x, y)$ is given only for $x \in \{0, 1\}$ or $y \in \{0, 1\}$. Next, one introduces the three interpolation operators F_1 , F_2 and $F_1 F_2$ which turn $c(x, y)$, defined only on the boundary of Ω' , into a function defined on all of Ω' :

$$\begin{aligned} F_1(x, y) &:= (1-x) \cdot c(0, y) + x \cdot c(1, y), \\ F_2(x, y) &:= (1-y) \cdot c(x, 0) + y \cdot c(x, 1), \\ F_1 F_2(x, y) &:= (1-x)(1-y) \cdot c(0, 0) + x(1-y) \cdot c(1, 0) \\ &\quad + (1-x)y \cdot c(0, 1) + xy \cdot c(1, 1). \end{aligned} \tag{15.16}$$

Finally, the operator TF of the transfinite interpolation is defined as

$$TF(x, y) := (F_1 + F_2 - F_1 F_2)(x, y). \tag{15.17}$$

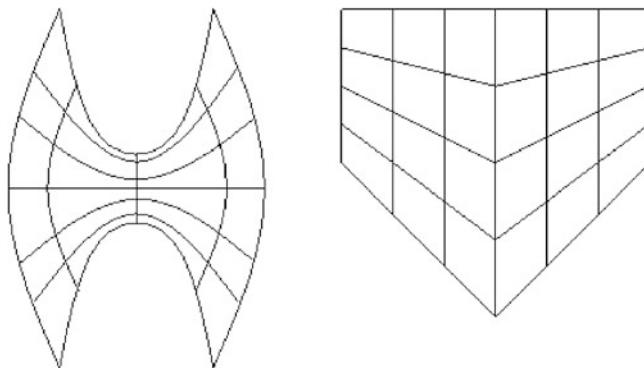


Fig. 15.8 Transfinite interpolation

The transfinite interpolation allows for a very cheap and simple control of the grid generation. Non-smooth boundaries, however, pass on this property to the interior. Unfortunate as well is that grid lines can once more leave the domain. Two examples illustrate the transfinite interpolation and are given in Fig. 15.8.

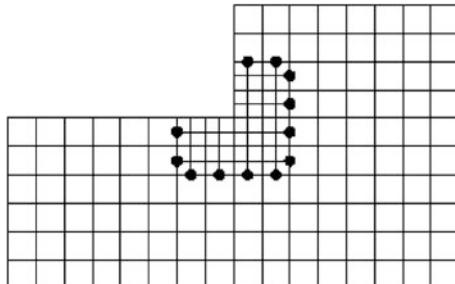
Adaptive Grids

Many numerical simulations require the use of locally adaptive grid refinements in the course of the computation, for example, in the case when one reaches a point at which a global refinement step would be uneconomical yet one cannot terminate the computation for reasons of accuracy. One could also, of course, locally employ a method of higher order, but this option does not concern the grid refinement which is what interests us in this section. In the case of structured grids, one usually chooses the *block adaption*: Identify the critical subdomains with the help of an *error indicator* and begin a grid refinement in the affected block (e.g., by dividing the mesh widths into halves). Particular attention has to be devoted to the so-called *hanging nodes*, i.e., grid points resulting from the refinement process and really only exist from the perspective of one side of an edge. Finally, discontinuities shall be avoided in general. See Fig. 15.9 for a simple example.

15.6.4 Unstructured Grids

Unstructured grids stand in close relationship with finite element methods, and one usually associates them with (almost) arbitrary triangulations or tetrahedral grids.

Fig. 15.9 Block adaptation and hanging nodes: If a node marked with a *black circle* carries a value different from zero, then this value exists only from the perspective of the finer grid



Delaunay Triangulations and Voronoi Diagrams

If grid points are known, a quite old method for the generation of unstructured finite element meshes goes back to Dirichlet and Voronoi. For a given point set $P_i, i = 1, \dots, N$, the *Voronoi domains* V_i are defined as

$$V_i := \{P : \|P - P_i\| < \|P - P_j\| \forall j \neq i\} . \quad (15.18)$$

Thus, V_i contains all points that are closer to P_i than any other grid point P_j . Altogether, one obtains the *Voronoi diagram*, a subdivision of the simulation domain into polygons or polyhedra. If in the Voronoi diagram one draws lines between all neighboring points (i.e., points with neighboring Voronoi domains), then a set of disjoint triangles or tetrahedra is formed that covers the convex hull of the P_i —the *Delaunay triangulation*; see Fig. 15.10 for a simple example. Delaunay triangulations are frequently used since they possess a number of properties favorable for both the efficient generation as well as later in the numerical solution process on the grid. In the following, we will use the term triangulation regardless of the spatial dimension d .

Point Generation

And how does one obtain suitable sets of gridpoints? A first possibility is provided by the *independent generation* in which the grid points of a coarse structured grid (which has nothing to do with the later computation) are used as a starting set. The second approach—*superposition and continued subdivision*—also uses a structured auxiliary grid and refines it successively (here, e.g., *octrees* are common). Alternatively, one can start with a *boundary-related triangulation* (for example, a Delaunay construction beginning with points only on the boundary) and then successively refine it. Here, in addition to the points on the boundary, one can systematically include points or lines, etc. that are in the interior into the construction process (if for example singularities are known).

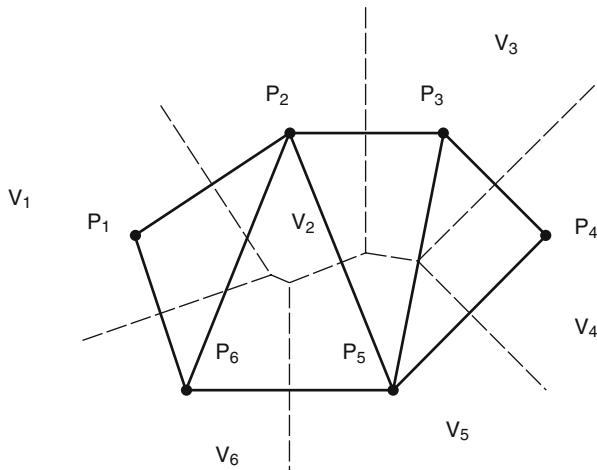


Fig. 15.10 Voronoi diagram (dashed) and Delaunay triangulation (solid) in 2 D

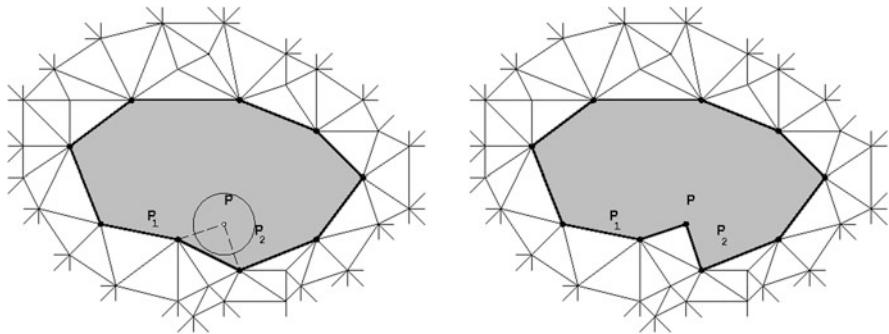


Fig. 15.11 One step in the advancing-front algorithm: Generation of a new interior point as well as an associate triangle

Alternatives

Naturally, the Delaunay construction is not the only one in existence. *Advancing-front methods* start with a discretization of the boundary and from there use a systematic injection of new points to move a front into the interior of the domain until this is completely meshed. Figure 15.11 shows one step of the respective algorithm.

Spacetrees (*Quadtrees* in two dimensions, *Octrees* in three dimensions) are based on a recursive uniform space partitioning of a d -dimensional cube until either a given tolerance or maximal depth is reached or until a cell lies completely inside or completely outside the simulation domain. In order to obtain a grid that is adapted to the boundary, the plain spacetree has to be modified by suitable cutting and reconnecting to triangles or tetrahedra.

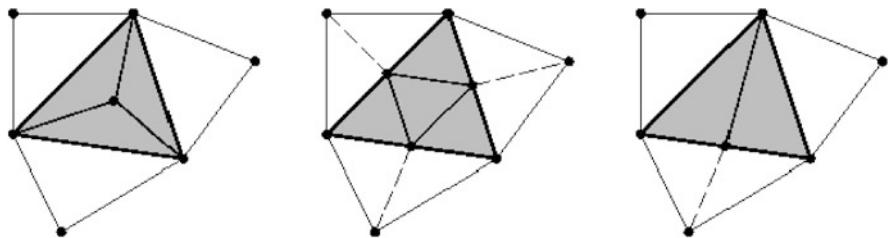


Fig. 15.12 Some refinement strategies (including the treatment of hanging nodes): addition of the center of gravity (left), red refinement (middle) as well as green refinement (right)

Finally, *hybrid grids* are hybrid constructions which bear within them the properties of both structured as well as unstructured grids.

Adaptive Grids

In the context of unstructured grids, adaptive grids represent to a certain extent the normal case since one already has to deal with elements of different size as well as with dynamic data structures. For this reason, adaptive grid refinement is more popular here than in the structured case. In order to be able to execute the adaptation effectively and efficiently, one requires four parts: First, one needs a *local error estimator* or *error indicator*, which in the course of the computation recommends local refinement at certain points. Second, a *refinement criterion*, which specifies the refinement objective (a distribution of the error as uniform as possible across all points, bounds for the error everywhere etc.) is needed. Third, one needs a (technical) method to implement the refinement (Fig. 15.12 shows a few possibilities for this) as well as fourth, a *global error estimator*, which gives insight on the accuracy of the current approximate solution. For all four parts, a multitude of suggestions have been made which would go beyond the scope of our small excursion.

15.6.5 Approaches for the Treatment of Changing Geometries

As indicated in the previous sections, changing geometries, and hence the techniques for their representation, play an ever increasing role—one thinks of free surfaces, multiphase flows or fluid–structure interactions. A permanently new meshing after every change of the geometry is in many cases obviously beyond current and future computer capacities. Therefore, those approaches gain greater importance that go beyond the repeated application of the methods introduced above.

Front-Tracking Methods

The so-called *front-tracking methods* describe the boundary or the *interface* between two phases (liquid–liquid or liquid–solid) directly, i.e., they permanently update a geometric representation, for example via *free-form surfaces* according to the current movements. This is obviously very precise, but also subject to considerable computational effort or even fundamental problems (for changes in the topology, if e.g. two bubbles unite), which is why this approach often does not provide a feasible path.

Front-Capturing Methods

Front-capturing methods are different: They describe the position of the interface only implicitly via a suitable global quantity. This implies lower accuracy but better feasibility. The *marker-and-cell* and *volume-of-fluid approaches* (VoF) previously mentioned fall within this category. For the first of them, particles are moved; cells without particles do not belong to the fluid, cells with particles which are neighboring empty cells belong to the surface or to the interface, and all other cells belong to the fluid. The VoF-method uses a global quantity, the volume portion of the fluid relative to the volume of the cell, in order to describe the position of the surface or the interface.

Clicking- and Sliding-Mesh Techniques

For regular or periodic movements (oscillations or the slow rotation of a mixer) it is intuitive to keep the basic grid of the flow fixed in space and only move the moving structure along with the neighboring fluid cells. This part of the grid *slides* further, and a *click* switches the neighborhood relationships, etc.

A last remark shall conclude this excursion: Global grid generation is a problem occurring most often at the so-called *Eulerian perspective* which assumes a fixed reference system. An alternative is the *Lagrangian perspective*, in which all “players” (e.g., moving structures) have their own local perspective. In a certain way, the *Arbitrary Lagrangian–Eulerian (ALE)* approach combines both perspectives and therefore simplifies the handling of moving structures and fluid–structure interactions.

Chapter 16

Global Illumination in Computer Graphics

In this chapter, we turn to a primordial computer science application of modeling and simulation—yet simultaneously we deal with physics on the computer. After all, one of the biggest goals of computer graphics is *photo realism*, i.e., the generation of computer pictures that appear as realistic as possible. We thereby will encounter models in many places—the description of objects and effects—as well as simulations—with their efficient graphical representation. As examples, let us mention the representation of natural objects (mountains, trees etc.) by means of fractals or grammar models, the representation of natural effects (fire, fog, smoke, folds in cloth etc.) by means of particle systems, the representation of biomechanical procedures (dinosaur bellies and other ‘wobbling masses’) or in general, the description of animation. Furthermore, of central importance for computer graphics are techniques for *global illumination*. This is what we will concentrate on in the following. Here, the representation follows [10]. Required are the tools of Chap. 2, in particular the sections regarding analysis and numerical analysis.

In computer graphics, one has studied *local* illumination models from early on in order to describe lighting conditions at a given point of a scene, and thereby made a distinction between ambient light, point light sources with diffusive reflection and point light sources with specular reflection. For the generation of photo-realistic images, the modeling of *global illumination*, i.e., the interactions of light between all objects of the scene, is important as well.

The first, the so-called *ray-tracing* approach, goes back to Whitted and Appel [4, 58]. It can perfectly describe specular reflection, but fails completely with diffuse illumination or ambient light. The resulting images appear synthetic and almost too perfect. In a certain sense the counterpart, the *radiosity method*, has been introduced by Goral et al. [27] and perfectly describes diffuse illumination but specular reflections are not possible. The resulting images appear more natural, but not yet realistic—after all, the specular and glossy effects are missing. Subsequently, numerous approaches that combine ray-tracing and radiosity have been developed, beginning with a simple consecutive execution up to more complicated methods.

Yet there remain problems in the description of indirectly specular illuminations (for example via a mirror), so-called *caustics*. Further steps in the development were *path-tracing* [36] and *light or backward ray-tracing* [5]. In principle, both are able to solve the global illumination problem, but have problems with respect to the effort or restrictions regarding the description of the geometry. A significant improvement is provided by *Monte Carlo ray-tracing with photon maps* [35], which can also efficiently describe caustics with satisfactory quality.

16.1 Quantities from Radiometry

Radiometry is the science of the (physical) measurement of electromagnetic energy. Some important quantities from this field are of importance in what follows. Here, the reference quantity is a surface A in the scene to be represented or illuminated. Let $x \in A$ denote a point on the surface, $\omega \in S^2$ a beam direction with the unit sphere S^2 (the surface of the sphere with its center in the origin and radius one) or $\omega \in H^2$ with the unit hemisphere H^2 . Additionally, the concept of a *solid angle* is needed that, in general, is measured via the area on the surface of H^2 , i.e., it assumes a maximal value of 2π or $2\pi sr$ with the artificial unit *steradian*. When considering a region on a hemisphere with arbitrary radius r , the solid angle is simply found by dividing the area of the region by r^2 . Finally, the angle $\theta = \theta(x, \omega)$ denotes the angle between the region's normal in x and the beam direction ω (see Fig. 16.1). We now turn to the radiometric quantities and in the following introduce the terminology consistent with that found in the general literature:

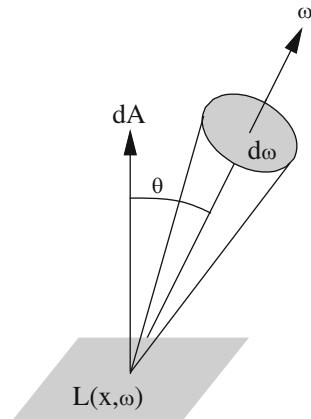
- By *radiant energy* Q , one denotes the electromagnetic energy or light energy (unit Joule).
- The *radiant flux* Φ is understood to be the energy flow, input or output per unit time (unit watts; typically, only the static case is considered from which the terms radiant energy and radiant flux are often used synonymously):

$$\Phi := \frac{\partial Q}{\partial t}.$$

- *Radiance* $L(x, \omega)$ denotes the radiant flux in an infinitesimally thin beam or the radiant flux per unit area, $dA \cos \theta$, perpendicular to the beam and per solid angle, $d\omega$, in the beam direction ω (unit W/m^2sr). The quantities L_i , L_o , L_r , L_e denote the respective incoming (in), outgoing (out), reflected and emitted radiance.
- The *irradiance* E is understood to be the impacting radiant flux per unit area (unit W/m^2 ; summation or integration with respect to all incoming directions ω_i):

$$dE := L_i(x, \omega_i) \cos \theta_i d\omega_i, \quad E(x) := \int_{H^2} dE.$$

Fig. 16.1 Oriented surface and solid angle



- Conversely, *radiosity* B denotes the outgoing radiant flux per unit area (also unit W/m^2 ; summation or integration now with respect to all outgoing directions ω_o):

$$dB := L_o(x, \omega_o) \cos \theta_o d\omega_o, \quad B(x) := \int_{H^2} dB.$$

- Radiant intensity* I denotes the outgoing radiant flux per solid angle, i.e., per direction in space (unit W/sr ; summation or integration over all area elements A , i.e., over $\Omega := \cup A$):

$$dI := L_o(x, \omega_o) \cos \theta_o dA, \quad I(\omega_o) := \int_{\Omega} dI.$$

The relationships between these quantities become apparent through another integration:

$$\begin{aligned} \Phi_o &= \int_{H^2} I(\omega_o) d\omega_o = \int_{\Omega} B(x) dA = \int_{\Omega} \int_{H^2} L_o(x, \omega_o) \cos \theta_o d\omega_o dA, \\ \Phi_i &= \int_{\Omega} E(x) dA = \int_{\Omega} \int_{H^2} L_i(x, \omega_i) \cos \theta_i d\omega_i dA. \end{aligned}$$

In the case of a complete reflection (no transparency or refraction, no absorption) of a surface it is always true that $\Phi_o = \Phi_i$.

The radiance has two important properties: It is constant along a beam as long as it does not hit any surface, and it is the essential quantity for the response of a light-sensitive sensor (cameras, eye).

Finally, all the quantities introduced depend on the *wave length* of the light. Theoretically, one additional integral with respect to the wave length is added and in practice one restricts oneself to the three-dimensional RGB-vector (R, G, B) for the base colors Red, Green and Blue—a common *color model* in computer graphics.

16.2 The Rendering Equation

We briefly have to bring to mind an important issue in local illumination. Here, the description of *reflection* is essential for the modeling of the optical appearance of surfaces. The reflection depends on the direction of the incoming and outgoing light, the wave length of the light as well as the respective point on the surface. The most important means used to describe this are through the *bidirectional reflection distribution functions (BRDF)* $f_r(x, \omega_i \rightarrow \omega_r)$ which quantify the dependence of the outgoing (reflected) light on the incoming light intensity:

$$dL_r = f_r(x, \omega_i \rightarrow \omega_r) dE = f_r(x, \omega_i \rightarrow \omega_r) L_i(x, \omega_i) \cos \theta_i d\omega_i . \quad (16.1)$$

One notes that reflected radiance is paired with the incoming irradiance, not with the incoming radiance (the justification for this would demand too much detail at this junction). In general the BRDF are anisotropic, that is, a rotation of the surface (with unchanged incoming and outgoing directions) can lead to a change in the reflected amount of light.

From (16.1), one can now derive the so-called *reflectance equation*

$$L_r(x, \omega_r) = \int_{H^2} f_r(x, \omega_i \rightarrow \omega_r) L_i(x, \omega_i) \cos \theta_i d\omega_i , \quad (16.2)$$

which establishes the (local) connection between the incoming and reflected light. The BRDF offers one possibility for a very general description of the reflection. For reasons of complexity, in practice one uses simple (local) reflection models for the different classes of BRDF in order to reduce the number of parameters (see Fig. 16.2):

- For an *ideally specular reflection*, an incoming ray is reflected into exactly one direction according to the law of reflection by Snellius. The modeling of the BRDF is performed with the help of Dirac δ -functions.
- For a *practically specular reflection*, light is reflected obeying a distribution which is strongly concentrated around the ideal reflection direction.
- For a *Lambertian* or *diffuse* reflection, the incoming light is uniformly reflected into all directions and hence the corresponding distribution is therefore a uniform distribution.

General reflection or illumination models frequently combine all approaches. Transparency and refraction can easily be incorporated by admitting also the lower or inner hemisphere as possible “reflection directions”. For reasons of simplicity, all integrals in the following are only formulated with respect to H^2 . For every case, however, this only describes the incident light, and what is happening with incident light. How the respective incident light originates, however, is left unanswered and must be performed as an additional step.

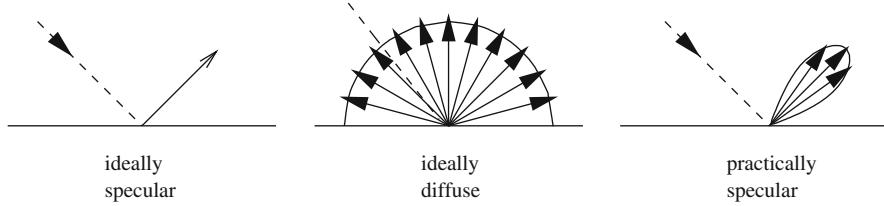


Fig. 16.2 Different reflection behavior and different types of BRDF

How is incident light composed? The clarification of this question initially requires that we consider the local case: Let a point light source in x_s have intensity I_s . The radiance L_i at a point x of the scene is then determined according to

$$L_i(x, \omega) = \begin{cases} I_s \cdot |x - x_s|^{-2} & \text{for } \omega = \omega_s := x - x_s, \\ 0 & \text{else.} \end{cases} \quad (16.3)$$

The integral (16.2) reduces to the expression (formally via a Dirac process)

$$L_r(x, \omega_r) = \frac{I_s}{|x - x_s|^2} f_r(x, \omega_s \rightarrow \omega_r) \cos \theta_s, \quad (16.4)$$

or as a sum over n such terms for n point light sources—admittedly a conceivably primitive illumination model. *Global* illumination is certainly more than the local illumination at all points of the scene. Additionally, the interplay between the individual surfaces must be depicted correctly. This interplay is none other than energy transport: Light sources emit radiance which is reflected, retracted or absorbed in the scene. Next, the amount of light that eventually reaches the eye or camera must be computed. To this end, for every image pixel, the radiance must be integrated over the respective visible surfaces. The resulting radiant flux then defines the brightness and color of the pixel. The radiance that leaves a point x in direction ω_o is composed of (self-) emitted and reflected radiance:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + L_r(x, \omega_o). \quad (16.5)$$

With (16.2), one obtains the *rendering equation* [36]:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{H^2} f_r(x, \omega_i \rightarrow \omega_o) L_i(x, \omega_i) \cos \theta_i d\omega_i, \quad (16.6)$$

which describes a global energy balance for the light conditions of the scene. The geometry of the scene, the BRDF of the surfaces and the light sources (interpreted as purely self-emitting surfaces) completely determine the light distribution.

The rendering equation in its above form (16.6), however, is not yet satisfactory: On one hand, there is the incoming radiance under the integral, yet the outgoing radiance on the left hand side, while on the other hand, the integral over the hemisphere is somewhat cumbersome. To better handle the notation of the rendering equation the *visibility function* $V(x, y)$ is required,

$$V(x, y) := \begin{cases} 1 & : x \text{ sees } y, \\ 0 & : \text{else.} \end{cases} \quad (16.7)$$

Here, let $x \in A_x$ and $y \in A_y$ be two points on two surfaces A_x and A_y of the scene. Given the above V it obviously holds that,

$$L_i(x, \omega_i) = L_o(y, \omega_i) V(x, y). \quad (16.8)$$

In order to change the domain of integration, we relate $d\omega_i$ to the area dA_y where the light originates:

$$d\omega_i = \frac{\cos \theta_y dA_y}{|x - y|^2}. \quad (16.9)$$

With the definition

$$G(x, y) := V(x, y) \cdot \frac{\cos \theta_i \cos \theta_y}{|x - y|^2} \quad (16.10)$$

one then obtains the following second form of the rendering equation:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_i \rightarrow \omega_o) L_o(y, \omega_i) G(x, y) dA_y. \quad (16.11)$$

With the integral operator

$$(Tf)(x, \omega_o) := \int_{\Omega} f_r(x, \omega_i \rightarrow \omega_o) f(y, \omega_i) G(x, y) dA_y, \quad (16.12)$$

(16.11) simplifies to

$$L_o = L_e + TL_o \quad \text{or, in short} \quad L = L_e + TL. \quad (16.13)$$

The *light transport operator* T therefore transforms the radiance on A_y into the radiance on A_x (after *one* reflection). If one applies (16.13) recursively, then one obtains

$$L = \sum_{k=0}^{\infty} T^k L_e, \quad (16.14)$$

where L_e denotes the emitted light, $T L_e$ the direct illumination and $T^k L_e$, $k > 1$, the indirect illumination after $k - 1$ reflections.

To determine the global illumination, the rendering equation taking the form of the integral equation given in (16.6) or (16.11)—this is our model—has to be solved—this is the corresponding simulation. We will deal with this in the next section.

16.3 Techniques for the Solution of the Rendering Equation

Most of the techniques used to solve the global illumination problem work to approximately solve the rendering equation in a more or less direct way. They can be classified by means of their capabilities in the consideration of different types of light interaction (characterized by means of a sequence of reflections on the path from the light source to the eye). With the notation L for the light source, E for the eye, D for a diffuse reflection and S for a specular reflection, an optimal method must have the ability to consider all sequences of a regular expression,

$$L (D | S)^* E \quad (16.15)$$

(see Fig. 16.3).

Because transparency can be distinguished between specular transparency (*transparent*, e.g. glass) and diffuse transparency (*translucent*, e.g. frosted glass), all cases are covered by (16.15). Furthermore, if X denotes a point of the scene then the following illumination types can be distinguished for it:

LX	direct illumination,
$L (D S)^+ X$	indirect illumination,
$LD^+ X$	purely diffuse indirect illumination,
$LS^+ X$	purely specular indirect illumination.

(16.16)

Since the last case is particularly “awful”, the corresponding illumination effects are called *caustics* (greek *Kaustikos*, from *Kaiein*—to burn).

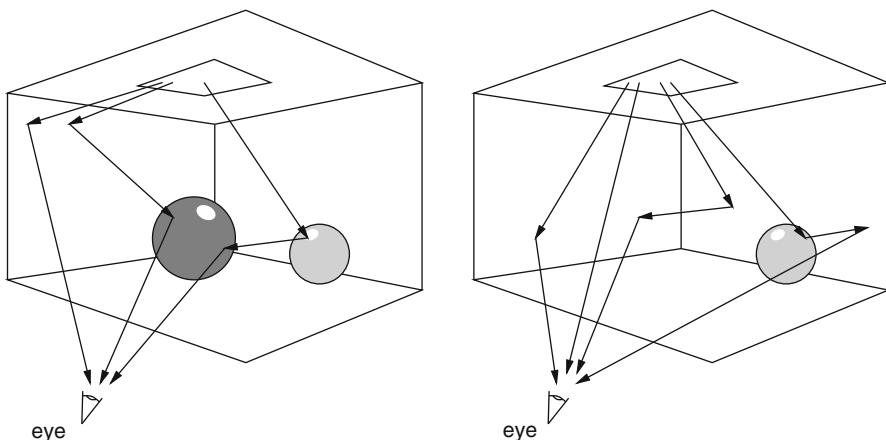


Fig. 16.3 Different light paths: LDE, LDSE, LSSE and LDE, LE, LDDE, LSDE (respectively from left to right)

16.3.1 Ray Tracing

Ray tracing is the oldest and probably simplest method for global illumination. Shading as well as ideal specular reflections and refraction are accounted for, but not diffuse illumination. The algorithmic foundation for this is the same as for ray-casting which is used in computer graphics for the solution of the visibility problem, namely tracing the ray. One traces individual light rays which originate from a viewer's perspective and end at the first intersection with an object at the scene, or possibly never. Energy is transported in the form of radiance.

Through the course of time several algorithmic variants and optimizations have been developed around ray-tracing. We restrict ourselves here to its qualities with respect to global illumination. Originating from the viewer, a ray is shot through every image pixel into the scene (*primary rays*). If a ray passes without intersecting an object, the pixel obtains the background color. Otherwise, three types of *secondary rays* start from the closest point of intersection, x : the perfectly reflected ray (for material that is not entirely dim), the perfectly refracted ray (for transparent material) as well as the so-called shadow rays to all light sources. Each light source which is reached by the respective shadow ray without obstacle (object in the scene), illuminates x directly. The radiance originating from x towards the viewer is then determined recursively from the radiance values of all incoming secondary rays, the respective directions and the BRDF in x according to the local illumination model given in Fig. 16.4. To abort the recursion, a maximal recursion depth or a minimum radiance must be specified. Finally, the missing consideration of diffusive reflection is modeled by a constant term for ambient light.

According to the classification given in Fig. 16.5, ray tracing depicts all paths of the type,

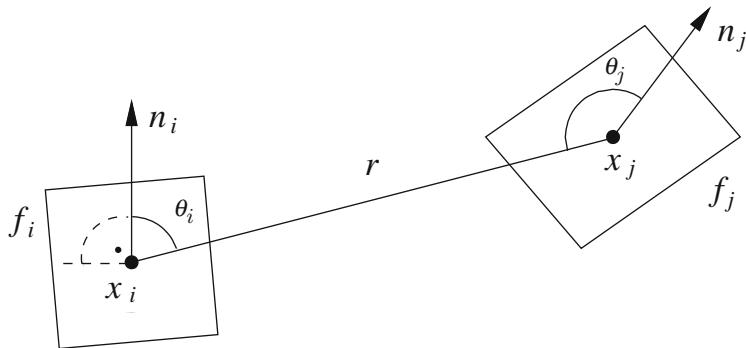


Fig. 16.4 Patches f_i and f_j with specified points x_i and x_j , the surface normals n_i and n_j as well as the angles θ_i and θ_j between n_i or n_j , resp., and the line segment of length r from x_i to x_j

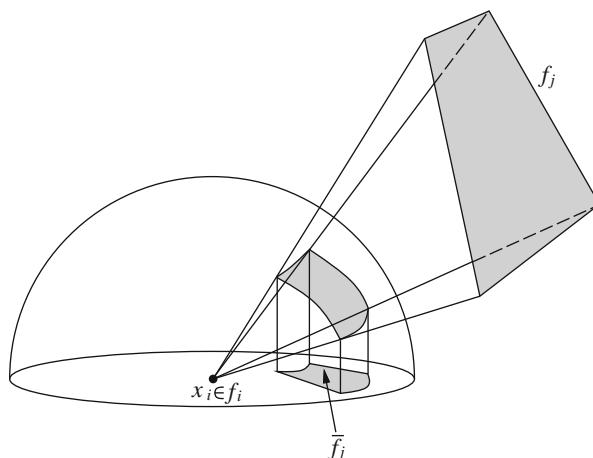


Fig. 16.5 Computation of the form factors via projections

$$LDS^* E. \quad (16.17)$$

The D in (16.17) is required since, seen from the perspective of the light source, the shadow rays can assume arbitrary directions independent of any “incoming direction”. Paths of the type $LS^* E$, however, cannot be represented since light sources are not treated as objects and consequently can only be hit by shadow rays.

As mentioned before, the determination of the outgoing radiance from an intersection point x to an observer is performed on the basis of a local illumination model (16.4). For the original ray-tracing, the incoming radiance was simply multiplied by a reflection- or refraction-coefficient that is independent of the incoming or outgoing directions. Analogous to the rendering equation in the form given by (16.14), we can also describe the function of ray-tracing in its general form as,

$$L = L_0 + \sum_{k=1}^{\infty} T_0^k L_e, \quad (16.18)$$

where L_0 denotes the ambient term. The term L_e is dropped since point light sources are not treated as emitting objects. In contrast to T , T_0 is not an integral but a simple sum operator which is summed over the three terms, direct illumination, reflected light and refracted light. From the perspective of the rendering equation to be solved this means that instead of an integral over *all* incoming directions we sample only a few firmly prescribed directions, namely the perfectly reflected, the perfectly refracted and the ones leading to the light sources.

16.3.2 Path-Tracing

Path-tracing (also called *Monte Carlo Ray-tracing* or *Monte Carlo Path-tracing*) [36] is based on the idea to approach the rendering equation (16.6) directly and to solve it via Monte-Carlo-integration. As for ray-tracing, we replace the integration over all directions by samples taken from a finite number of directions. The direction paths to the light sources are retained (using random rays one cannot hit point light sources even though they are very important), the perfectly reflected and the perfectly refracted ray, however, are replaced by a single ray whose direction is determined randomly (thus *Monte Carlo* as a part of the name). Instead of the cascadic recursion at the ray-tracing this results in a linear recursion, a path of rays (thus *path* as a part of the name). The recursion depth can be prescribed as fixed, chosen adaptively or determined by *Russian roulette*, where each time it is determined randomly whether a path should be terminated or continued (up to the maximal depth—just as for Russian roulette).

Instead of the uniform distribution over all directions one can use a weighted distribution as well that is based on the BRDF which favors those more important directions resulting from the BRDF during the selection process. By means of

$$\rho(x, \omega_o) := \int_{H^2} f_r(x, \omega_i \rightarrow \omega_o) \cos \theta_i d\omega_i, \quad (16.19)$$

$$\frac{f_r(x, \omega_i \rightarrow \omega_o) \cos \theta_i}{\rho(x, \omega_o)} \quad (16.20)$$

becomes a distribution on H^2 . From a direction ω_i which has been determined according to this distribution, one next obtains the estimator for the integral value in (16.6)

$$\rho(x, \omega_o) \cdot L_i(x, \omega_i). \quad (16.21)$$

The introduction of a distribution as in (16.20) is important since one can only distinguish effectively between diffuse and specular reflection via a preference of the directions “in the proximity” of the perfectly reflected ray.

Since in path-tracing one uses only a single direction as a sample for the determination of the integral value, a significant amount of noise develops. Since the linear (path) recursion shall remain untouched for efficiency reasons, one now shoots several rays instead of a single one through a pixel and averages the result.

According to (16.15), as far as the classification is concerned, path-tracing theoretically can depict all desired paths

$$LD(D | S)^* E. \quad (16.22)$$

The first D must appear for the same reasons that it was required in ray-tracing. Apart from that and due to the random choice of direction, the difference between specular and diffuse reflection does not exist for path-tracing (for a choice as suggested in (16.20), the directions of the perfectly reflected or perfectly refracted ray only become more likely). Two essential disadvantages have yet to be stated. First, path-tracing is still extremely expensive despite the linear recursion now present. This is true since typical scenes require between 25 and 100 rays to be fired per pixel if one wants to obtain a satisfactory result. Second, there remain problems with the depiction of the caustics. According to (16.16), these were characterized as $LS^+ X$ or $LS^+ D E$. Let us consider the example of a mirror that illuminates a surface indirectly. The direct light from a light source to the surface is considered separately via the shadow rays, but not the light that arrives via the mirror (which in a certain sense acts as a secondary light source). Here, randomness in the framework of the Monte-Carlo process must ensure that it will be considered, which, however, only occurs up to a certain probability.

16.3.3 Further Ray-Tracing Derivates

Light ray-tracing or *backward ray-tracing* was developed especially for the simulation of indirect illumination. As a first path-tracing step, rays are shot into the scene from the light sources. If one such ray intersects a surface that is at least partially diffuse, then its radiance is reduced after the reflection and the difference in the amount of energy is credited to the surface and stored in an *illumination map*. The concept is quite similar to the texture maps known from computer graphics. In the second step, an ordinary ray-tracing is performed where the illumination maps of the surface enter into the illumination calculation.

Light ray-tracing is the first of the methods discussed that can cope with caustics. The cost of the effort is extremely high, however, since the resolution of the illumination maps must be high and consequently the number of rays in the first sweep must be very large. Furthermore, the need to parameterize the surfaces via the illumination maps limits the form taken by the surfaces. Finally, the location

of energy is known via the illumination map, but not the direction from which the light arrives. This can be a disadvantage for the second step (the computation of the outgoing radiance).

Monte Carlo ray-tracing with photon maps or in short *photon tracing* is the most recently introduced of the methods discussed so far and is superior with respect to the others because of the following:

- **Quality:** The method builds upon path-tracing, a good initial solution of the illumination problem. For the depiction of caustics, a light ray-tracing step is added. The information from this step is stored in *photon maps*, similar to the *illumination maps*.
- **Flexibility:** Photon-tracing builds only on ray-tracing techniques. Neither a restriction to polygonal objects (as for radiosity) nor a parametrization of the surfaces (as for the illumination maps) are required. Even fractal objects can be employed.
- **Velocity:** The second step employs an optimized path-tracer, whose runtime can be reduced through the use of the information from the photon map.
- **Parallelizability:** As it is true for all ray-tracing methods, photon-tracing can be simply and efficiently parallelized.

Similar for light ray-tracing, the first step consists of a path-tracing originating from the light sources. *Photons*, i.e., energy-afflicted particles, are shot from the light sources into the scene. If a photon intersects with a surface, both its energy as well as its incoming direction are entered into the photon map. Then the photon is reflected randomly, or it ends (is absorbed). After completion of this phase, the photon map provides an approximation of the light conditions in the scene: The higher the photon density, the more light. One notes the significant difference between the illumination maps and the photon map: The former are local, associated with every surface and require a parametrization which maps a two-dimensional array to this area. The latter is global and exists in three-dimensional world coordinates. Furthermore, both the energy as well as the incoming direction are stored. Thus the photon map is an enormous scattered data set in 3D.

Employed in the second step is an optimized path-tracer. After only a few recursion levels the path is aborted, and the illumination conditions are approximated by estimating the density distribution of the photons in the neighborhood of the respective point. Hence, the photon map can also be employed for the simulation of caustics.

16.4 The Radiosity Method

16.4.1 Basic Principle

Radiosity is both the name of the previously introduced radiometric quantity as well as of a method for global illumination which chooses a fundamentally different

approach than the ray-oriented one. Its starting point is the realization that, on average, about 30 % of the light in a scene do not directly originate from a light source but are the result from one or several reflections from surfaces. At times, peak values of up to 80 % are reached.

Radiosity disclaims reflections and exclusively assumes diffuse surfaces. The light sources are now considered as objects of the scene and therefore as surfaces. Thus, all BRDF do not depend on the direction of the incoming or the outgoing light and from the rendering equation given in (16.11) one can pull the BRDF outside the integral:

$$\begin{aligned} L_o(x, \omega_o) &= L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_i \rightarrow \omega_o) L_o(y, \omega_i) G(x, y) dA_y \\ &= L_e(x, \omega_o) + \int_{\Omega} f_r(x) L_o(y, \omega_i) G(x, y) dA_y \\ &= L_e(x, \omega_o) + f_r(x) \cdot \int_{\Omega} L_o(y, \omega_i) G(x, y) dA_y. \end{aligned} \quad (16.23)$$

Since the outgoing radiance of a diffuse surface does not display any dependence on direction, one can also formulate (16.23) by means of $B(x)$ and obtains as a model the so-called *radiosity equation*, a Fredholm integral equation of the second kind:

$$B(x) = B_e(x) + \frac{\rho(x)}{\pi} \cdot \int_{\Omega} B(y) G(x, y) dA_y. \quad (16.24)$$

Here, for the *reflectance* $\rho(x)$ it holds that,

$$\rho(x) := \int_{H^2} f_r(x) \cos \theta_i d\omega_i = f_r(x) \cdot \int_{H^2} \cos \theta_i d\omega_i = \pi \cdot f_r(x) \quad (16.25)$$

(see also (16.19)).

Since radiosity is independent of perspective, the global illumination is computed once for the entire scene rather than only for a certain observer position. For the solution, the radiosity equation (16.25) is discretized in the finite-element sense. One covers all surfaces of the scene with a net of planar patches f_i with area A_i , $1 \leq i \leq n$, where it is no longer distinguished between light sources and actual objects (note the new meaning of A_i). For the description of the light transport from f_i to f_j , *form factors* F_{ij} are introduced.

Here, F_{ij} denotes the portion of radiant flux $A_i \cdot B_i$ (the flux leaving f_i) arriving at f_j . It obviously holds that $0 \leq F_{ij} \leq 1$. In fact, the form factors essentially correspond to the function G in (16.10) or (16.24). Thus (16.24) becomes,

$$B_i = B_{e,i} + \rho_i \cdot \sum_{j=1}^n \frac{B_j \cdot A_j \cdot F_{ji}}{A_i} \quad \forall 1 \leq i \leq n. \quad (16.26)$$

The factor π^{-1} has wandered into the form factors. Due to the relationship

$$A_i \cdot F_{ij} = A_j \cdot F_{ji}, \quad (16.27)$$

(16.26) can be simplified, and one obtains the *discrete radiosity equation*:

$$B_i = B_{e,i} + \rho_i \cdot \sum_{j=1}^n B_j \cdot F_{ij} \quad \forall 1 \leq i \leq n. \quad (16.28)$$

Equation (16.28) defines the two essential, major tasks of radiosity methods: First, the form factors have to be computed and second, the resulting system of linear equations must be solved numerically. Although this will be discussed further below, at first we are only interested in the representable light paths.

Radiosity only considers diffuse surfaces. Thus, only light paths of the type

$$LD^* E \quad (16.29)$$

can be simulated. Since light sources and surfaces are treated in the same way, the direct path LE is possible. In contrast to ray-tracing, the result does not consist of a raster graphics image for a certain view of the scene, but of the radiosity values of all (discrete) surfaces f_i . After a time-intensive radiosity computation, one can therefore use a (hardware accelerated) z -buffer method for the generation of certain views which, for example, permits realtime flights through the scene.

But ray-tracing can also be used for the generation of a view. Such a composition of ray-tracing and radiosity enables light paths of the type $LD^+ S^* E$. This is better than any individual solution, but caustics are still not representable. Because of this, additional generalizations have been developed (*extended form factors*) that integrate caustics as well. Regardless, the remaining problem is that radiosity always works with plane patches which implies that once again caustics cannot be depicted.

16.4.2 Computation of the Form Factors

We now come back to the discrete radiosity equation (16.28). Before its solution, the form factors F_{ij} have to be determined. Since these do not depend on the wavelength of the light but exclusively on the geometry of the scene, they can simply be computed once at the beginning and then stored. For changes in material properties ($B_{e,i}, \rho_i$) as well as changes in the illumination, they can be carried over.

The form factor F_{ij} denotes the portion of the light energy originating from f_i that arrives at f_j . One can show that for F_{ij} there holds the relationship

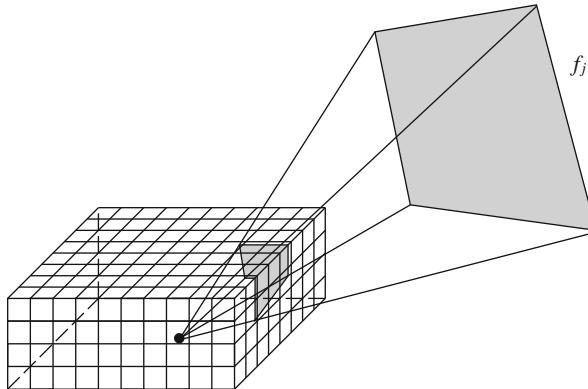


Fig. 16.6 Form factor approximation after Cohen and Greenberg

$$\begin{aligned}
 F_{ij} &= \frac{1}{A_i} \cdot \int_{f_i} \int_{f_j} \frac{\cos \theta_i(x_i, x_j) \cdot \cos \theta_j(x_i, x_j)}{\pi \cdot r^2(x_i, x_j)} V(x_i, x_j) dA_j dA_i \\
 &= \frac{1}{A_i} \cdot \frac{1}{\pi} \cdot \int_{f_i} \int_{f_j} G(x_i, x_j) dA_j dA_i
 \end{aligned} \tag{16.30}$$

where θ_i and θ_j denote the angles between the connecting line segment from x_i to x_j and the surface normal n_i from f_i in x_i and n_j from f_j in x_j , respectively, while r denotes the distance between x_i and x_j (see Fig. 16.4). The functions V and G are defined as in (16.7) or (16.10), respectively.

Even for simple geometric conditions, the computation of the integrals in (16.30) is very complex. Thus, efficient methods for the approximate computation of the form factors are of great importance. One possibility for the simplification of (16.30) is, for example, to assume the angles θ_i and θ_j as well as the distance r to be constant over the surfaces f_i and f_j except possibly where the quality loss becomes too large.

A popular method giving an approximate computation of the form factors F_{ij} goes back to Cohen and Greenberg [12]. Using a central projection of f_j onto the unit hemisphere around $x_i \in f_i$ (whose basis circle is coplanar to f_i) and a subsequent orthogonal parallel projection onto the basis circle of the hemisphere (see Fig. 16.5), one can show that the exact computation of the inner integral over f_j in (16.30) corresponds to the computation of the area of the surface $\overline{f_j}$.

Cohen and Greenberg have replaced the hemisphere with a semi-cube whose surface is divided into square surface elements (see Fig. 16.6). Typically, the resolution amounts to between fifty and several hundreds of elements in each direction. For each of these surface elements, a form factor can be computed beforehand and stored in a table. Subsequently, the patches f_j , for $j \neq i$, are projected to the cube faces and the previously computed form factors of the respective surface elements are added whereby the visibility question is solved in a z -buffer manner. With respect to small patches f_i as well as large distances r

relative to A_i , the outer integral in (16.30) is often neglected since the variation over the different $x_i \in f_i$ has only a small influence.

16.4.3 Solution of the Radiosity Equation

We now address the solution of the radiosity equation. Altogether, the n balance equations represent a linear system of equations

$$M \cdot B = E \quad (16.31)$$

in which,

$$\begin{aligned} B &:= (B_1, \dots, B_n)^T \in \mathbb{R}^n, \\ E &:= (E_1, \dots, E_n)^T := (B_{e,1}, \dots, B_{e,n})^T \in \mathbb{R}^{n \times n}, \\ M &:= (m_{ij})_{1 \leq i,j \leq n} \in \mathbb{R}^{n \times n}, \\ m_{ij} &:= \begin{cases} 1 - \rho_i \cdot F_{ii} & \text{for } i = j, \\ -\rho_i \cdot F_{ij} & \text{else,} \end{cases} \\ M &= \begin{pmatrix} 1 - \rho_1 F_{11} & -\rho_1 F_{12} & -\rho_1 F_{13} & \cdots \\ -\rho_2 F_{21} & 1 - \rho_2 F_{22} & -\rho_2 F_{23} & \cdots \\ -\rho_3 F_{31} & -\rho_3 F_{32} & 1 - \rho_3 F_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \end{aligned}$$

In view of $0 < \rho_i < 1$ and $\sum_{j=1}^n F_{ij} \leq 1$ the square matrix M is strictly diagonally dominant for all $i = 1, \dots, n$. The matrix M is not symmetric, however, which disqualifies it for use in several iterative solution methods. For us to be able to work with a symmetric matrix we can multiply the radiosity equation for f_i with the factor A_i / ρ_i and consider B_i as unknowns and also now $E_i \cdot A_i / \rho_i$ as components of the right hand side. Equation (16.31) then becomes,

$$\overline{M} \cdot B = \overline{E}, \quad (16.32)$$

where,

$$\overline{E} := (E_1 \cdot A_1 / \rho_1, \dots, E_n \cdot A_n / \rho_n)^T \text{ and}$$

$$\overline{M} := \begin{pmatrix} \rho_1^{-1} A_1 - A_1 F_{11} & -A_1 F_{12} & -A_1 F_{13} & \cdots \\ -A_2 F_{21} & \rho_2^{-1} A_2 - A_2 F_{22} & -A_2 F_{23} & \cdots \\ -A_3 F_{31} & -A_3 F_{32} & \rho_3^{-1} A_3 - A_3 F_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

The matrix \overline{M} is once more strictly diagonally dominant and due to the symmetry condition (16.27) it is now also symmetric and consequently also positive definite. In addition, both M as well as \overline{M} are typically sparse since for average scenes with a great number of patches the light cast from one patch f_i reaches only a few other patches f_j . Finally, one notes that the $F_{ii}, 1 \leq i \leq n$ are, in general, close to zero but they do not always vanish completely. For example, for a concave surface f_i , light travels on a direct path from f_i to f_i .

For the solution of the system given in (16.31), iterative methods such as the Gauß–Seidel- or the Jacobi-iteration are available. Since the matrix M is strictly diagonally dominant, both iteration methods converge. As a starting value one chooses $R^{(0)} := E$. Here, the Jacobi-iteration allows for an interesting interpretation: The emitted light alone serves as an initial value and after the first iteration step the simply reflected light is considered as well while the second iterative step brings the two-fold reflected light into the game, etc. In the following, the scheme of the Jacobi-iteration for s iterative steps is represented:

$$\begin{aligned} \text{it} &= 1, 2, \dots, s: \\ i &= 1, 2, \dots, n: \\ S_i &:= \left(E_i + \rho_i \cdot \sum_{j \neq i} B_j F_{ij} \right) / (1 - \rho_i F_{ii}); \\ i &= 1, 2, \dots, n: \\ B_i &:= S_i; \end{aligned}$$

Here, the image quality improves with increasing s , the considered interreflection levels, but this also increases the computational effort which is of order $O(n^2 \cdot s)$.

One notes that the system of equations (16.31) or (16.32) has to be set up and solved separately for every wave length λ or partial region (i.e., every region $[\lambda_i, \lambda_{i+1}], i = 0, \dots, n-1$) of wave lengths, since—in contrast to the form factors F_{ij} —as material parameters, both ρ_i as well as E_i depend on the wave length λ of the light. At a minimum, the radiosity values have to be computed for the three (monitor) basic colors red, blue and green.

16.4.4 Remarks and Improvements

Thus far, the radiosity values have been determined for every patch f_i . If one now wants to couple the results of the radiosity method by means of an interpolating shading method such as Gouraud-shading, for example, then the radiosity values are required at the vertices of the polygonal surfaces. The common procedure for the determination of such quantities is to obtain radiosity values at the vertices by means of some suitable averaging technique of the radiosity values at neighboring patches and then to interpolate these quantities according to the shading method.

The classical radiosity method that is introduced here puts very high demands on the computational time and storage requirements. For a scene with 10,000 patches, according to (16.27), fifty million form factors already have to be computed, and

even for a sparsely populated matrix there is a significant storage requirement. The quadratic computational time and storage complexity as well as the fact that one must always wait for a complete Gauß-Seidel- or Jacobi-step before the next output can be computed, initially acts as a stumbling block for an efficient implementation of the radiosity method. Two approaches have been developed for the solution of this problem.

The *stepwise refinement* approach reverses the previous technique: According to the Jacobi iteration shown in the previous section, for every iteration step the contributions in a neighborhood of a patch f_i were picked up (*gathering*) and thus used for the update of R_i . In contrast, the patches f_i will now be successively treated so that their outgoing light energy is sent into the surroundings (*shooting*). Instead of summing up the contributions $\frac{\rho_i \cdot B_j \cdot F_{ij}}{1 - \rho_i F_{ii}}$ of all $f_j, j \neq i$, to B_i at step i , the contributions of f_i at all $B_j, j \neq i$, are now considered instead: $\frac{\rho_j \cdot \Delta B_i \cdot F_{ji}}{1 - \rho_j F_{jj}} = \frac{\rho_j \cdot \Delta B_i \cdot F_{ij} \cdot A_i / A_j}{1 - \rho_j F_{jj}}$. Here, ΔB_i is the radiosity of f_i that has just accumulated since the last sending of B_i . Initially, the ΔB_i and B_i are preset for each respective E_i . It then always sends out the light amount for the respective patch f_i for which the value $\Delta B_i \cdot A_i$ is maximal. In this way the stepwise, incremental computation of the radiosity values becomes possible: At first, the patches having large radiosity are treated (initial light sources), which already gives a good approximation for the illuminated image. The iteration is aborted if $\Delta B_i \cdot A_i$ falls below a given bound ε for all i . We therefore obtain the following algorithm:

Stepwise refinement:

```

 $\forall$  patches  $f_j$ :
begin
     $B_j := E_j$ ;
     $\Delta B_j := E_j$ ;
end
while  $\neg(\Delta B_j \cdot A_j < \varepsilon \ \forall j)$  :
begin
    determine  $i : \Delta B_i \cdot A_i \geq \Delta B_j \cdot A_j \ \forall j$ ;
     $\forall$  patches  $f_j \neq f_i$ :
        begin
            contribution :=  $\rho_j \cdot \Delta B_i \cdot F_{ij} \cdot A_i / (A_j \cdot (1 - \rho_j F_{jj}))$ ;
             $\Delta B_j := \Delta B_j +$  contribution;
             $B_j := B_j +$  contribution;
        end
         $\Delta B_i := 0$ ;
    end;

```

The *adaptive subdivision (recursive substructuring)* of the patches allows us to begin with a few patches and then to further subdivide only at places where the computed radiosity values on neighboring patches differ strongly. At those locations requiring a finer resolution, the form factors also have to be recomputed. This

adaptive procedure allows for the concentration of the effort to be over regions where a high resolution (i.e., many patches) is actually necessary for a respective image quality.

Moreover, in recent times modern numerical methods such as approaches of higher order or wavelets have found use in radiosity applications.

In summary, we conclude that the radiosity method provides a method that admittedly is very intensive with respect to computational time, but is suitable for photorealistic images for the modeling of diffuse light. The generated images are independent of the vantage point with respect to illumination—diffuse light is undirected. The method is thereby also very well suited for interactive and animated applications involving a changing perspective (walking virtually in buildings, driving through cities or flying etc.).

However, it is certainly a disadvantage that direction-dependent illumination properties such as shining light cannot be realized at first. There exist, however, extensions of the radiosity method that in addition allow for consideration of specular reflection. Furthermore, the combined use of ray-tracing and radiosity has also been analysed. These approaches are based on two separate sweeps to depict the scene, a first perspective-independent (radiosity) and a second perspective-dependent sweep (ray-tracing). Without wanting to go into further detail here, let us point out that the simple addition of the intensity values in the pixels from the two sweeps is not sufficient for the realization of the desired combination of ray-tracing and radiosity.

Concluding Remarks

Is there appetite for more? Then we have reached at least one of our essential goals—namely, the goal to provide a broad overview that encompasses the methodology of modeling and simulation as well as to simultaneously stir up the desire for a deeper examination of these subject areas. It was our intention here, in contrast to other books, to risk exploring the wide arc while refraining from a concentrated study from a single perspective—be it a modeling-related (modeling with partial differential equations, graph theory or fuzzy logic), a simulation-related (numerical treatment of differential equations, scientific computing or stochastic processes), an application-driven (e.g. numerical simulation of fluid flow) or a software-related focus (for example simulating with Simulink). It is recognizably obvious that important details have time and again been neglected, and inserting sentences of the type “This would delve into it too much at this time” was not easy and for the authors at times even a bit painful. But it is also important to recognize the underlying systematics, perceive analogies and thus to reach transfer conclusions. A good modeler does not only know the details of drawer number 93 well, but he also masters the art of describing, abstracting and simplifying; a good simulations person cannot only improve a certain algorithm by yet another small iota, but she must also exhibit a certain intuition regarding what should be approached and how, as well as its expected resulting complexity.

Hence: Use this overview as the first step. For all topics addressed in Chaps. 3–16, there is a lot more to discover—and to model and to simulate. It’s a journey, so have fun on it!

References

1. V.G. Adlakha and V.G. Kulkarni. A classified bibliography of research on stochastic PERT networks: 1966–1987. *INFOR*, 27(3):272–296, 1989.
2. Alfred V. Aho, Jeffrey D. Ullman, and John E. Hopcroft. *Data Structures and Algorithms*. Addison-Wesley, 1983.
3. M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford University Press, New York, 1989.
4. A. Appel. Some Techniques for Shading Machine Renderings of Solids. In *Proceedings of the Spring Joint Computer Conference*, pages 37–45, 1968.
5. J. R. Arvo. Backward Ray Tracing. In *Developments in Ray Tracing*, A. H. Barr, Hrsg., *Course Notes 12 for ACM SIGGRAPH '86*, 1986.
6. Jerry Banks, John Carson, Barry L. Nelson, and David Nicol. *Discrete-Event System Simulation, Fourth Edition*. Prentice Hall, 2004.
7. Edward Beltrami. *Mathematics for Dynamic Modeling*. Academic Press, 1987.
8. L. von Bortkewitsch. *Gesetz der kleinen Zahlen*. Teubner, 1889.
9. Bundesministerium für Verkehr, Bau und Stadtentwicklung. *Verkehr in Zahlen 2010/2011*. Deutscher Verkehrs-Verlag, 2011.
10. Hans-Joachim Bungartz, Michael Griebel, and Christoph Zenger. *Introduction to Computer Graphics*. Charles River Media, 2004.
11. Marek Capiński and Tomasz Zastawniak. *Mathematics for Finance: An Introduction to Financial Engineering*. Springer, 2003.
12. M. F. Cohen and D. P. Greenberg. The Hemi-Cube: A Radiosity Solution for Complex Environments. In *Proceedings of SIGGRAPH '85, Computer Graphics*, 19(3), pages 31–40, ACM SIGGRAPH, New York, 1985.
13. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
14. Germund Dahlquist and Åke Björck. *Numerical Methods in Scientific Computing*. SIAM, Philadelphia, 2008.
15. Richard C. Dorf and Robert H. Bishop. *Modern Control Systems*. Pearson Studium, 12. edition, 2011.
16. Dimiter Driankov, Hans Hellendoorn, and Michael Reinfrank. *An Introduction to Fuzzy Control*. Springer, 1993.
17. Susanna Epp. *Discrete Mathematics with Applications*. Cengage Learning, 4. edition, 2010.
18. European commission. *Commission Staff working paper “Impact Assessment” accompanying the White Paper: Roadmap to a Single European Transport Area Towards a competitive and resource efficient transport system*, 2011.

19. Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics*. Springer, 3. edition, 2002.
20. George S. Fishman. *Discrete-Event Simulation – Modeling, Programming, and Analysis*. Springer, 2001.
21. N.D. Fowkes and J.J. Mahony. *An Introduction to Mathematical Modelling*. UMI books on demand. Wiley, 1994.
22. Daan Frenkel and Berend Smit. *Understanding Molecular Simulation*. Academic Press, 2. edition, 2002.
23. Drew Fudenberg and Jean Tirole. *Game Theory*. The MIT Press, 1991.
24. Delbert Ray Fulkerson. Expected critical path lengths in PERT networks. *Operations Research*, 10:808–817, 1962.
25. Walter Gander and Jirí Hřebíček. *Solving Problems in Scientific Computing Using Maple and MATLAB*. Springer, 4. edition, 2004.
26. John Geanakoplos. Three brief proofs of Arrow’s impossibility theorem. Technical Report 1116, Cowles Foundation for Research in Economics at Yale University, 2005.
27. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modeling the Interaction of Light Between Diffuse Surfaces. In *Proceedings of SIGGRAPH ’84, Computer Graphics*, 18(3), pages 213–222, ACM SIGGRAPH, New York, 1984.
28. Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete mathematics: A foundation for computer science*. Addison-Wesley, 2. edition, 1994.
29. C.G. Gray and K.E. Gubbins. *Theory of Molecular Fluids. Volume 1: Fundamentals*, volume 9 of *International Series on Monographs on Chemistry*. Clarendon Press, Oxford University Press, New York, 1984.
30. Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM Monographs on Mathematical Modeling and Computation. SIAM, 1998.
31. Michael Griebel, Stephan Knapek, and Gerhard Zumbusch. *Numerical Simulation in Molecular Dynamics*. Springer, 2007.
32. D. Gross, W. Hauger, J. Schröder, W.A. Wall, and S. Govindjee. *Engineering Mechanics* 3. Springer, 3. edition, 2011.
33. J.M. Haile. *Molecular Dynamics Simulation*. John Wiley & Sons, 1997.
34. Dirk Helbing. *Verkehrsdynamik: Neue physikalische Konzepte*. Springer, 1997.
35. H. W. Jensen. Global Illumination Using Photon Maps. *Rendering Techniques ’96*, pages 21–30, 1996.
36. J. Kajiya. The Rendering Equation. In *Proceedings of SIGGRAPH ’86, Computer Graphics*, 20(4), pages 143–150, ACM SIGGRAPH, New York, 1986.
37. George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic: theory and applications*. Addison Wesley, 1995.
38. Reinhart D. Kühne and Malte B. Rödiger. Macroscopic simulation model for freeway traffic with jams and stop-start waves. In *WSC ’91: Proceedings of the 23rd conference on Winter simulation*, pages 762–770, Washington, DC, USA, 1991. IEEE Computer Society.
39. Los Alamos National Laboratory. *TRANSIMS – Transportation Analysis Simulation System*. <http://transims.tsasa.lanl.gov>.
40. William Mendenhall and Terry Sincich. *Statistics for engineering and the sciences*. Pearson, 5. edition, 2006.
41. Rolf H. Möhring. Scheduling under uncertainty: Optimizing against a randomizing adversary. In Klaus Jansen, editor, *Approximation Algorithms for Combinatorial Optimization*, volume 1913 of *Lecture Notes in Computer Science*, pages 15–26. Springer, 2000.
42. Rolf H. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. In Helmut Alt, editor, *Computational Discrete Mathematics: Advanced Lectures*, volume 2122 of *Lecture Notes in Computer Science*, pages 79–97. Springer, 2001.
43. K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I*, 2:2221–2229, December 1992.
44. Katsuhiko Ogata. *Modern control engineering*. Prentice Hall, 5. edition, 2010.

45. Martin J. Osborne. *An introduction to game theory*. Oxford University Press, 2004.
46. Edward Ott. *Chaos in Dynamical Systems*. Cambridge University Press, 1993.
47. Michael L. Pinedo. *Planning and Scheduling in Manufacturing and Services*. Springer, 2005.
48. D.C. Rapaport. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2. edition, 2004.
49. Statistisches Bundesamt, Wiesbaden. *Verkehr in Deutschland 2006*, September 2006. Im Blickpunkt.
50. Statistisches Bundesamt, Wiesbaden. *Statistisches Jahrbuch 2012*, 2012.
51. James Stewart. *Calculus*. Cengage Learning, 7. edition, 2011.
52. Gilbert Strang. *Introduction to Linear Algebra*. Wellesley Cambridge Press, 4. edition, 2009.
53. Alan Taylor and Allison M. Pacelli. *Mathematics and Politics: Strategy, Voting, Power, and Proof*. Springer, 2. edition, 2008.
54. Henk C. Tijms. *A First Course in Stochastic Models*. Wiley, 2 edition, 2003.
55. M. Treiber and A. Kesting. *Traffic Flow Dynamics*. Springer Berlin Heidelberg, 2013.
56. Stefan Turek and Michael Schäfer. Benchmark computations of laminar flow around a cylinder. In Ernst Heinrich Hirschel, editor, *Flow Simulation with High-Performance Computers II*, volume 52 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*. Vieweg, 1996.
57. Aslak Tveito and Ragnar Winther. *Introduction to Partial Differential Equations - A Computational Approach*. Springer, 2005.
58. T. Whitted. An Improved Illumination Model for Shaded Display. *Communications of the ACM*, 23(6):343–349, 1980.
59. Ronald R. Yager and Dimitar P. Filev. *Essentials of Fuzzy Modeling and Control*. John Wiley & Sons, New York/Chichester/Brisbane, 1994.
60. Sidney Yakowitz and Ferenc Szidarovszky. *An introduction to numerical computations*. Prentice Hall, 2. edition, 1989.
61. H.-J. Zimmermann. *Fuzzy Set Theory - and Its Applications*. Springer, 4. edition, 2001.

Index

A

A*-algorithm, 196
Accumulation, 283
Aggregation, 282
Alternative, irrelevant, 107–110
Aperiodic state, 227
Approximate reasoning, 279
Arrival process, 210, 213
Arrival rate, 207, 210
Arrow’s theorem, 109–110
Asymptotic analysis, 220–222
Attractor, 299–313

B

Backward recurrence time, 211
Battle of the sexes, 87, 89, 94–96
Bifurcation, 295–314
Birth-death process. *See* Stochastic process
Birth process. *See* Stochastic process
Birth rate, 226, 242, 251, 253
Black–Scholes model, 131, 140
Boundary condition, 24, 69, 176
 Dirichlet, 80
 discretization, 346
 fluid dynamics, 359–360, 364–365, 369
 heat transfer, 346
 molecular dynamics, 329
 Neumann, 80
 traffic, 161, 162
Boundary value. *See* Boundary condition
Boundary value problem, 24, 79, 340
 free, 371
Braess’s paradox, 146
BRDF, 384, 385, 393

C

Cantor set, 302
Car-following model, 172
Caustic, 382, 387, 391, 394
Cell space, 172
Cellular automaton, 172–174, 178
Chain. *See* Stochastic process
Chaos, 291–316
Chaotic behavior, 327
Chapman–Kolmogorov equation, 227
Characteristics, 167, 168
Choice function, 102–110
Coefficient of variation, 35, 212, 238
Collisions, absence of, 175
Competition, 248
Completion time, 113
Condition, 50, 59, 74
Condorcet method, 105
Confidence interval, 44–45
Conjunctive edge, 125
Conservation law
 energy, 370
 fluid dynamics, 360
 heat transfer, 338
 molecular dynamics, 328
 vehicles, 175
Continuity, 22
Continuity equation. *See* Differential equation
Control, 255
Control engineering, 255
Controller, 258–261
 design of, 257
Control loop, 257–258
Critical path method (CPM), 112, 117, 126
Cut-off radius, 323

- Cycle**
 chaos theory, 311–312
 graph theory, 115
- D**
DAG, 115
 Death rate, 242, 251
 Decision making method, 102, 103
 Defuzzification, 283–284
 Delaunay triangulation, 377
 De Moivre–Laplace theorem, 133
Density
 critical, 178
 stochastics (*see* Distribution)
 traffic, 150, 171
Derivative, partial, 26
Descent, steepest, 65
Dictator, 103–104, 109–110
Difference quotient. *See* Discretization method
Differential equation, 26
 autonomous, 25, 71
 discretization (*see* Discretization method)
 linear, 24–25, 69, 79, 242
 linearization, 244, 246
 linear system, 70–73, 253
 ordinary, 24, 68–78
 equation of motion, 324
 logistic, 244
 one-species model, 242–245
 pendulum, 261–270, 309
 two-species model, 245–250
 partial, 78–82
 continuity equation, 159, 160, 161, 359,
 366
 elliptic, 79, 340
 Euler equation, 361, 370
 heat equation, 68, 338–340
 hyperbolic, 79
 Laplace equation, 374
 momentum equation, 359, 366, 367
 Navier–Stokes equation, 357
 parabolic, 79, 339
 Poisson equation, 66, 366
 traffic equation, 160
 transport equation, 370
 system, 246
Dijkstra algorithm, 194
Directional derivative, 26
Direction field, 69–72, 245, 249
Discrete event simulation, 236
Discretization method
 consistency, 76
 convergence, 76
difference quotient, 22, 74, 80, 161, 164,
 342
Euler, 74, 161, 164, 309, 325, 363
MacCormack, 164
 3-point stencil, 80, 342
 5-point stencil, 80, 344
 7-point stencil, 80
 predictor-corrector, 78, 164
 stability, 163, 164, 368
 Velocity–Störmer–Verlet, 326
Disjunctive edge, 125–128
Distribution
 Bernoulli, 35, 132
 binomial, 35, 133, 226
 exponential, 40, 209
 geometric, 36, 231
 joint, 119
 normal, 39, 132
 addition theorem, 134
 linear transformation, 132
 quantile, 141
 Poisson, 36, 209
 uniform, 38, 119, 208
Divergence, 26
Domain decomposition, 334, 372
Drift, 134, 139
- E**
Eigenvalue, 21
 for ODE systems, 68, 246–250
Eigenvector, 21
Elementary queuing system. *See* Queuing, system
Equation of motion. *See* Differential equation
Equilibrium flow, 152
Equilibrium point, 71, 243–250
Estimator, 43
Euler equation. *See* Differential equation
Euler method. *See* Discretization method
Event, 30
Event rate, 210
Event risk, 209
Event simulation. *See* Discrete event simulation
Execution time, 112
Expected value, 33, 38
 conditional, 211
 of the execution time, 119
 as payoff, 96
Extremum, 22
- F**
FCFS, 213
Feedback, 257

- Final vertex, 114
Finite differences, 74–78, 80, 161, 309,
 341–344, 361
Finite elements, 81
Fixed point
 chaos theory, 293–298
 Markov chains, 229
Floating point number, 47–49
Flow
 compressible, 357, 370
 incompressible, 357
 inviscid, 370
 laminar, 356
 traffic (*see* Traffic, flow)
 turbulent, 356
Flow-shop, 129
Fluid, 356
Fluid dynamics, 356
Force, 318–321
Form factor, 393, 394
Forward recurrence time, 211
Fourier series, 24
Four-phase model, 190
Fractal dimension, 302
Free flow phase, 156, 179, 185
Fundamental diagram, 154–158,
 182–186
Fuzzification, 281
Fuzzy
 control, 284–288
 logic, 277–279
 sets, 271–276
- G**
Game of Life, 173
Gantt diagram, 117
Gauß-Seidel method. *See* System
 of equations, linear
Gauß' theorem, 27, 339
Gaussian elimination, 60
Gradient, 26
Graph, 19
 acyclic, 115
 connected, 187
 directed, 115, 186
 queuing network, 218
 for time schedules, 113
 traffic graph, 186
Graphical iteration, 296–298
Grid, 80
 adaptive, 376, 379
 Cartesian, 361
 staggered, 362
- structured, 373
unstructured, 376
Growth rate, 242, 246, 247
- H**
Hazard rate, 209
Heat equation. *See* Differential equation
Hénon mapping, 304
Heuristic, 196
Hitchhiker's paradox, 210
- I**
Iid. *See* Random variable
Illumination map, 391
Implication, 278, 282
Implicit methods, 78
Independence. *See* Random variable
Inflection point, 22
Inhomogeneous traffic participants, 199
Initial condition. *See* Initial value
Initial value, 24, 69, 160, 176, 369
 problem, 24, 69
Initial vertex, 114
Input signal, 257
Integral, 23, 27
Interarrival times, 207
Interpolation
 polynomial, 51
 trigonometric, 53
Intersection, 188–193
Irradiance, 382, 384
Iterated function, 293
Iterative method, 58
- J**
Jackson network, 234
Jacobi iteration. *See* System of equations,
 linear
Jacobi matrix, 26, 246
Jensen's inequality, 121
Job, 112, 124, 205
Job entry, 205
Job-shop model, 112, 124
- K**
Kendall notation, 213, 214
Kinetic energy, 329
- L**
Lack of memory, 210
Lagrange equations, 266

- Laplace
 equation (*see* Differential equation)
 operator, 27
- Law of large numbers, 41
- Lead time, 115
- Light transport operator, 386
- Limit distribution, stationary, 227
- Limit process, stationary, 223
- Limit theorem, central, 42
- Line integral, 27
- Linguistic term, 276
- Linguistic value, 276
- Linguistic variable, 276
- Linked-cells method, 331
- Little's law, 152, 216–217
- Logistic differential equation. *See* Differential equation
- Logistic mapping, 293–300
- Loss system, 229
- M**
- MacCormack method. *See* Discretization method
- Malthus, 242, 253
- Mamdani implication, 279
- Manipulation, 108
- Mapping
 discrete, 293
 linear, 20
- Markov chain. *See* Stochastic process
- Markov process. *See* Stochastic process
- Maximum throughput, 215
- Memoryless process. *See* Lack of memory
- Mixing rules, 322
- M/M/m, 214, 230–233
- Momentum equation. *See* Differential equation
- Monte Carlo method, 42
- Monte Carlo ray-tracing, 382–392
- Moore neighborhood, 173
- Multi agent simulation, 193
- Multibody system, 261
- Multigrid method. *See* System of equations, linear
- Multiphysics problem, 371
- Multiscale problem, 356
- N**
- Nagel-Schreckenberg model, 178
- Nash equilibrium, 94
- Navier-Stokes equation. *See* Differential equation
- Neighborhood relation, 173
- Network plans, 117
- Normal derivative, 26
- Normal form, strategic, 90
- Normalization condition, 209, 223
- Null recurrent state, 227
- Number of jobs in the system, 215
- Number of visits, 219
- NVT ensemble, 328
- O**
- ODE. *See* Differential equation, ordinary
- OD matrix, 194
- One-species model, 226, 242–245, 250–253
- Optimization of time schedules, 111
- Order, arrival of, 207
- Output signal, 257
- P**
- Parallelization, 333
- Parameter space, 224
- Pareto condition, 103, 107
- Passing maneuvers, 199
- Path, 114
 critical, 115, 118
 length, 114
- Path-tracing, 382, 390
- Payoff function, 89, 90, 97
- Payoff matrix, 89–98
- PDE. *See* Differential equation, partial
- Pendulum
 chaotic behavior, 307
 control, 284–288
 model, 261–270
- Period doubling, 299
- Periodic state, 227
- Period length, 227
- PERT, 123
- Photon map, 382, 392
- Poincaré section, 311
- Poisson counting process, 210
- Poisson equation. *See* Differential equation
- Pollaczek–Khinchin formula, 238
- Polynomial splines, 53
- Positively recurrent state, 227
- Potential, 319
 Lennard-Jones, 322
 van-der-Waals, 320
- Potential energy, 328
- Precedence condition, 113
- Predator-prey model, 249
- Predictor-corrector method. *See* Discretization method

- Preorder, 100
 Prisoners' dilemma, 87, 89, 90, 93
 Probability, 30
 conditional, 31
 as hazard rate, 208
 Probability space
 continuous, 36–40
 discrete, 30–36
 Process. *See* Stochastic process
 Prolongation, 352
 Pseudo random numbers. *See* Random numbers
- Q**
 Quadrature, numerical, 54–59
 Quantile
 normal distribution, 141
 at scheduling, 120
 Queue, 206
 Queuing
 discipline, 213
 model, 204
 network, 218
 closed, 218
 open, 218
 system
 elementary, 206, 207
 stable, 229
 unstable, 229
 unit, 206
- R**
 Radiance, 382–386, 388
 Radian
 energy, 382
 flux, 382, 385
 intensity, 383
 Radiosity, 383
 equation, 393–399
 method, 381–399
 Randomization parameter, 178
 Random numbers
 normally distributed, 134
 pseudo, 236
 Random variable
 as execution time, 118
 iid, 41, 42, 132, 208, 237
 independent, 118
 as interarrival times, 207
 as service time, 213
 Random walk. *See* Stochastic process
 Rank addition, 104
 Rank mapping, 100
- Ray-tracing, 381–399
 Recirculation, 124
 Recurrence time, 227
 Recurrent state, 227
 Reflection, 383–399
 Regularity condition, 28
 Relation, 100
 properties, 100
 Relaxation method. *See* System of equations, linear
 Remaining rotation time, 208
 Remaining time, 210
 Rendering equation, 385–399
 Response mapping, 91–97
 Restriction, 351
 Reynolds number, 356, 359, 368
 Richardson method. *See* System of equations, linear
 Risk, 90, 98, 140
 Rotation, 26
 Roundabout. *See* Intersection
 Rounding, 49
 Route planning, 193
- S**
 Saddle point, 26, 95
 Saturation number of jobs, 221
 Scheduling, 111–129, 218
 Search algorithm, 194–197
 Self-similarity, 302
 Separation of variables, 24, 209
 Service
 process, 212
 rate, 212
 station, 206
 strategy, 213–214
 time, 212, 214
 unit, 206
 Shock wave, 164
 Signal velocity, 165–169, 182, 186
 Slack, 117
 Smoother, 351, 352
 Smoothness, 28
 condition, 28
 Solid angle, 386
 SOR method. *See* System of equations, linear
 Stability, 295
 of algorithms, 51
 of difference methods, 77
 of differential equations (*see* Differential equation)
 of discretization methods (*see* Discretization method)

- Standard deviation, 34, 135, 137, 212
 Start time, 113
 State graph, 225
 State probability, 223
 State quantity, 223
 State space, 224, 241, 258, 307
 State space model, 258
 - linear, 265
 - non-linear, 270
 States, set of, 173
 Stationary phase, 223, 236, 294
Stencil. *See* Discretization method
 Steradian (sr), 382
 Stochastic process, 207–208, 251
 - arrival process, 210
 - continuous, 136, 224
 - in continuous time, 136, 224
 - discrete, 224, 225
 - in discrete time, 224, 225
 Markov chain, 224
 - irreducible, 227
 - stationary, 227
 Markov process, 223
 - birth-death process, 224, 225
 - birth process, 226, 251
 - homogeneous, 224, 225
 - stationary, 225
 random walk, 225
 service process, 212
 stationary, 207
 Wiener process, 136, 140, 224
 Stop-and-go wave, 150, 182
 Strategy, 88–98
 Subjob, 124
 Surface integral, 27
 System, 256
 - mechanical, 261
 - multibody, 261
 - non-linear, dynamical, 261
 System of equations, linear, 20, 229, 397
 - direct solver, 59–61
 - iterative method, 61–66
 - Gauß-Seidel, 62, 348
 - Jacobi, 62, 347, 397
 - multigrid, 348–353
 - relaxation, 61
 - Richardson, 62
 - SOR, 62, 348
 - sparse, 344, 345

T
 Task. *See* Job
 Taylor series, 23

Test, 45, 236, 237
 Thermostat, 329
 Throughput, 215
 Time schedule, 111–129
 Time step, 365
 Topological sorting, 115
 Total execution time, 114–129
 Total time spent in the system, 215
 Traffic

 - bottleneck, 220
 - equation (*see* Differential equation)
 - flow, 150, 152, 158, 171
 - flow, unstable, 185
 - load curve, 197, 198
 - measurement, 183, 184
 - model, 151, 155, 157
 - Lighthill–Witham–Richards, 151
 - simulation
 - macroscopic, 149–170
 - microscopic, 171–201
 - stochastic, 203–208
 - state equation, 152
 - traffic jam phase, 156, 185
 - traffic jams out of nowhere, 180
 Trajectory, 69–73, 168, 169, 310–312
 Transient behavior, 294
 Transient phase, 223, 236, 294
 Transient state, 227
 Transition function, 173
Transport equation. *See* Differential equation
 Turbulence, 356, 357
 Two-species model, 245–250

U
 Unanimity

 - as a choice function, 105
 - Pareto condition, 107
 Underestimation, 122
 Unit hemisphere, 382
 Unit sphere, 382
 Utilization, 215

V
 Variance, 34, 38, 135
 Vector space, 20
 Velocity-Störmer-Verlet, 326
 Verhulst, 242
 Viscosity, 356
 Visibility function, 386
 Volatility, 136, 140
 Volume integral, 27, 339
 Von-Neumann neighborhood, 173

Vortex street, 357

Voter, 102

Waiting time, 215

Wiener process. *See* Stochastic process

Wiggles, 167

World population, 213

W

Wait and serve system, 205, 206

Y

Yield, 140