

A very wide range of physical processes lead to wave motion, where signals are propagated through a medium in space and time, normally with little or no permanent movement of the medium itself. The shape of the signals may undergo changes as they travel through matter, but usually not so much that the signals cannot be recognized at some later point in space and time. Many types of wave motion can be described by the equation $u_{tt} = \nabla \cdot (c^2 \nabla u) + f$, which we will solve in the forthcoming text by finite difference methods.

2.1 Simulation of Waves on a String

We begin our study of wave equations by simulating one-dimensional waves on a string, say on a guitar or violin. Let the string in the undeformed state coincide with the interval $[0, L]$ on the x axis, and let $u(x, t)$ be the displacement at time t in the y direction of a point initially at x . The displacement function u is governed by the mathematical model

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad x \in (0, L), t \in (0, T] \quad (2.1)$$

$$u(x, 0) = I(x), \quad x \in [0, L] \quad (2.2)$$

$$\frac{\partial}{\partial t} u(x, 0) = 0, \quad x \in [0, L] \quad (2.3)$$

$$u(0, t) = 0, \quad t \in (0, T] \quad (2.4)$$

$$u(L, t) = 0, \quad t \in (0, T]. \quad (2.5)$$

The constant c and the function $I(x)$ must be prescribed.

Equation (2.1) is known as the one-dimensional *wave equation*. Since this PDE contains a second-order derivative in time, we need *two initial conditions*. The condition (2.2) specifies the initial shape of the string, $I(x)$, and (2.3) expresses that the initial velocity of the string is zero. In addition, PDEs need *boundary conditions*, given here as (2.4) and (2.5). These two conditions specify that the string is fixed at the ends, i.e., that the displacement u is zero.

The solution $u(x, t)$ varies in space and time and describes waves that move with velocity c to the left and right.

Sometimes we will use a more compact notation for the partial derivatives to save space:

$$u_t = \frac{\partial u}{\partial t}, \quad u_{tt} = \frac{\partial^2 u}{\partial t^2}, \quad (2.6)$$

and similar expressions for derivatives with respect to other variables. Then the wave equation can be written compactly as $u_{tt} = c^2 u_{xx}$.

The PDE problem (2.1)–(2.5) will now be discretized in space and time by a finite difference method.

2.1.1 Discretizing the Domain

The temporal domain $[0, T]$ is represented by a finite number of mesh points

$$0 = t_0 < t_1 < t_2 < \cdots < t_{N_t-1} < t_{N_t} = T. \quad (2.7)$$

Similarly, the spatial domain $[0, L]$ is replaced by a set of mesh points

$$0 = x_0 < x_1 < x_2 < \cdots < x_{N_x-1} < x_{N_x} = L. \quad (2.8)$$

One may view the mesh as two-dimensional in the x, t plane, consisting of points (x_i, t_n) , with $i = 0, \dots, N_x$ and $n = 0, \dots, N_t$.

Uniform meshes For uniformly distributed mesh points we can introduce the constant mesh spacings Δt and Δx . We have that

$$x_i = i \Delta x, \quad i = 0, \dots, N_x, \quad t_n = n \Delta t, \quad n = 0, \dots, N_t. \quad (2.9)$$

We also have that $\Delta x = x_i - x_{i-1}$, $i = 1, \dots, N_x$, and $\Delta t = t_n - t_{n-1}$, $n = 1, \dots, N_t$. Figure 2.1 displays a mesh in the x, t plane with $N_t = 5$, $N_x = 5$, and constant mesh spacings.

2.1.2 The Discrete Solution

The solution $u(x, t)$ is sought at the mesh points. We introduce the mesh function u_i^n , which approximates the exact solution at the mesh point (x_i, t_n) for $i = 0, \dots, N_x$ and $n = 0, \dots, N_t$. Using the finite difference method, we shall develop algebraic equations for computing the mesh function.

2.1.3 Fulfilling the Equation at the Mesh Points

In the finite difference method, we relax the condition that (2.1) holds at all points in the space-time domain $(0, L) \times (0, T]$ to the requirement that the PDE is fulfilled at the *interior* mesh points only:

$$\frac{\partial^2}{\partial t^2} u(x_i, t_n) = c^2 \frac{\partial^2}{\partial x^2} u(x_i, t_n), \quad (2.10)$$

for $i = 1, \dots, N_x - 1$ and $n = 1, \dots, N_t - 1$. For $n = 0$ we have the initial conditions $u = I(x)$ and $u_t = 0$, and at the boundaries $i = 0, N_x$ we have the boundary condition $u = 0$.

2.1.4 Replacing Derivatives by Finite Differences

The second-order derivatives can be replaced by central differences. The most widely used difference approximation of the second-order derivative is

$$\frac{\partial^2}{\partial t^2} u(x_i, t_n) \approx \frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2}.$$

It is convenient to introduce the finite difference operator notation

$$[D_t D_t u]_i^n = \frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2}.$$

A similar approximation of the second-order derivative in the x direction reads

$$\frac{\partial^2}{\partial x^2} u(x_i, t_n) \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} = [D_x D_x u]_i^n.$$

Algebraic version of the PDE We can now replace the derivatives in (2.10) and get

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{\Delta t^2} = c^2 \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}, \quad (2.11)$$

or written more compactly using the operator notation:

$$[D_t D_t u = c^2 D_x D_x]_i^n. \quad (2.12)$$

Interpretation of the equation as a stencil A characteristic feature of (2.11) is that it involves u values from neighboring points only: u_i^{n+1} , $u_{i\pm 1}^n$, u_i^n , and u_i^{n-1} . The circles in Fig. 2.1 illustrate such neighboring mesh points that contribute to an algebraic equation. In this particular case, we have sampled the PDE at the point $(2, 2)$ and constructed (2.11), which then involves a coupling of u_1^2 , u_2^3 , u_2^2 , u_2^1 , and u_3^2 . The term *stencil* is often used about the algebraic equation at a mesh point, and the geometry of a typical stencil is illustrated in Fig. 2.1. One also often refers to the algebraic equations as *discrete equations*, *(finite) difference equations* or a *finite difference scheme*.

Algebraic version of the initial conditions We also need to replace the derivative in the initial condition (2.3) by a finite difference approximation. A centered difference of the type

$$\frac{\partial}{\partial t} u(x_i, t_0) \approx \frac{u_i^1 - u_i^{-1}}{2\Delta t} = [D_{2t} u]_i^0,$$

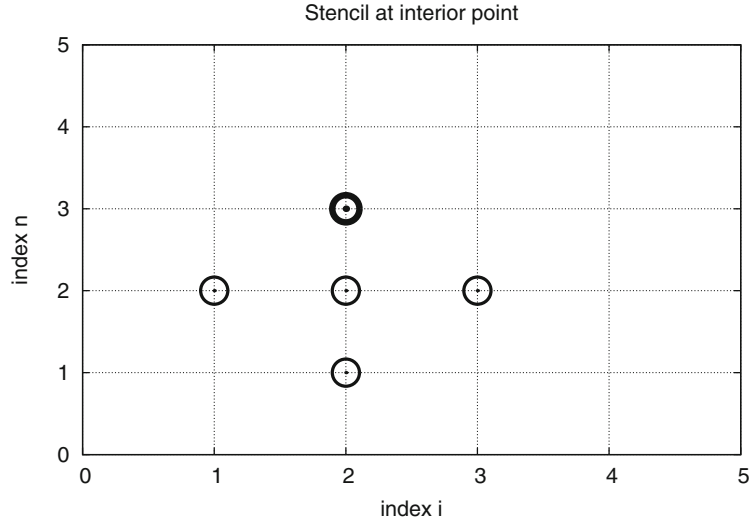


Fig.2.1 Mesh in space and time. The circles show points connected in a finite difference equation

seems appropriate. Writing out this equation and ordering the terms give

$$u_i^{-1} = u_i^1, \quad i = 0, \dots, N_x. \quad (2.13)$$

The other initial condition can be computed by

$$u_i^0 = I(x_i), \quad i = 0, \dots, N_x.$$

2.1.5 Formulating a Recursive Algorithm

We assume that u_i^n and u_i^{n-1} are available for $i = 0, \dots, N_x$. The only unknown quantity in (2.11) is therefore u_i^{n+1} , which we now can solve for:

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + C^2 (u_{i+1}^n - 2u_i^n + u_{i-1}^n). \quad (2.14)$$

We have here introduced the parameter

$$C = c \frac{\Delta t}{\Delta x}, \quad (2.15)$$

known as the *Courant number*.

***C* is the key parameter in the discrete wave equation**

We see that the discrete version of the PDE features only one parameter, C , which is therefore the key parameter, together with N_x , that governs the quality of the numerical solution (see Sect. 2.10 for details). Both the primary physical parameter c and the numerical parameters Δx and Δt are lumped together in C . Note that C is a dimensionless parameter.

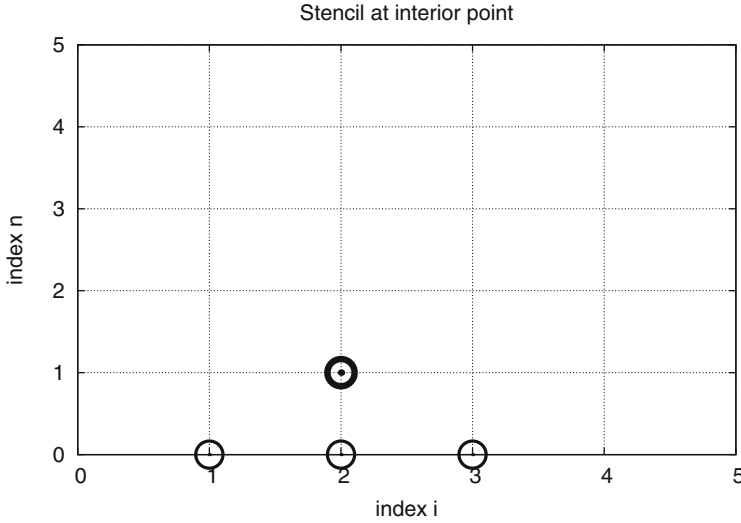


Fig. 2.2 Modified stencil for the first time step

Given that u_i^{n-1} and u_i^n are known for $i = 0, \dots, N_x$, we find new values at the next time level by applying the formula (2.14) for $i = 1, \dots, N_x - 1$. Figure 2.1 illustrates the points that are used to compute u_2^3 . For the boundary points, $i = 0$ and $i = N_x$, we apply the boundary conditions $u_i^{n+1} = 0$.

Even though sound reasoning leads up to (2.14), there is still a minor challenge with it that needs to be resolved. Think of the very first computational step to be made. The scheme (2.14) is supposed to start at $n = 1$, which means that we compute u^2 from u^1 and u^0 . Unfortunately, we do not know the value of u^1 , so how to proceed? A standard procedure in such cases is to apply (2.14) also for $n = 0$. This immediately seems strange, since it involves u_i^{-1} , which is an undefined quantity outside the time mesh (and the time domain). However, we can use the initial condition (2.13) in combination with (2.14) when $n = 0$ to eliminate u_i^{-1} and arrive at a special formula for u_i^1 :

$$u_i^1 = u_i^0 - \frac{1}{2}C^2(u_{i+1}^0 - 2u_i^0 + u_{i-1}^0). \quad (2.16)$$

Figure 2.2 illustrates how (2.16) connects four instead of five points: u_2^1 , u_1^0 , u_2^0 , and u_3^0 .

We can now summarize the computational algorithm:

1. Compute $u_i^0 = I(x_i)$ for $i = 0, \dots, N_x$
2. Compute u_i^1 by (2.16) for $i = 1, 2, \dots, N_x - 1$ and set $u_i^1 = 0$ for the boundary points given by $i = 0$ and $i = N_x$,
3. For each time level $n = 1, 2, \dots, N_t - 1$
 - (a) apply (2.14) to find u_i^{n+1} for $i = 1, \dots, N_x - 1$
 - (b) set $u_i^{n+1} = 0$ for the boundary points having $i = 0, i = N_x$.

The algorithm essentially consists of moving a finite difference stencil through all the mesh points, which can be seen as an animation in a [web page](#)¹ or a [movie file](#)².

2.1.6 Sketch of an Implementation

The algorithm only involves the three most recent time levels, so we need only three arrays for u_i^{n+1} , u_i^n , and u_i^{n-1} , $i = 0, \dots, N_x$. Storing all the solutions in a two-dimensional array of size $(N_x + 1) \times (N_t + 1)$ would be possible in this simple one-dimensional PDE problem, but is normally out of the question in three-dimensional (3D) and large two-dimensional (2D) problems. We shall therefore, in all our PDE solving programs, have the unknown in memory at as few time levels as possible.

In a Python implementation of this algorithm, we use the array elements $u[i]$ to store u_i^{n+1} , $u_n[i]$ to store u_i^n , and $u_nm1[i]$ to store u_i^{n-1} .

The following Python snippet realizes the steps in the computational algorithm.

```
# Given mesh points as arrays x and t (x[i], t[n])
dx = x[1] - x[0]
dt = t[1] - t[0]
C = c*dt/dx          # Courant number
Nt = len(t)-1
C2 = C**2            # Help variable in the scheme

# Set initial condition u(x,0) = I(x)
for i in range(0, Nx+1):
    u_n[i] = I(x[i])

# Apply special formula for first step, incorporating du/dt=0
for i in range(1, Nx):
    u[i] = u_n[i] - \
        0.5*C**2*(u_n[i+1] - 2*u_n[i] + u_n[i-1])
u[0] = 0; u[Nx] = 0 # Enforce boundary conditions

# Switch variables before next step
u_nm1[:], u_n[:] = u_n, u

for n in range(1, Nt):
    # Update all inner mesh points at time t[n+1]
    for i in range(1, Nx):
        u[i] = 2*u_n[i] - u_nm1[i] - \
            C**2*(u_n[i+1] - 2*u_n[i] + u_n[i-1])

    # Insert boundary conditions
    u[0] = 0; u[Nx] = 0

    # Switch variables before next step
    u_nm1[:], u_n[:] = u_n, u
```

¹ http://tinyurl.com/hbcasmj/book/html/mov-wave/D_stencil_gpl/index.html

² http://tinyurl.com/gokgkov/mov-wave/D_stencil_gpl/movie.ogg

2.2 Verification

Before implementing the algorithm, it is convenient to add a source term to the PDE (2.1), since that gives us more freedom in finding test problems for verification. Physically, a source term acts as a generator for waves in the interior of the domain.

2.2.1 A Slightly Generalized Model Problem

We now address the following extended initial-boundary value problem for one-dimensional wave phenomena:

$$u_{tt} = c^2 u_{xx} + f(x, t), \quad x \in (0, L), \quad t \in (0, T] \quad (2.17)$$

$$u(x, 0) = I(x), \quad x \in [0, L] \quad (2.18)$$

$$u_t(x, 0) = V(x), \quad x \in [0, L] \quad (2.19)$$

$$u(0, t) = 0, \quad t > 0 \quad (2.20)$$

$$u(L, t) = 0, \quad t > 0. \quad (2.21)$$

Sampling the PDE at (x_i, t_n) and using the same finite difference approximations as above, yields

$$[D_t D_t u = c^2 D_x D_x u + f]_i^n. \quad (2.22)$$

Writing this out and solving for the unknown u_i^{n+1} results in

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + C^2(u_{i+1}^n - 2u_i^n + u_{i-1}^n) + \Delta t^2 f_i^n. \quad (2.23)$$

The equation for the first time step must be rederived. The discretization of the initial condition $u_t = V(x)$ at $t = 0$ becomes

$$[D_{2t} u = V]_i^0 \Rightarrow u_i^{-1} = u_i^1 - 2\Delta t V_i,$$

which, when inserted in (2.23) for $n = 0$, gives the special formula

$$u_i^1 = u_i^0 - \Delta t V_i + \frac{1}{2} C^2 (u_{i+1}^0 - 2u_i^0 + u_{i-1}^0) + \frac{1}{2} \Delta t^2 f_i^0. \quad (2.24)$$

2.2.2 Using an Analytical Solution of Physical Significance

Many wave problems feature sinusoidal oscillations in time and space. For example, the original PDE problem (2.1)–(2.5) allows an exact solution

$$u_e(x, t) = A \sin\left(\frac{\pi}{L}x\right) \cos\left(\frac{\pi}{L}ct\right). \quad (2.25)$$

This u_e fulfills the PDE with $f = 0$, boundary conditions $u_e(0, t) = u_e(L, t) = 0$, as well as initial conditions $I(x) = A \sin\left(\frac{\pi}{L}x\right)$ and $V = 0$.

How to use exact solutions for verification

It is common to use such exact solutions of physical interest to verify implementations. However, the numerical solution u_i^n will only be an approximation to $u_e(x_i, t_n)$. We have no knowledge of the precise size of the error in this approximation, and therefore we can never know if discrepancies between u_i^n and $u_e(x_i, t_n)$ are caused by mathematical approximations or programming errors. In particular, if plots of the computed solution u_i^n and the exact one (2.25) look similar, many are tempted to claim that the implementation works. However, even if color plots look nice and the accuracy is “deemed good”, there can still be serious programming errors present!

The only way to use exact physical solutions like (2.25) for serious and thorough verification is to run a series of simulations on finer and finer meshes, measure the integrated error in each mesh, and from this information estimate the empirical convergence rate of the method.

An introduction to the computing of convergence rates is given in Section 3.1.6 in [9]. There is also a detailed example on computing convergence rates in Sect. 1.2.2.

In the present problem, one expects the method to have a convergence rate of 2 (see Sect. 2.10), so if the computed rates are close to 2 on a sufficiently fine mesh, we have good evidence that the implementation is free of programming mistakes.

2.2.3 Manufactured Solution and Estimation of Convergence Rates

Specifying the solution and computing corresponding data One problem with the exact solution (2.25) is that it requires a simplification ($V = 0, f = 0$) of the implemented problem (2.17)–(2.21). An advantage of using a *manufactured solution* is that we can test all terms in the PDE problem. The idea of this approach is to set up some chosen solution and fit the source term, boundary conditions, and initial conditions to be compatible with the chosen solution. Given that our boundary conditions in the implementation are $u(0, t) = u(L, t) = 0$, we must choose a solution that fulfills these conditions. One example is

$$u_e(x, t) = x(L - x) \sin t .$$

Inserted in the PDE $u_{tt} = c^2 u_{xx} + f$ we get

$$-x(L - x) \sin t = -c^2 2 \sin t + f \quad \Rightarrow \quad f = (2c^2 - x(L - x)) \sin t .$$

The initial conditions become

$$\begin{aligned} u(x, 0) &= I(x) = 0, \\ u_t(x, 0) &= V(x) = x(L - x) . \end{aligned}$$

Defining a single discretization parameter To verify the code, we compute the convergence rates in a series of simulations, letting each simulation use a finer mesh than the previous one. Such empirical estimation of convergence rates relies on an

assumption that some measure E of the numerical error is related to the discretization parameters through

$$E = C_t \Delta t^r + C_x \Delta x^p,$$

where C_t , C_x , r , and p are constants. The constants r and p are known as the *convergence rates* in time and space, respectively. From the accuracy in the finite difference approximations, we expect $r = p = 2$, since the error terms are of order Δt^2 and Δx^2 . This is confirmed by truncation error analysis and other types of analysis.

By using an exact solution of the PDE problem, we will next compute the error measure E on a sequence of refined meshes and see if the rates $r = p = 2$ are obtained. We will not be concerned with estimating the constants C_t and C_x , simply because we are not interested in their values.

It is advantageous to introduce a single discretization parameter $h = \Delta t = \hat{c} \Delta x$ for some constant \hat{c} . Since Δt and Δx are related through the Courant number, $\Delta t = C \Delta x / c$, we set $h = \Delta t$, and then $\Delta x = hc / C$. Now the expression for the error measure is greatly simplified:

$$E = C_t \Delta t^r + C_x \Delta x^r = C_t h^r + C_x \left(\frac{c}{C}\right)^r h^r = D h^r, \quad D = C_t + C_x \left(\frac{c}{C}\right)^r.$$

Computing errors We choose an initial discretization parameter h_0 and run experiments with decreasing h : $h_i = 2^{-i} h_0$, $i = 1, 2, \dots, m$. Halving h in each experiment is not necessary, but it is a common choice. For each experiment we must record E and h . Standard choices of error measure are the ℓ^2 and ℓ^∞ norms of the error mesh function e_i^n :

$$E = \|e_i^n\|_{\ell^2} = \left(\Delta t \Delta x \sum_{n=0}^{N_t} \sum_{i=0}^{N_x} (e_i^n)^2 \right)^{\frac{1}{2}}, \quad e_i^n = u_e(x_i, t_n) - u_i^n, \quad (2.26)$$

$$E = \|e_i^n\|_{\ell^\infty} = \max_{i,n} |e_i^n|. \quad (2.27)$$

In Python, one can compute $\sum_i (e_i^n)^2$ at each time step and accumulate the value in some sum variable, say `e2_sum`. At the final time step one can do `sqrt(dt*dx*e2_sum)`. For the ℓ^∞ norm one must compare the maximum error at a time level (`e.max()`) with the global maximum over the time domain: `e_max = max(e_max, e.max())`.

An alternative error measure is to use a spatial norm at one time step only, e.g., the end time T ($n = N_t$):

$$E = \|e_i^n\|_{\ell^2} = \left(\Delta x \sum_{i=0}^{N_x} (e_i^n)^2 \right)^{\frac{1}{2}}, \quad e_i^n = u_e(x_i, t_n) - u_i^n, \quad (2.28)$$

$$E = \|e_i^n\|_{\ell^\infty} = \max_{0 \leq i \leq N_x} |e_i^n|. \quad (2.29)$$

The important point is that the error measure (E) for the simulation is represented by a single number.

Computing rates Let E_i be the error measure in experiment (mesh) number i (not to be confused with the spatial index i) and let h_i be the corresponding discretization parameter (h). With the error model $E_i = Dh_i^r$, we can estimate r by comparing two consecutive experiments:

$$\begin{aligned} E_{i+1} &= Dh_{i+1}^r, \\ E_i &= Dh_i^r. \end{aligned}$$

Dividing the two equations eliminates the (uninteresting) constant D . Thereafter, solving for r yields

$$r = \frac{\ln E_{i+1}/E_i}{\ln h_{i+1}/h_i}.$$

Since r depends on i , i.e., which simulations we compare, we add an index to r : r_i , where $i = 0, \dots, m-2$, if we have m experiments: $(h_0, E_0), \dots, (h_{m-1}, E_{m-1})$.

In our present discretization of the wave equation we expect $r = 2$, and hence the r_i values should converge to 2 as i increases.

2.2.4 Constructing an Exact Solution of the Discrete Equations

With a manufactured or known analytical solution, as outlined above, we can estimate convergence rates and see if they have the correct asymptotic behavior. Experience shows that this is a quite good verification technique in that many common bugs will destroy the convergence rates. A significantly better test though, would be to check that the numerical solution is exactly what it should be. This will in general require exact knowledge of the numerical error, which we do not normally have (although we in Sect. 2.10 establish such knowledge in simple cases). However, it is possible to look for solutions where we can show that the numerical error vanishes, i.e., the solution of the original continuous PDE problem is also a solution of the discrete equations. This property often arises if the exact solution of the PDE is a lower-order polynomial. (Truncation error analysis leads to error measures that involve derivatives of the exact solution. In the present problem, the truncation error involves 4th-order derivatives of u in space and time. Choosing u as a polynomial of degree three or less will therefore lead to vanishing error.)

We shall now illustrate the construction of an exact solution to both the PDE itself and the discrete equations. Our chosen manufactured solution is quadratic in space and linear in time. More specifically, we set

$$u_e(x, t) = x(L - x) \left(1 + \frac{1}{2}t \right), \quad (2.30)$$

which by insertion in the PDE leads to $f(x, t) = 2(1 + t)c^2$. This u_e fulfills the boundary conditions $u = 0$ and demands $I(x) = x(L - x)$ and $V(x) = \frac{1}{2}x(L - x)$.

<http://www.springer.com/978-3-319-55455-6>

Finite Difference Computing with PDEs

A Modern Software Approach

Langtangen, H.P.; Linge, S.

2017, XXIII, 507 p. 150 illus., Hardcover

ISBN: 978-3-319-55455-6