INSTITUTE FOR PARALLEL AND DISTRIBUTED SYSTEMS

CHAIR OF SCIENTIFIC COMPUTING

SIMULATION TECHNOLOGY DEGREE COURSE

Bachelor thesis

Submitted to the University of Stuttgart

A new method for implicit time integration of the acoustic wave equation

First Supervisor

Prof. Dr. rer. nat. Dirk PFLÜGER

Institute for Parallel and Distributed Systems

Submitted by

Author	S. Nathanael RÖMHELD
Matriculation number	3261338
SimTech-Nr.	134
Submission date	April 2021

Abstract

Wave equations are usually simulated using explicit time integration methods. But the strong restriction on time step sizes in media with large differences in wave speeds can lead to high computational cost. Implicit time integration methods on the other hand can achieve better stability properties, which allow larger time steps. In this work, we consider diagonally implicit Runge-Kutta-based (DIRK) time integration schemes for second-order ordinary differential equations (ODEs) applied to the acoustic wave equation and attempt to identify efficient combinations of time integration schemes, Runge-Kutta coefficients, and linear system solver parameters.
We derive new general time integration schemes for second-order ODEs similar to the diagonally implicit Runge-Kutta-Nyström (DIRKN) scheme. But instead of reordering the DIRK equations to solve for the second temporal derivative of the acoustic pressure, they solve for the first derivative (DIRK-2-1) or the variable itself (DIRK-2-0). The latter scheme allows further substitutions when applied to linear ODEs like the acoustic wave equation. These substitutions lead to another scheme (DIRK-2-0s) that avoids matrix-vector multiplications and therefore has potential cost benefits. Because DIRK-2-1 has higher costs than the others, we do not include it in the performance analysis. We apply these time integration schemes to the acoustic wave equation and analyze their performance. To solve the arising linear systems, the conjugate gradient method with a multigrid preconditioner is used. We analyze the efficiency of the schemes with the coefficients of various highly-stable DIRK and DIRKN methods and for various configurations of the linear system solver.
The best configurations for the three schemes DIRKN, DIRK-2-0, and DIRK-2-0s perform similarly in regard to cost and accuracy. The new schemes do not outperform DIRKN consistently. While avoidance of the matrix-vector multiplications decreases the costs per time step in theory, DIRK-2-0s amplifies the error of inexact linear system solvers more than the alternatives. Therefore, it requires lower tolerances for the conjugate gradient method which increases its overall cost. While their performance is similar, DIRKN tends to deliver the most efficient results in a medium accuracy range, which is probably relevant for most applications. Regarding the choice of methods (sets of coefficients), a general recommendation is not possible. At different levels of accuracy, different methods are the most efficient. The effects of varying solver parameters like the conjugate gradient method's tolerance and the use of the Galerkin condition are also difficult to predict. Therefore, efficient combinations of a time integration scheme, Runge-Kutta coefficients, and linear system solver parameters should be determined experimentally for the desired level of accuracy.
The numerical experiments show that our approach is unlikely to outperform efficient explicit methods. While there are combinations that lead to lower costs than some explicit methods, their accuracy is also reduced. Increasing the time step size for implicit methods relative to explicit ones reduces their cost further but also makes them less accurate, even for relatively high temporal resolutions of the simulated waves. This shows that the beneficial effect of small time steps on accuracy should not be underestimated.

Contents

1 Introduction and literature overview

1.1 Application: Seismic surveys

In the petroleum industry, determining the properties of subsurface structures and inferring potential locations of natural resources like oil and gas is an essential task. Non-invasive seismic surveys, using the principles of reflection and refraction seismology, are commonly used. A controlled source, like an explosion, dropped weight, or vibration device mounted to a truck or ship, introduces seismic waves into the domain of interest. Multiple sensors (geophones or hydrophones) distributed over large areas on the surface record the seismic activity. Determining the structure of the ground from this data is an inverse problem. Many methods used to solve such problems are based on iterative methods using forward simulations (e.g. [Tar84; VO09]). Optimizing their efficiency is essential since they are responsible for most of the overall computational costs.
Usually, the elastic wave equation or the simpler acoustic wave equation is used as a model for the wave propagation. The latter can only describe P-waves (primary or pressure waves) and not S-waves (secondary or shear waves). But in some scenarios, this does not matter because the acoustic equation approximates the occurring waves well (in acoustic media). The P-waves also travel faster than S-waves, which is why they can be measured separately in some use-cases. The simplicity and lower simulation costs make the acoustic wave equation a popular choice in the seismic community. But both wave equations are second-order linear partial differential equations and can be often treated with the same numerical methods. Other variations such as the poroelastic wave equation are not as widely used and therefore not included in this discussion.

1.2 About this project

This project was enabled by the visiting student research program (VSRP) at the King Abdullah University of Science and Technology (KAUST) and the generous support of Dr. Longfei Gao and Prof. David Keyes. It continues a research project of Longfei Gao, which is aimed at determining whether efficient implicit time-stepping methods used in combination with modern linear system solvers can compete with more commonly used explicit techniques in terms of computational efficiency. This work continues and generalizes his unpublished approach [Gao19].
After working on it in KAUST under the supervision of Longfei Gao, it was continued at the University of Stuttgart under the supervision of Prof. Dirk Pflüger.
Longfei Gao's approach consists of using standard finite differences for the spatial discretization and a time-stepping scheme derived from an A-stable singly diagonal Runge-Kutta method (RK7 in Table 5.1). In this work, we derive the general time-stepping scheme behind his ideas (DIRK-2-0s) plus an intermediary version (DIRK-2-0). Both can be used with the coefficients of other diagonally implicit Runge-Kutta or Nyström methods. This allows a thorough comparison with other established time-stepping methods, such as the diagonally implicit Runge-Kutta-Nyström scheme (DIRKN).

Like in Longfei Gao's original proposal, the conjugate gradient method is used with a multigrid preconditioner. We also follow his proposal of analyzing the cost of the time-stepping scheme(s) using complexity analysis. This enables the identification of good values for various parameters in the time-stepping scheme and a comparison of different time-stepping schemes, implicit and explicit.

1.3 Literature overview: Numerical treatment of wave equations [Propädeuticum]

1.3.1 Spatial discretization methods for wave equations

Many different methods for the discretization and simulation of the elastic and acoustic wave equation have been developed over the years. The elastic wave equation is included in this section since it is a more general description that includes the acoustic case. This makes methods developed for the elastic formulation applicable to the acoustic version as well. In some cases the reverse is also true: Since both are linear differential equations, some methods developed in the context of the acoustic wave equation can be generalized for the elastic wave equation as well.

Established methods for the spatial discretization of the two wave equations include finite difference (FD), finite element (FEM), discontinuous Galerkin (DG), nearly-analytic discrete (NADM), pseudospectral (PS), and spectral element (SPEM) methods. All of these have been widely used, but their use-case-dependent advantages and disadvantages should be considered.

Methods based on finite differences (FD) are especially popular because of their widespread successful use, universal applicability, "fast speed, programming simplicity, low storage requirements, and high parallelism"[HYW15]. This was not always the case. Earlier publications (e.g. [ESS82; Muf85]) considered finite difference methods to be computationally expensive, especially compared to now less popular methods based on ray theory. But more powerful computers and greater interest in more accurate methods have led to a shift of perspective in the seismic community. Examples for the use of finite difference methods in the literature are [AKB74; BR97; CH99; Dab86; GDCK19; IMR95; KWTA76; RBS94; TG00].
The biggest disadvantage of finite difference methods is the high numerical dispersion when the grids are too coarse or too few samples per wavelength are used. Numerical dispersion is a frequency-dependent numerical error caused by inaccurate computation of derivatives based on truncated Taylor series expansions [LSJ+09]. High frequencies that are too fine for the grid get distorted and propagate at different speeds. Because higher-order methods only help to a certain degree and finer resolutions lead to high computational costs, other methods like the pseudo-spectral methods gained attention. But most publications introducing new finite difference methods for wave equations have addressed this problem to some extent. Multiple authors have found ways to reduce the numerical dispersion effectively, at least compared to standard finite difference methods.
There are various sub-divisions of finite difference methods, such as between the ones on standard and staggered grids, or sub-classes like compact finite difference methods or Lax-Wendroff correction methods ([LW64]). A short overview of various flavors of advanced FD methods can be found in [LSJ+09] and [LS09]. Publications also vary in whether they use FD only for the spatial or temporal discretization or both.
We use the standard finite difference method for spatial discretization because of its simplicity and

computational efficiency. Since the focus of this work lies on Runge-Kutta-based time integration and not on spatial discretization, we are not concerned about potentially high dispersion here. But our proposed temporal discretization could likely also be applied in combination with many other spatial discretization strategies, like finite element, discontinuous Galerkin, nearly-analytic, or other finite difference methods. The calculations regarding the computational complexity would have to be adapted.

Finite element methods (e.g [EJ91; KL73; LDB72; Mar84]) (meaning continuous Galerkin methods) allow the use of unstructured grids, which can minimize the numerical noise resulting from interfaces that are not aligned with grid points. Examples are the earth's surface or layers with different wave propagation speeds, but also scenarios like a dynamically rupturing fault, where geometries change over time. But such methods typically require the solving of large-scale linear systems, which requires a lot of computational resources and makes parallelization difficult.

Discontinuous Galerkin methods (e.g. [DK06; GSS06; HYMQ20; HYW15; KD06; RW03]) have gained much attention in the seismic community in recent years. They share the same advantages as FEM, but by allowing discontinuous basis functions, they are more flexible, which leads to further advantages. Solutions that are discontinuous across element interfaces are allowed, which enables more accurate representations of shocks and differences between adjacent layers with large velocity contrasts. Hanging nodes are allowed, which makes it easier to deal with complex structures and local grid refinement. Many discontinuous Galerkin methods achieve hp-adaptivity, which makes them more flexible in regard to locally varying polynomial degrees and element shapes. Their complete localization, which is reflected in the mass matrix, makes them well-suited for parallelization. Furthermore, they usually exhibit good conservation, stability, and convergence properties. [HYM20]

Pseudo-spectral methods (e.g. [Car94; KB82]) are exact up to the Nyquist frequency, which especially makes them more accurate than finite difference methods. But since they rely on the Fourier transform, these methods are computationally expensive. Furthermore, they should only be applied to media with smooth variations since sharp boundaries can cause problems. Another criticism of the methods is that each point in the domain influences the value of every other, which does not reflect the physics of wave propagation. [YLWP04]

Spectral element methods (e.g. [KT99; KV98; SP94]) combine the accuracy of pseudo-spectral methods with the flexibility regarding grid structures of finite element methods. They use diagonalization techniques to reduce the bandwidths of the resulting matrices in order to avoid the same drawbacks of FEM regarding high computational cost and difficult parallelization. But the need to solve linear systems persists, which makes them more computationally expensive than finite difference methods.

The nearly analytical discrete methods (NADM) (e.g. [KHI94; YLWP04; YPLT06; YTZL03]) rely on truncated Taylor series expansions like many finite difference methods. What differentiates them, is the use of the gradients of the wave displacement or pressure to approximate the higher-order spatial derivatives. Additionally, local interpolation is used to increase accuracy and suppress numerical dispersion. Because of its effectiveness, much coarser grids can be used, which can reduce the computational costs drastically.

Other methods that have been successfully used for wave field simulations include spectral or reflectivity (e.g. [BC83a; BC83b; Che93; ZCC03]), ray (e.g. [Cha78; Hel68; KK59]), and boundary integral methods (e.g. [Bou96; ZC06; ZC08]). But they are either less popular than the other mentioned methods nowadays, less flexible, or incompatible with conventional time stepping methods.

Hybrid methods between different approaches have also been suggested. The most popular way of doing this is splitting the computational domain into different parts and to solve each part with a method that is well suited for the contained geological structures. An example would be using FEM for complex interfaces like free surfaces or faults and FD for the rest of the domain (e.g. [GMK08; MAL04]). Another way is combining different methods for time and space (e.g. [AM+80]). With hybrid methods, improvements in accuracy and efficiency can be achieved while only using existing techniques.

1.3.2 Time integration methods for wave equations

Explicit and implicit methods

Before 1985, the propagation of seismic waves was simulated almost exclusively with explicit time-stepping schemes. The large scale of most seismic models already resulted in relatively high storage and CPU time requirements for that time [Muf85]. The additional computational cost of solving the large linear systems resulting from implicit approaches made – and still makes – them unattractive to many researchers. But the Courant-Friedrichs-Lewy (CFL) condition limits the time step size of explicit methods significantly, especially if the wave speed varies greatly in the medium (see Section 4.4.4). Implicit methods allow for much better stability properties, even unconditional stability, which allows arbitrarily large time steps. The larger step sizes could potentially reduce the overall computational costs.
Emerman, Schmidt, and Stephen [ESS82] were among the first in the seismic community to recognize this potential and suggest the use of implicit methods. But while their derived method is stable for larger time steps, it is inaccurate enough for them to conclude: *"We cannot recommend further work on implicit formulations of the elastic wave equation."* [ESS82]. Nevertheless, this approach was picked up again in [Muf85] and the authors demonstrated that the problems regarding accuracy can be alleviated. Since then, more implicit or implicit-based methods have been suggested, but explicit time-stepping remains popular due to its simplicity and low cost per time step.

Implicit approaches require the solving of large linear systems. For a method to be able to compete with the explicit alternatives, it needs to solve them efficiently. Researchers usually are not confident about the ability of standard linear system solvers to solve them efficiently but use approximations to either arrive at a linear system that is easier to solve or avoid the implicit dependencies altogether. Most of the existing publications can be assigned to one of two classes. The first class consists of methods that are based on finite differences and do not avoid the linear systems. Instead, they use splitting algorithms that replace the linear system with multiple tridiagonal systems, which can be solved efficiently.
The second class consists of methods that use techniques to avoid the linear systems and approximate the required value instead.

Implicit finite difference methods leading to tridiagonal linear systems

Emerman et al. [ESS82] derived an implicit finite difference method for the 2D elastic wave equation. Their approach consists of doubling the resolutions through the introduction of half rows and half columns in the spatial grid, as well as half steps in time. At the first time half step, velocities are computed in every other column, using only the spatial derivatives in direction of the rows, by solving a tridiagonal linear system. Then, the stresses in the columns in between are computed using these velocities. The missing stress and velocity values at the other rows are interpolated. At the second time half step, the same is repeated in the other dimension for the columns. Which finite difference formulas are used exactly is not stated and not obvious from the equations. Since only tridiagonal linear systems appear without the use of higher derivatives and the equations each only use the previous time half step, a finite difference method that is second-order in space and first-order in time would be plausible. While this method is stable for any time step size, it is less accurate than an explicit alternative at any step size. Furthermore, it is computationally more expensive, and increasing the time step size above the CFL limit leads to large errors. As cited previously, the authors do not recommend its use and were not confident about the potential of similar approaches.

Mufti derived a method for the self-adjoint version of the 2D acoustic wave equation in [Muf85]. He starts with a standard finite difference scheme that is second-order accurate in space and time. Despite him claiming that it is implicit, there are no equations to be solved. This method also produces inaccurate results, which the author attributes to numerical dispersion. To alleviate this, he uses the mean of the spatial derivatives at the time step currently being computed and at the one two steps earlier. This is essentially the Crank-Nicolson method. It introduces an implicit dependency that corresponds to a five-band linear system. Similar to the previously outlined method, only derivatives in one spatial dimension are used at every second temporal half step and the derivatives along the other dimension in the other steps, but without the half rows and columns. This leads to only two tridiagonal linear systems per time step that need to be solved instead of the five-band system, which can be achieved more efficiently. The author also provides proof that the method with the splitting algorithm is equivalent to the one without. The numerical experiments in the paper show low dispersive noise and that the new method is similar in accuracy to explicit methods.

In [KL07], Kim and Lim give explicit second- and fourth-order time-stepping schemes as well as a first or second-order scheme before introducing a new implicit method plus a splitting algorithm for the 3D acoustic wave equation. It is a three-level implicit method, which means that three time steps are involved in the equation of which two are known, and it has one design parameter $\theta \in [0, \frac{1}{2}]$. This parameter determines whether the scheme is unconditionally stable, fourth-order accurate, implicit or explicit, and three- or two-level.
Here, we use the following notations: The discrete pressure values at time step n are denoted as the vector P^n, the discrete spatial derivative operator as K, the source term at time step n as S^n, and the wave propagation speed as v. The finite difference operator for the second derivative in time used by the authors is denoted as $\bar{\partial}_{tt}$ with the time step width Δt,

$$\bar{\partial}_{tt} P^n := \frac{P^{n+1} - 2P^n + P^{n-1}}{\Delta t^2}. \tag{1.1}$$

The scheme has the following form:

$$\frac{1}{v^2}\bar{\partial}_{tt}P^n + K(P^n + \theta\Delta t^2\bar{\partial}_{tt}P^n) = S^n + \theta\Delta t^2\bar{\partial}_{tt}S^n \quad \Leftrightarrow \tag{1.2a}$$

$$(1 + \theta\Delta t^2 v^2 K)P^{n+1} = -(1 + \theta\Delta t^2 v^2 K)P^{n-1} + \left(2 + (2\theta - 1)\Delta t^2 v^2 K\right) P^n + \Delta t^2 v^2 \left(\theta S^{n-1} + (1 - 2\theta)S^n + \theta S^{n+1}\right). \tag{1.2b}$$

What sets this method apart from the previous ones is the summation of an additional finite-difference term where the spatial discretization is applied in Equation 1.2a. When comparing this equation to the source material, note that it is equivalent to equation (15) in the paper, but uses the following relationship:

$$P^n + \theta\Delta t^2\bar{\partial}_{tt}P^n = \theta P^{n-1} + (1 - 2\theta)P^n + \theta P^{n+1}. \tag{1.3}$$

To reduce the computational complexity, a locally one-dimensional (LOD) method is used, which is also known as an alternating direction implicit (ADI) method. Similar to the methods of Mufti and Emerman, the spatial discretization is applied separately for each dimension. If the split discretization operators K_x, K_y, K_z with $K = K_x + K_y + K_z$ are each equivalent to tridiagonal matrices, only three tridiagonal linear system need to be solved in three dimensions. The resulting algorithm with the identity matrix I is:

$$P^{n+1,0} = 2P^n - P^{n-1} + \Delta t^2 v^2 (S^n + \theta\Delta t^2\bar{\partial}_{tt}S^n - KP^n) \tag{1.4a}$$

$$(I + \theta\Delta t^2 v^2 K_x)P^{n+1,1} = P^{n+1,0} + \theta\Delta t^2 v^2 K_x(2P^n - P^{n-1}) \tag{1.4b}$$

$$(I + \theta\Delta t^2 v^2 K_y)P^{n+1,2} = P^{n+1,1} + \theta\Delta t^2 v^2 K_y(2P^n - P^{n-1}) \tag{1.4c}$$

$$(I + \theta\Delta t^2 v^2 K_z)P^{n+1} = P^{n+1,2} + \theta\Delta t^2 v^2 K_z(2P^n - P^{n-1}). \tag{1.4d}$$

The first equation is essentially an explicit time step, which yields an approximation for P^{n+1} and the other three implicit equations improve it. In contrast to the algorithm by Mufti, this splitting method introduces an additional error, which is in $O(\Delta t^4)$ and relatively large, but can be reduced to a level where it does not deteriorate the overall accuracy through the introduction of a correction term.
The comparison of the fourth-order three-level implicit method with $\theta = \frac{1}{12}$ to a three-level explicit method shows that they are similar in terms of stability, while the implicit one produces less numerical dispersion and is 40% more expensive. According to the authors, the low dispersion is especially relevant in very oscillatory media, where an explicit method might have to use smaller time steps to keep the errors low.

Zhang et al. [ZZS07] propose a scheme similar to the one by Mufti with some improvements for the 3D acoustic wave equation. The authors introduce the implicit splitting finite difference (ISFD) scheme, which does not use sub-time steps but auxiliary wave fields that are independent of each other. This leads to one finite difference equation per dimension which can be computed in parallel. These equations require the solving of one tridiagonal linear system each. The pressure value is computed at each time step as the average of the variables. Using this approach has the added benefit of eliminating numerical anisotropy. The ISFD scheme has two parameters that can be adjusted to minimize dispersion, increase stability, and increase spatial order. Since the scheme is based on central differences in space and time, one would expect it to only achieve second order in space and time. However, the authors demonstrate that with proper parameters ISFD can produce results of similar quality as explicit finite-difference schemes of higher order.

Chu and Stoffa [CS11] suggest a general theoretical framework for the derivation of implicit three-level time-stepping schemes for second-order-in-time wave equations, like the acoustic or elastic wave equation. They use the Pade expansion to derive coefficients. Since Taylor expansions are special cases of Pade expansions this approach can be interpreted as a generalization of methods based on Taylor expansions. The framework also includes a strategy for the analysis of stability properties and numerical dispersion. Furthermore, the authors provide a way to optimize the methods, which produces unconditionally stable schemes. Ways to solve the linear systems efficiently are not included in this publication.
The framework also provides an alternative derivation of the three-level implicit method introduced by Kim and Lim without the splitting algorithm.

The same authors propose a finite difference method that is implicit in time and space in [CS12]. The time discretization is based on the three-level implicit method (central difference in time) by Kim and Lim, but used with a slightly different splitting algorithm. The authors provide it with a neglected source term in the form

$$(1 - 3\theta\Delta t^2 v^2 \frac{\partial^2}{\partial x^2}) P_x^{n+1} = -\left(1 - 3\theta\Delta t^2 v^2 \frac{\partial^2}{\partial x^2}\right) P^{n-1} + 2[1 - 3(\theta\frac{1}{2})\Delta t^2 v^2 \frac{\partial^2}{\partial x^2}] P^n \tag{1.5a}$$

$$(1 - 3\theta\Delta t^2 v^2 \frac{\partial^2}{\partial y^2}) P_y^{n+1} = -\left(1 - 3\theta\Delta t^2 v^2 \frac{\partial^2}{\partial y^2}\right) P^{n-1} + 2[1 - 3(\theta\frac{1}{2})\Delta t^2 v^2 \frac{\partial^2}{\partial y^2}] P^n \tag{1.5b}$$

$$(1 - 3\theta\Delta t^2 v^2 \frac{\partial^2}{\partial z^2}) P_z^{n+1} = -\left(1 - 3\theta\Delta t^2 v^2 \frac{\partial^2}{\partial z^2}\right) P^{n-1} + 2[1 - 3(\theta\frac{1}{2})\Delta t^2 v^2 \frac{\partial^2}{\partial z^2}] P^n \tag{1.5c}$$

$$P^{n+1} = \frac{1}{3}\left(P_x^{n+1} + P_y^{n+1} + P_z^{n+1}\right). \tag{1.5d}$$

Despite their different appearance, the splitting method by Kim and Lim in Equation 1.4 is similar to this one (Equation 1.5). For the first equation, the only difference consists in the use of $3\frac{\partial^2}{\partial x^2}$ instead of only $\frac{\partial^2}{\partial x^2}$ as one-dimensional derivative operators. This also leads to having to weigh P_x^{n+1} with $\frac{1}{3}$ when calculating the time step result (see Equation 1.5d). But in the other equations, further differences exist. The method by Kim and Lim also has an iterative structure, where the result from one splitting equation is used in the next. The three equations in the algorithm by Chu and Stoffa, on the other hand, are independent of each other and can therefore be computed in parallel. This makes it similar to the approach by Zhang et al. [ZZS07]. The methods of Emerman et al. and Mufti on the other hand have interdependent sub-time steps. Despite these differences, Chu and Stoffa regard their method as an extension to the approaches of all three publications ([ESS82; Muf85; ZZS07]).
Analogously to the method by Zhang et al. [ZZS07], the splitting algorithm used here also does not introduce numerical anisotropy. Whether the method only requires solving tridiagonal linear systems or not is determined by the spatial discretization method used.
While the approach outlined above can be used with any spatial discretization, Chu and Stoffa explore the possibility of using an implicit spatial finite difference operator, which can be integrated into the other implicit equations. They show that with proper implementation it is possible to use an implicit spatial discretization without additional cost over explicit ones when implicit time integration is used. But even when a non-optimal implementation is used, the additional cost can be offset by significant improvements in terms of accuracy.

FD methods implicit in space

For this thesis, we focus on temporally implicit and spatially explicit methods since our goal is to increase efficiency by increasing the time step size. Nevertheless, it may prove fruitful to be

aware of methods that are implicit in space, since they suffer from the same problem of having to solve linear systems efficiently. Furthermore, the paper by Chu and Stoffa [CS12] shows that the combination of methods that are implicit in time and space can improve accuracy without a considerable increase in cost.
Difference operators that only lead to tridiagonal linear systems are popular for spatially implicit methods too. The papers [KPE08; KPT10; LS09] contain examples of such methods. All of them use a recursive second derivative operator, which can achieve arbitrarily high orders while only having to solve tridiagonal linear systems. But the number of linear systems which have to be solved increases with the order. The use of spatially implicit operators decreases dispersion and can produce results with similar accuracy as higher-order explicit operators [LS09].

There has been a great interest in spatially implicit finite difference methods in the seismic community recently. For an overview of different strategies and recent developments, see e.g. [RL19; WL18; YYL17].

Predictor-corrector methods

Not all implicit approaches to wave equations in the literature are based on finite-difference time-stepping combined with a splitting algorithm. There are also several papers where implicit time-stepping schemes are used in combinations with methods that deliver approximations for values that would have to be computed from implicit equations. This essentially transforms them into explicit schemes and therefore likely prevents unconditional stability. But the authors referenced in this section still achieve good stability properties with this approach.
In [YWL12], Yang et al. propose the use of an explicit method for the computation of the stage values in the two-stage, 3rd-order singly diagonally implicit Runge-Kutta (SDIRK) method found in [HNW93]. This method is not included in our selection of singly diagonal Runge-Kutta methods (see Section 5.1) because does not seem to be A-stable. But their approach would be applicable to other diagonally implicit Runge-Kutta methods as well. The used explicit method is a strong stability-preserving m-step Runge-Kutta method, which the authors also call a predictor-corrector method or SSPC for short.
Let us assume that the equation which ordinarily would need to be solved at the SDIRK stage s is

$$Q_s^n = K\left(P^n + \sum_{k=1}^{s} a_{s,k}\Delta t Q_k^n\right) + S(t_n + c_s\Delta t) \tag{1.6}$$

with the time step result P^n at the time step n, the stage result for its temporal derivative Q_k^n at stage the k, the corresponding time t_n, the time step width Δt, the Runge-Kutta stage time coefficients c_k, a source term S, the discrete derivation operator K, and the Runge-Kutta coefficients $a_{s,k}$. The latter is determined by the NADM here, but any spatial discretization method that yields a linear

operator is compatible with this approach. The explicit m-step Runge-Kutta scheme, which is used to compute Q_s^n instead of solving the linear system equivalent to Equation 1.6, is

$$Q_s^{(0)} = K\left(P^n + \sum_{k=1}^{s-1} a_{s,k}\Delta t Q_k^n\right) \tag{1.7a}$$

$$Q_s^{(i)} = \sum_{j=0}^{i-1}\left(\alpha_{i,j}Q_s^{(j)} + a_{s,s}\beta_{i,j}\Delta t K Q^{(j)}\right), \quad i = 1, 2, \ldots, m \tag{1.7b}$$

$$Q_s^n = Q_s^{(m)} + S(t_n + c_s\Delta t). \tag{1.7c}$$

The variables $\alpha_{i,j}, \beta_{i,j}$ are the coefficients of the m-step method. Equation 1.7a is referred to as the predictor and Equations 1.7b and 1.7c as corrector steps.
Regardless of the number of levels and coefficients, the scheme can be expressed as

$$Q_s^{(0)} = KP^n + \sum_{k=1}^{s-1} a_{s,k}\Delta t Q_k^n \tag{1.8a}$$

$$Q_s^n = \sum_{i=0}^{m} \gamma_i (a_{s,s}\Delta t)^i K^i Q_s^{(0)} + S(t_n + c_s\Delta t) \tag{1.8b}$$

with some coefficients $\gamma_i \in \mathbb{R}$. If the operators K^i for the higher derivatives, which are equivalent of to the application of K i times, are time-independent, they can be computed beforehand and only their application contributes to the cost of this method. Ideally, their stencils could be derived so these operators would not need to be stored as matrices. Whether this is the case and how large they get depends on the used discretization method.
The two-step method used in the paper has the coefficients $m = 2, \alpha_{1,0} = 1, \beta_{1,0} = 1, \alpha_{2,0} = \eta, \alpha_{2,1} = 1 - \eta, \beta_{2,0} = 0, \beta_{2,1} = 1, \eta \in [0, 1]$. This leads to the following formula,

$$Q_s^n = Q_s^{(0)} + (2 - \eta)a_{s,s}\Delta t K Q_s^{(0)} + (a_{s,s}\Delta t)^2 K^2 Q_s^{(0)} + S(t_n + c_s\Delta t). \tag{1.9}$$

When the parameter η is chosen as $\eta = 1$, one obtains the method proposed for the same SDIRK method in [YWCS09]. There, it was derived as a differentiator series method (DSM) (see [HH99]), which is essentially a truncated Taylor expansion using the discrete derivative operator.
In [YWD10], the same method is applied to a two-step, third-order implicit linear multistep method. Naturally, the resulting equations are different, but the approach is based on the same DSM method. Here, the implicit equation needs to be solved for the value P^n, instead of its derivative Q_s^n like in the previous cases. This is an interesting parallel to our work since one of the differences between our newly derived schemes and DIRKN is whether the Runge-Kutta equations are solved for the acoustic pressure or its second derivative.
Both papers also recommend a specific order of computing the values in order to avoid having to compute the highest-order spatial derivatives, which they call split-step algorithm (SSA). It consists of first computing $\bar{P}_s^n := P^n + \sum_{k=1}^{s-1} a_{s,k}\Delta t Q_k^n$, then $Q_s^{(0)} = K\bar{P}_s^n$, then $K^2 Q_s^{(0)}$ and finally $K^2\bar{P}_s^n$.
The publications [WYL12] and [WZ14] also use the presented SSPC scheme, but with $\eta = 0$, and the SSA. They also provide references to a similar method used by Shu and Osher [Shu88; SO88].
The publications [HYW15] and [HYM20] also use the same SDIRK method. To avoid having to solve the linear systems, they propose an iterative procedure, which they call "weighted Runge-Kutta

time discretization"(WRK). The SDIRK stage equation (Equation 1.6) with neglected source term is transformed into an iterative scheme,

$$Q_s^{(k+1)} = \Delta t a_{s,s} \Delta t Q_s^{(k)} + K \left(P^n + \sum_{k=1}^{s-1} a_{s,k} \Delta t Q_k^n \right). \tag{1.10}$$

As a starting value,

$$Q_s^{(0)} = K P^n + \sum_{k=1}^{s-1} a_{s,k} \Delta t Q_k^n \tag{1.11}$$

is used, which is identical to the previously used predictor in Equation 1.7a. The number of iterations depends on the desired accuracy, with k iterations delivering a $(k+1)$-th order approximation. If the result after two iterations is used, as suggested by the authors, the corresponding formula is the same as Equation 1.9 with $\eta = 1$, which we labeled as DSM. But in contrast to previous methods, the authors suggest using an additional parameter $\mu \in [0, 1]$. Instead of just using the value produced by the last iteration, a weighted average of the last two is used. The second-to-last is weighted with $(1 - \mu)$ and the last with μ. This can also be expressed as a direct formula,

$$Q_s^n = (1-\mu) Q_s^{(1)} + \mu Q_s^{(2)} = Q_s^{(0)} + a_{s,s} \Delta t K Q_s^{(0)} + \mu (a_{s,s} \Delta t)^2 K^2 Q_s^{(0)}. \tag{1.12}$$

Choosing $\mu < 1$ leads to a decrease of accuracy by one order but has advantages for numerical stability and dispersion. Note that the same formula can be derived from the SSPC scheme in Equation 1.9 if $\beta_{2,1} = \mu$ is chosen. Whether or not this would be compliant with the order conditions of m-step Runge-Kutta methods is left for the interested reader to determine.

All of the papers mentioned above derive their time-stepping method for the elastic wave equation and transform the second-order ODE into a first-order system of ODEs.
The methods in [HYM20; HYW15] use a discontinuous Galerkin method for the spatial discretization and the others NADM. But all methods are compatible with all spatial discretization methods that result in a linear derivative operator.
The reason why there are often two publications with the same approach is that the papers [HYW15; WYL12] introduce and analyze their respective algorithm for 2D scenarios while the papers [HYM20; WZ14] extend them to 3D. The same could be said for the publications[YWCS09] and [YWD10], but they have the additional difference that the former applies the DSM-SSA approach to the SDIRK method and the latter to an implicit linear multistep method.

All papers conclude that their approach reduces the numerical dispersion significantly compared to finite difference methods such as fourth-order Lax-Wendroff-Correction methods (LWC) or staggered-grid finite difference methods. Since these methods differ in spatial and temporal discretization, it is not clear to which extent the outlined time integration methods contribute to this suppression of numerical dispersion. The newly introduced methods are consistently more expensive than the alternatives they are compared to. But the reduced numerical dispersion allows the new methods to use much coarser grids. In [WZ14] the authors report that using only three points per minimum wavelength still produces good results. Furthermore, the introduced methods are usually more stable than the finite-difference alternatives, which allows larger time steps and further reduces costs. The actual amount varies. The IRK-DSM only has a slightly larger Courant number than the LWC, while it is 2 to 3.5 times larger than for the DG-based methods compared to TVD-RK-DG methods. None of the methods are free from time step limitations as A-stable methods would be. Larger time steps also reduce the numerical dispersion in their experiments.

When the numerical dispersion of the SSPC-based methods is matched with that of rival methods by increasing the spatial discretization and using the maximum time step size, the new methods become much more efficient. For example, in an experiment in [YWL12] the SSPC method is 70 times faster (CPU time) than a fourth-order LWC and uses 16% of its storage. Compared to a fourth-order staggered grid finite difference method, SSPC is 29 times faster and uses 6% of its storage. More details on the scenarios and values of the other methods can be found in their respective publications.
The authors of the papers also imply that the numerical dispersion error dominates the overall error in their numerical experiments. This means that it is likely beneficial to sacrifice algebraic order for better dispersion and stability properties. Examples for this are all presented SSPC-based methods that do not use a value of $\eta = 1$.
Comparisons between the different discussed methods are not undertaken by the authors. The only exception is [YWL12], where it is stated that values of $\eta < 1$ lead to more practically useful methods than the DSM variants with $\eta = 1$ because of the reduced dispersion.

Other implicit methods

Besides the two outlined classes of implicit methods, hardly any others seem to exist for the acoustic and elastic wave equation.
However, implicit finite difference schemes have also been suggested for other use-cases in seismology, like migration problems (e.g. [RR97; Sha07]).
Some have also been developed for poroelastic wave equations (e.g. [IIP16]), as well as plenty in areas outside seismology (see [LS09] for exemplary references). But these applications are beyond the scope of this work.

General-purpose time integration methods and stability

Besides the aforementioned approaches, general numerical time integration methods can also be applied to a spatially discretized wave equation. We are especially interested in unconditionally stable methods, so the time step size is only limited by concerns regarding accuracy. Appendix A contains a proof by Longfei Gao [Gao19] that the eigenvalues of the evolution matrix for the acoustic wave equation discretized with standard finite differences all lie on the imaginary axis. Methods that are A(α)-stable are therefore not suited for this application. Since absorbing boundary conditions can lead to eigenvalues with a negative real part, it is useful to account for this possibility. To fulfill these desired properties, at least A-stability is required for first-order ODE integrators. This also includes S- and L-stable methods. L-stability leads to quicker damping of oscillations, which can be useful for stiff equations [HW96]. S-stability extends the concept of A-stability to non-linear ODEs [PR74]. But since the acoustic wave equation is linear it does not provide additional benefits here. For second-order ODE integrators, a similar stability class does not exist or is not commonly used because the usually oscillatory behavior of the ODEs requires other test equations. Only a few publications, like [Con93], use the term A-stability in the context of second-order ODEs as well. Dahlquist's definition of unconditional stability [Dah78] may also be relevant in this context. More commonly used is the concept of R-stability which guarantees that the amplitude of the numerical solution does not increase for all time step sizes. Special cases of it are P-stability, which indicates an absence of numerical dissipation, and RL-stability, which denotes complete dissipation

of oscillations at infinity [SFB90], similarly as L-stability does for first-order methods. Note that the conventional definition of P-stability has been deemed insufficient (and redefined) by [ACM06]. We refer to methods for first or second-order ODEs that satisfy any of these stability criteria as unconditionally stable methods.

Multiple different time integration schemes for first-order ODEs exist, but few can be A-stable at arbitrary orders. Linear multistep methods for example can only achieve it with first- and second-order methods, as the Dahlquist barrier states [Dah63]. The most popular group of methods that can be A-stable for algebraic orders higher than two are the implicit Runge-Kutta methods.
For second-order ODEs, fewer time integration schemes have been developed. Instead of using such schemes, the differential equations are often transformed into systems of first-order ODEs and integrated with first-order methods. Linear multistep methods for second-order ODEs also exist but are limited in the stability they can achieve here too. Van der Houwen and Sommeijer prove in [HS89a] that these methods cannot be P-stable). One of the more popular options is the class of Runge-Kutta-Nyström methods (RKN) [HNW93; Nys26]. They can be interpreted as a more efficient and general version of Runge-Kutta methods applied to second-order ODEs. Like them, RKN can also have good stability properties at arbitrary orders if they are implicit.

Of these methods, the diagonally implicit Runge-Kutta (DIRK) and Nyström (DIRKN) methods are especially attractive for our use-case because of their potential for good stability properties and computational efficiency. In their equations, the value of a stage result depends only on previously computed values and ones of the current stage, but not on future ones. This corresponds to lower triangular matrices in the corresponding Butcher tableaus. The diagonally implicit methods allow for the resulting systems of equations at the stages to be solved successively instead of having to solve them all at once as is the case for fully implicit methods. They can still achieve the same stabilities as fully implicit methods but are less computationally expensive.
For various publications containing highly-stable DIRK or DIRKN methods, see Table 5.1.
Note that publications focusing on time integration methods in general, independent of specific applications, are usually not concerned with how the resulting systems of equations might be solved efficiently. Furthermore, they typically assume nonlinear ODEs, which excludes linear system solvers from consideration. If a solver is mentioned, it is often Newton's method, which requires solving one, possibly non-sparse, linear system per iteration. Newton's method is therefore less efficient than linear solvers if the ODE is linear, which is the case for the elastic and acoustic wave equation.

1.4 Overview of this thesis

Chapter 2 introduces the used notations for the acoustic wave equation and its discretized form resulting from the application of a finite difference method. Furthermore, it introduces the Runge-Kutta formulas from which the different time-stepping schemes are derived. This derivation is demonstrated for the diagonally implicit Runge-Kutta-Nyström scheme. Details on the multigrid preconditioned conjugate gradient method used to solve the arising linear systems follow.
Chapter 3 contains the derivation of two diagonally implicit Runge-Kutta-based time integration schemes, DIRK-2-1 and DIRK-2-0, that can be used instead of DIRKN. For DIRK-2-0, an alternative formulation, which uses substitutions to avoid the matrix-vector multiplications, is derived for the acoustic wave equation. The resulting scheme is called DIRK-2-0s.

Chapter 4 contains comparisons of the different time integration schemes considering various aspects. These are put into perspective by also including the explicit Runge-Kutta-Nyström scheme (ERKN). All comparisons are only based on the schemes' equations and not on experimental results regarding their performance. The chapter starts with a summary of the different schemes and a comment on the use of DIRKN coefficients with the newly derived schemes. A comparison with regard to the amplification of the CG error follows. The costs of the schemes and the linear solver with preconditioner are quantified using complexity analysis. This allows the identification of scenarios in which a specific scheme is most efficient in theory and the conditions that have to be met for implicit methods to be potentially more efficient than explicit ones. These calculations also involve the derivation of time step sizes for both. Furthermore, different initial guess strategies are introduced and their costs quantified.
In Chapter 5, different numerical experiments using the Marmousi2 wave speed model provide insight into the performance of the different schemes. We use 10 DIRKN and 21 DIRK methods with DIRKN, DIRK-2-0, and DIRK-2-0s to allow comparisons of the three schemes independent of the used method. The first experiment uses a direct solver to demonstrate the analytical equivalence between the schemes and to compare the errors originating from the used methods. The second one demonstrates the changes when an inexact solver is used and the iterations required by the unpreconditioned CG method for the different RK(N) methods. Then a multigrid preconditioner is introduced and its effects on iterations and errors analyzed. The third experiment evaluates the different initial guess strategies. The fourth one identifies the combinations of schemes and methods that benefit from using the Galerkin condition and the ones that are more efficient without it. The fifth and last experiment compares different configurations regarding the accuracy and cost they produce. This allows the identification of the most efficient implicit configurations and a comparison with an explicit time integration method.
Chapter 6 summarizes our findings and points out potential starting points for further investigations into the improvement of implicit time integration methods' efficiency.

2 Preliminaries

2.1 The acoustic wave equation

We concern ourselves with the acoustic wave equation, which is also known as scalar wave equation, in the following form:

$$\frac{1}{v(\mathbf{x})^2}\frac{\partial^2 p(t,\mathbf{x})}{\partial t^2} = \Delta p(t,\mathbf{x}) + s(t,\mathbf{x}), \tag{2.1}$$

with the time t, location vector $\mathbf{x}$ and the spatial Laplace operator Δ. The variable p denotes the acoustic pressure, which the equation is solved for. The wave speed given by the function v can vary within the domain. As initial conditions, we assume p and its derivatives to be zero. Waves are introduced to the domain through the source term s.

2.2 Spatial discretization

For the spatial discretization of the equation, we choose standard finite differences on an equidistant grid. This yields the semi-discrete acoustic wave equation as a system of ordinary differential equations in time with one equation per grid point,

$$M\frac{\partial^2 P(t)}{\partial t^2} = KP(t) + S(t). \tag{2.2}$$

For a 3D domain with the spatial resolution $N_x \times N_y \times N_z$, and thus $N = N_x \cdot N_y \cdot N_z$ grid points, P and S are vectors with N entries. M is a diagonal $N \times N$ matrix with the squared inverses of the wave speeds at the grid points as diagonal entries. It is also referred to as mass matrix. K is the $N \times N$ stiffness matrix resulting from the finite difference discretization of the Laplace operator. In the case of a second order approximation (central differences) in two dimensions, which will be used in the numerical experiments, the resulting stencil for an equidistant regular grid with the resolution Δx would be

$$\frac{1}{\Delta x^2}\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}. \tag{2.3}$$

This corresponds to a 5-band structure in the matrix K, potentially with some modifications from the boundary conditions.

2.2.1 First order form

Splitting a second-order ordinary differential equation in a system of first-order equations increases the number of applicable time integration schemes. It especially allows us to use Runge-Kutta formulas as a basis for the derivation of time integration schemes for second-order ODEs.
For a general ODE with the same structure as the model problem,

$$x''(t) = f(t, x), \tag{2.4}$$

we denote the corresponding system as

$$x'(t) = u(t, x) \tag{2.5a}$$
$$u'(t, x) = f(t, x). \tag{2.5b}$$

Applied to the semi-discrete acoustic wave equation, we use the notation

$$\frac{\partial P(t)}{\partial t} = Q(t) \tag{2.6a}$$
$$\frac{\partial Q(t)}{\partial t} = M^{-1}(KP(t) + S(t)). \tag{2.6b}$$

Depending on the integration scheme, a different but equivalent way of splitting the equation is advantageous. We use the following variant for DIRK-2-0(s), which keeps M on the left-hand side of the first equation,

$$M\frac{\partial P(t)}{\partial t} = Q(t) \tag{2.7a}$$
$$\frac{\partial Q(t)}{\partial t} = KP(t) + S(t). \tag{2.7b}$$

This introduces an alternative definition of Q but is of minor importance here. Further clarification for the different variable definitions in the different schemes is provided in Section 4.1.
The temporal derivatives of u and Q are denoted by $r = \frac{\partial u}{\partial t} = u'$ and $R = \frac{\partial Q}{\partial t}$, respectively.

2.3 Diagonally implicit Runge-Kutta methods

We use diagonally implicit Runge-Kutta methods (DIRK) as a starting point for our new schemes. For the general equation $x''(t) = f(t, x)$, with the first order form

$$x'(t, x) = u(t, x) \tag{2.8a}$$
$$u'(t, x) = r(t, x) \tag{2.8b}$$
$$r(t, x) = f(t, x), \tag{2.8c}$$

the diagonally implicit Runge-Kutta (DIRK) formulas for m stages at the time step N with $t^N = N \cdot \Delta t$ are the following:

$$x_i = x^N + \Delta t \sum_{j=1}^{i} a_{ij} u_j, \quad \forall i = 1, 2, \ldots, m \tag{2.9a}$$

$$u_i = u^N + \Delta t \sum_{j=1}^{i} a_{ij} r_j, \quad \forall i = 1, 2, \ldots, m \tag{2.9b}$$

$$r_i = f(t^N + c_i \Delta t, x_i), \quad \forall i = 1, 2, \ldots, m \tag{2.9c}$$

$$x^{N+1} = x^N + \Delta t \sum_{i=1}^{m} b_i u_i \tag{2.9d}$$

$$u^{N+1} = u^N + \Delta t \sum_{i=1}^{m} b_i r_i, \tag{2.9e}$$

with the corresponding Runge-Kutta coefficients in Butcher Tableau notation:

$$\begin{array}{c|ccc} c_1 & a_{11} & & \\ \vdots & \vdots & \ddots & \\ c_m & a_{m1} & \ldots & a_{mm} \\ \hline & b_1 & \ldots & b_m \end{array} . \tag{2.10}$$

Note that for singly diagonally implicit Runge-Kutta (SDIRK) methods the diagonal entries in the Butcher tableau are identical,

$$a_{11} = a_{22} = ... = a_{mm}, \tag{2.11}$$

and for singly diagonally implicit Runge-Kutta methods with an explicit first stage (ESDIRK) the diagonal entries are identical except for the first one, which is zero. The first three equations for x_i, u_i and r_i, which are also referred to as stage equations, must be computed at each of the m stages, $i = 1, 2, \ldots, m$. The notation $\forall i = 1, 2, \ldots, m$ is omitted for all following equations. The last and second-to-last formulas for u^{N+1} and x^{N+1} combine the stage results and deliver the values at the next time step. For this reason, we also refer to them as stage equations.
In the case of explicit Runge-Kutta methods, one could simply compute u_i, then x_i, and then r_i with the results of the previous stages. But here, the stage variables all depend on each other: x_i on u_i, u_i on r_i, and r_i on x_i. The naive approach would be to treat the stage equations as one large system of equations, e.g. with the variable vector $[x_i, u_i, r_i]^T$, and solve it as such. The better alternative is to plug the equations into each other in order to arrive at one equation that can be solved for one variable. This yields an equation system that is only one-third the size of the naive approach, which makes solving it more efficient. For linear ODEs like the acoustic wave equation, the function f is linear, which leads to a linear system and allows the use of linear solvers instead of more computationally expensive non-linear solvers.
Solving the equations for r_i leads to Runge-Kutta-Nyström methods. We derive the two schemes that result from solving the equations for u_i and p_i in the next chapter.

2.4 Diagonally implicit Runge-Kutta-Nyström methods

Nyström methods were introduced by Nyström in [Nys26] as a way of directly applying Runge-Kutta-style integration to second-order differential equations. The methods were extended over time and can be found in their general form in [HNW93].

To arrive at one equation that can be solved for r_i, we start with the RK formula for r_i (Equation 2.9c) and first substitute the x_i using Equation 2.9a and then the u_j using Equation 2.9b,

$$\begin{aligned} r_i &= f(t^N + c_i\Delta t, x_i) \\ &= f\left(t^N + c_i\Delta t, x^N + \Delta t \sum_{j=1}^{i} a_{ij} u_j\right) \\ &= f\left(t^N + c_i\Delta t, x^N + \Delta t \sum_{j=1}^{i} a_{ij}\left(u^N + \Delta t \sum_{k=1}^{j} a_{jk} r_k\right)\right). \end{aligned} \tag{2.12}$$

It might seem beneficial to only substitute the unknown u_i in the equation for r_i and keep the previously computed u_j in the equation, but that would increase the number of required operations compared to the final version of this time integration scheme. Note that these substitutions eliminate the implicit dependencies on the unknowns x_i and u_i, but introduce one r_i on the right-hand side, which necessitates the use of a solver.
We can avoid having to compute the u_i at all by substituting them in the Equation 2.9d for x^{N+1} as well,

$$x^{N+1} = x^N + \Delta t \sum_{i=1}^{m} b_i \left(u^N + \Delta t \sum_{j=1}^{i} a_{ij} r_j\right). \tag{2.13}$$

These equations can be simplified through the introduction of new coefficients,

$$\bar{a}_{ij} = \sum_{k=1}^{i} a_{ik} a_{kj} \quad \text{and} \quad \bar{b}_i = \sum_{j=1}^{i} b_j a_{ji}, \tag{2.14}$$

and using the following properties of Runge-Kutta coefficients:

$$\sum_{j=1}^{m} a_{ij} = c_i \quad \text{and} \quad \sum_{i=1}^{m} b_i = 1. \tag{2.15}$$

The resulting scheme is:

$$r_i = f\left(t^N + c_i\Delta t, x^N + c_i\Delta t u^N + \Delta t^2 \textstyle\sum_{j=1}^{i} \bar{a}_{ij} r_j\right) \tag{2.16a}$$

$$x^{N+1} = x^N + \Delta t u^N + \Delta t^2 \textstyle\sum_{i=1}^{m} \bar{b}_i r_i \tag{2.16b}$$

$$u^{N+1} = u^N + \Delta t \textstyle\sum_{i=1}^{m} b_i r_i. \tag{2.16c}$$

Nyström generalized these methods by allowing coefficients that do not necessarily satisfy the equations for $\bar{a}$ and $\bar{b}$. He derived order conditions for the coefficients and did not derive the equations from the first-order Runge-Kutta scheme.

DIRKN applied to the acoustic wave equation in the form of Equation 2.6 yield the following equations:

$$R_i = M^{-1}K\left(P^N + c_i\Delta t Q^N + \Delta t^2 \textstyle\sum_{j=1}^{i} \bar{a}_{ij} R_j\right) + M^{-1} S(t^N + c_i\Delta t) \tag{2.17a}$$

$$P^{N+1} = P^N + \Delta t Q^N + \Delta t^2 \textstyle\sum_{i=1}^{m} \bar{b}_i R_i \tag{2.17b}$$

$$Q^{N+1} = Q^N + \Delta t \textstyle\sum_{i=1}^{m} b_i R_i. \tag{2.17c}$$

Reordering the first row and a multiplication with M yields the linear system

$$(M - \Delta t^2 \bar{a}_{ii} K) R_i = K \left(P^N + c_i \Delta t Q^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} R_j \right) + S(t^N + c_i \Delta t). \tag{2.18}$$

The multiplication reduces the required number of operations and makes the system matrix symmetrical because M is diagonal and the finite difference matrix K is symmetric. Without it, the non-diagonal entries of the system matrix introduced by K would be multiplied by the wave speeds at their respective points. This would cause asymmetry for heterogeneous media.
When the diagonal elements of the Butcher tableau a_{ii} are chosen identically the system matrices are the same for all stages. Such methods are called singly diagonally implicit Runge-Kutta-Nyström methods (SDIRKN). Methods with identical diagonal elements except for zeros, which correspond to explicit stages, have the same benefit of identical system matrices. The most popular variant of this is the explicit first stage singly diagonally implicit Runge-Kutta scheme (ESDIRK).
The final scheme is

$$\hat{P}_i = P^N + c_i \Delta t Q^N + \Delta t^2 \sum\nolimits_{j=1}^{i-1} \bar{a}_{ij} R_j \tag{2.19a}$$

$$(M - \Delta t^2 \bar{a}_{ii} K) R_i = K \hat{P}_i + S_i \tag{2.19b}$$

$$P^{N+1} = P^N + \Delta t Q^N + \Delta t^2 \sum\nolimits_{i=1}^{m} \bar{b}_i R_i \tag{2.19c}$$

$$Q^{N+1} = Q^N + \Delta t \sum\nolimits_{i=1}^{m} b_i R_i. \tag{2.19d}$$

The variable $\hat{P}_i$ is introduced to improve readability. Note that it is not necessarily an approximation of P_i because the summand $\Delta t^2 \bar{a}_{ii} R_i$ is missing. This becomes clearer when comparing it to Equation 3.10a in the derivation of DIRK-2-0.

A wide range of RKN methods is available, although not as many as there are Runge-Kutta methods. Runge-Kutta coefficients can also be used with the transformation given in Equation 2.14. A unique reverse transformation also exists but does not necessarily lead to valid RK coefficients, since RKN generalizes the RK scheme. Furthermore, the transformations in both directions are not ideal because the conservation of a method's properties is not always given. Especially, stability classes usually have no equivalent for the other scheme. These problems are elaborated further in Section 4.2.

2.5 Solving the linear systems with the conjugate gradient method

Except for small domains, direct solvers are impractical for solving the arising linear systems. On a two-dimensional domain with $N = N_x \times N_y$ grid points with a second-order finite difference discretization, the system matrix would be $N \times N$ and sparse with $5N$ entries. The use of a direct solver would result in a fill-in, increasing the required storage from $O(5N)$ to $O(N^2)$. Furthermore, solvers like the Gaussian elimination would require $O(N^3)$ operations. Others like the LU or Cholesky factorization could complete the required $O(N^3)$ computations beforehand, but they would still need $O(N^2)$ operations for each occurring linear system.
Iterative Solvers on the other hand require only $O(N)$ additional storage. The number of operations per iteration is mostly determined by a multiplication involving the system matrix, which can be

achieved in $O(5N)$ in this example. The overall computational cost is highly dependent on the used solver and the number of iterations – and thus by the desired accuracy – but complexities below $O(N^2)$ are common.
The properties of our system determine which solvers are applicable. The matrix $(M - \Delta t^2 \bar{a}_{ii} K)$ is symmetric, because K is symmetric and M diagonal. K is weakly diagonally dominant, since

$$|k_{ii}| = \sum_{j \neq i} |k_{ij}| \tag{2.20}$$

for all finite difference stencils for second derivatives. The entries of M are the squared reciprocal of the wave velocities at the respective grid points and are therefore greater than zero. With the diagonal elements of $(-\Delta t^2 a_{ii}^2 K)$ being positive too, the system matrix is strictly diagonally dominant with positive diagonal elements. The Gershgorin circle theorem implies that such a system has only positive eigenvalues. Thus, the matrix is positive definite.
This enables the use of the conjugate gradient method as a solver.

2.5.1 Preconditioning with multigrid

While CG is efficient, it is not fast enough on its own to compete with explicit methods. Its complexity is $O(e\sqrt{\kappa})$, where e is the number of nonzero entries in the linear system matrix ($e \in O(N)$) and κ the condition number of the matrix, which is equal to the ratio of the largest to the smallest eigenvalue. While the number of entries e is determined by the domain size and finite difference method, the condition number κ can be reduced through preconditioning.
Since the matrix is symmetric positive definite, the incomplete Cholesky factorization is one option. Like some direct linear system solvers, it has the benefit that the factorization only needs to be computed once. But it is still expensive to apply at each time step and stage. Some preliminary experiments showed that this preconditioner reduces the iteration counts, but not far enough to get near the efficiency of explicit methods.
Multigrid is another popular option as a preconditioner for the conjugate gradient method, which is more effective for reducing the required preconditioned conjugate gradient method (PCG) iterations and has an optimal complexity of $O(N)$. Preconditioners for the PCG must be fixed linear symmetric positive definite operators to guarantee proper convergence behavior. This puts multiple constraints on the multigrid parameters. The symmetry requirement reduces the available options for the restriction and prolongation operator. Additionally, the pre-smoothing must be identical to the post-smoothing, especially the number of iterations. To be a fixed linear operator, all multigrid components must be linear operators too. These restrictions could be relaxed when switching to the flexible PCG method [BDK15], but we use the conventional method here.
We choose a simple V-cycle with a grid ratio of two. For pre- and post-smoothing, we use one damped Jacobi iteration with a dampening factor of $\frac{2}{3}$ each, which delivers good results with regard to accuracy and efficiency.
For the restriction operator R we use full weighting, which corresponds to the following stencil in 2D,

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2.21}$$

The prolongation operator p is chosen based on the transpose of the restriction operator, $P = 4R^T$, which corresponds to bilinear interpolation between the closest grid points. We will denote the linear system matrix as $A = (M - \Delta t^2 \bar{a}_{ii} K)$ and the next-coarser matrix with halved grid resolution as A^c. Ordinarily, the coarse matrix is chosen according to the Galerkin condition:

$$A^c = RAP. \tag{2.22}$$

But this transformation with the given grid transfer operators turns diagonal matrices and five-point stencils A into nine-point stencils A^c. To avoid the corresponding cost increase for matrix-vector multiplications, we consider alternative ways to coarsen the system matrix that do not increase the number of stencil points on coarser levels. The saved storage increases with higher-order finite differences and grid transfer operators because of larger fill-ins.
Since A is the weighted sum of the two matrices M and K the Galerkin condition can also be applied to just one of them. The matrix M is diagonal, while K contains the second order finite difference Laplace operator which is equivalent to the following five-point stencil in two dimensions with second-order finite differences:

$$\frac{1}{\Delta x^2} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.23}$$

For M, we use $M^c = RM$ as an alternative, which preserves the diagonal matrix structure. For K, we use the same stencil again on the coarser levels.
The different characteristics of the two matrices make a combination of using the Galerkin condition for K but not for M attractive, at least with regard to storage. Since M cannot be represented through a stencil for heterogeneous media, using the Galerkin condition for M would require storing the five-band matrix $M^c = RMP$. Using $M^c = RM$ instead requires less storage and still delivers averaged squared inverse wave speed values at the grid points. The stencil of $K^c = RKP$, on the other hand, can be determined analytically by tracing the points that influence one coordinate through the three operators. This is also possible for the operators on the coarser levels, like RK^cP. Therefore, one does not necessarily need to store the coarse versions of K. While this combination does not reduce the computational complexity by avoiding the fill-in, it reduces the required storage compared to using the Galerkin condition for the entire matrix A.

3 Alternative Runge-Kutta-based second-order time integration schemes

In this chapter, we derive two diagonally implicit Runge-Kutta-based second-order time integration schemes, which are similar to DIRKN but solve the Runge-Kutta equations for different variables. Furthermore, we use substitutions to create a version of one of the schemes for linear ODEs without matrix-vector multiplications.

The time integration schemes derived in this section are referred to as DIRK-2-1, DIRK-2-0, and DIRK-2-0s. Their name starts with DIRK because they are derived from diagonally implicit Runge-Kutta equations. The first number indicates that they are time integration methods for second-order ODEs. The second number denotes the derivative the equations are solved for. A method with a zero solves for the value itself, e.g. the acoustic pressure, one with a one for its first derivative, etc. Consequently, DIRKN could also be expressed as DIRK-2-2. One could theoretically generalize this approach for other orders of ODEs, but this is beyond the scope of this work. The appended 's' for the third scheme refers to the substitutions applied to it, which avoid matrix-vector multiplications for linear ODEs.

3.1 DIRK-2-1

The DIRK-2-1 scheme introduced in this section solves the diagonally implicit Runge-Kutta equations (Equation 2.9) for u_i. To that end, we substitute all stage variables r_j and x_j, $j = 1, \dots, i$ in Equation 2.9b for u_i. This means first replacing the r_j using Equation 2.9c and subsequently the newly introduced x_j using Equation 2.9a,

$$\begin{aligned} u_i &= u^N + \Delta t \sum_{j=1}^{i} a_{ij} r_j \\ &= u^N + \Delta t \sum_{j=1}^{i} a_{ij} f(t^N + c_j \Delta t, x_j) \\ &= u^N + \Delta t \sum_{j=1}^{i} a_{ij} f(t^N + c_j \Delta t, x^N + \Delta t \sum_{k=1}^{j} a_{jk} u_k). \end{aligned} \tag{3.1}$$

The r_i in the formula for u^{N+1} are replaced in a similar fashion, resulting in the scheme

$$u_i = u^N + \Delta t \sum_{j=1}^{i} a_{ij} f(t^N + c_j \Delta t, x^N + \Delta t \sum_{k=1}^{j} a_{jk} u_k) \tag{3.2a}$$

$$x^{N+1} = x^N + \Delta t \sum_{i=1}^{m} b_i u_i \tag{3.2b}$$

$$u^{N+1} = u^N + \Delta t \sum_{i=1}^{s} b_i f(t^N + c_i \Delta t, x^N + \Delta t \sum_{j=1}^{i} a_{ij} u_j). \tag{3.2c}$$

Because one sum is located inside the function call and the other one outside of it in each of the two equations for u_i and u^N, the equations cannot be simplified through the introduction of new coefficients, like $\bar{a}_{ij}$ and $\bar{b}_i$ for DIRKN (see Equation 2.14). Therefore, computing, storing, and using the variables x_i and r_i at all stages has no disadvantages except for requiring more storage. Using these values where possible instead of substituting them also decreases the number of required operations. The stored values are likely also useful for the derivation of initial guess strategies for the non-linear solver.

The situation is different for linear f, like in the case of the acoustic wave equation. The linearity allows the introduction of the same coefficients as for DIRKN. The scheme of Equation 3.2 applied to first-order form of the acoustic wave equation of Equation 2.6 results in:

$$Q_i = Q^N + \Delta t \sum_{j=1}^{i} a_{ij} M^{-1} \left(K \left(P^N + \Delta t \sum_{k=1}^{j} a_{jk} Q_k \right) + S(t^N + c_j \Delta t) \right) \tag{3.3a}$$

$$P^{N+1} = P^N + \Delta t \sum_{i=1}^{m} b_i Q_i \tag{3.3b}$$

$$Q^{N+1} = Q^N + \Delta t \sum_{i=1}^{m} b_i M^{-1} \left(K P_i + S(t^N + c_i \Delta t) \right). \tag{3.3c}$$

The first equation is reordered so that Q_i only appears on the left-hand side and is multiplied with M analogously to DIRKN. The last equation is also multiplied with M to reduce the required number of operations. The final DIRK-2-1 scheme for the acoustic wave equation with the DIRKN coefficients from Equation 2.14 is:

$$(M - \Delta t^2 \bar{a}_{ii} K) Q_i = M Q^N + K \left(\Delta t c_j P^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} Q_j \right) + \Delta t \sum_{j=1}^{i} a_{ij} S(t_j^N) \tag{3.4a}$$

$$P^{N+1} = P^N + \Delta t \sum_{i=1}^{m} b_i Q_i \tag{3.4b}$$

$$M Q^{N+1} = M Q^N + K \left(\Delta t P^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i Q_i \right) + \Delta t \sum_{i=1}^{m} b_i S(t_j^N). \tag{3.4c}$$

This scheme requires more operations than DIRKN and DIRK-2-0, as is shown later on in Table 4.1. Since it has no apparent advantages over the other schemes, we do not explore its properties further.

3.2 DIRK-2-0

The DIRK-2-0 scheme introduced in this section solves the diagonally implicit Runge-Kutta equations (Equation 2.9) for x_i. This approach was suggested by Longfei Gao [Gao19] (including the substitutions of DIRK-2-0s), not as a general time integration scheme but applied to the acoustic wave equation with the Runge-Kutta method RK7 from Table 5.1. His version stored the first derivative, which is not necessary and increases the overall computational costs. Recognition of

the similarity to DIRKN and especially the possibility of using the same coefficients are also new contributions.
To derive the general scheme, we again start with the diagonally implicit Runge-Kutta Equations 2.9. First, we substitute the u_j in Equation 2.9a for x_i using Equation 2.9b for u_i,

$$\begin{aligned} x_i &= x^N + \Delta t \sum_{j=1}^{i} a_{ij} u_j \\ &= x^N + \Delta t \sum_{j=1}^{i} a_{ij} \left(u^N + \Delta t \sum_{k=1}^{j} a_{jk} r_k \right). \end{aligned} \tag{3.5}$$

Additionally, we substitute the r_i on the right-hand side, but only for the current stage. Storing and reusing the r_j of previous stages result in a scheme that requires less operations per time step,

$$x_i = x^N + \Delta t \sum_{j=1}^{i} a_{ij} \left(u^N + \Delta t \sum_{k=1}^{\min\{j,i-1\}} a_{jk} r_k \right) + \Delta t^2 a_{ii}^2 f(t^N + c_i \Delta t, x_i). \tag{3.6}$$

The u_i in Equation 2.9d for are substituted similarly,

$$\begin{aligned} x^{N+1} &= x^N + \Delta t \sum_{i=1}^{m} b_i u_i \\ &= x^N + \Delta t \sum_{i=1}^{m} b_i \left(u^N + \Delta t \sum_{j=1}^{i} a_{ij} r_j \right). \end{aligned} \tag{3.7}$$

These equations can be simplified by introducing the same coefficients as for the Nyström methods,

$$\bar{a}_{ij} = \sum_{k=1}^{i} a_{ik} a_{kj} \quad \text{and} \quad \bar{b}_i = \sum_{j=1}^{i} b_j a_{ji} \tag{3.8}$$

and using the properties of Runge-Kutta methods

$$\sum_{j=1}^{m} a_{ij} = c_i \quad \text{and} \quad \sum_{i=1}^{m} b_i = 1. \tag{3.9}$$

The resulting scheme is

$$x_i = x^N + \Delta t c_i u^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} r_j + \Delta t^2 \bar{a}_{ii} f(t^N + c_i \Delta t, x_i) \tag{3.10a}$$

$$r_i = f(t^N + c_i \Delta t, x_i) \tag{3.10b}$$

$$x^{N+1} = x^N + \Delta t u^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i r_i \tag{3.10c}$$

$$u^{N+1} = u^N + \Delta t \sum_{i=1}^{m} b_i r_i, \tag{3.10d}$$

The first equation must be solved for x_i at each stage.

We apply it to the acoustic wave equation in the first-order form of Equation 2.7. Note that the different splitting of the equation compared to DIRKN and DIRK-2-1 causes Q and R to refer to different values,

$$MQ_{\text{DIRKN}} = Q_{\text{DIRK-2-0(s)}} \quad \text{and} \quad MR_{\text{DIRKN}} = R_{\text{DIRK-2-0(s)}}. \tag{3.11}$$

After reordering the first equation so P_i only occurs on the left-hand side, the scheme becomes:

$$(M - \Delta t^2 \bar{a}_{ii} K) P_i = MP^N + \Delta t c_i Q^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} R_j + \Delta t^2 \bar{a}_{ii} S_i \tag{3.12a}$$

$$R_i = KP_i + S_i \tag{3.12b}$$

$$MP^{N+1} = MP^N + \Delta t Q^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i R_i \tag{3.12c}$$

$$Q^{N+1} = Q^N + \Delta t \sum_{i=1}^{m} b_i R_i. \tag{3.12d}$$

By itself, this scheme does not have apparent advantages over DIRKN. The potential improvements in computational efficiency of the method are the result of further substitutions.

3.3 Substitutions

3.3.1 DIRK-2-0 with substitutions: DIRK-2-0s

In order to reduce the number of required operations, Longfei Gao [Gao19] suggested further substitutions in the method he developed based on the RK7 method (see Table 5.1). The substitutions remove the need for the matrix-vector multiplications KP_i, which occur in the equations used to compute the R_i. This approach can be generalized to all DIRK-2-0 methods for linear ODEs. Additionally, we introduce simplifications to reduce the computational cost further. The resulting scheme is called DIRK-2-0s. We use DIRK-2-0(s) to refer to both schemes, with and without substitutions.
The starting points for the derivation of the substitutions consists of the diagonally implicit Runge-Kutta equations (see Equation 2.9) applied to the acoustic wave equation in the first-order form of Equation 2.7,

$$MP_i = MP^N + \Delta t \sum_{j=1}^{i} a_{ij} Q_j \tag{3.13a}$$

$$Q_i = Q^N + \Delta t \sum_{j=1}^{i} a_{ij} R_j \tag{3.13b}$$

$$R_i = KP_i + S_i \tag{3.13c}$$

$$MP^{N+1} = MP^N + \Delta t \sum_{i=1}^{m} b_i Q_i \tag{3.13d}$$

$$Q^{N+1} = Q^N + \Delta t \sum_{i=1}^{m} b_i R_i. \tag{3.13e}$$

Reordering the Runge-Kutta equation for Q_i yields an alternative, matrix-multiplication-free way of computing R_i,

$$Q_i = Q^N + \Delta t \sum_{j=1}^{i} a_{ij} R_j \quad \Leftrightarrow \tag{3.14}$$

$$R_i = -\frac{1}{\Delta t a_{ii}} \left(Q^N - Q_i \right) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} R_j. \tag{3.15}$$

But this necessitates a new way to calculate Q_i that is independent of R_i. An alternative expression can be obtained by reordering the original equation for MP_i,

$$MP_i = MP^N + \Delta t \sum_{j=1}^{i} a_{ij} Q_j \quad \Leftrightarrow \tag{3.16}$$

$$Q_i = -\frac{1}{\Delta t a_{ii}}(MP^N - MP_i) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} Q_j. \tag{3.17}$$

Since it is computationally more efficient to not store the Q_i, we first substitute the Q_j here with the original Runge-Kutta Equation 3.13b for Q_i.

$$\begin{aligned} Q_i &= -\frac{1}{\Delta t a_{ii}}(MP^N - MP_i) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} \left(Q^N + \Delta t \sum_{k=1}^{j} a_{jk} R_k \right) \\ &= -\frac{1}{\Delta t a_{ii}}(MP^N - MP_i) - \frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} \right) Q^N - \frac{\Delta t}{a_{ii}} \sum_{k=1}^{i-1} \left(\sum_{j=k}^{i-1} a_{ij} a_{jk} \right) R_k \\ &= -\frac{1}{\Delta t a_{ii}}(MP^N - MP_i) - \frac{c_i - a_{ii}}{a_{ii}} Q^N - \frac{\Delta t}{a_{ii}} \sum_{j=1}^{i-1} \tilde{a}_{ij} R_j. \end{aligned} \tag{3.18}$$

with the new coefficients $\tilde{a}_{ij}$,

$$\tilde{a}_{ij} = \sum_{k=j}^{i-1} a_{ik} a_{kj} = \sum_{k=1}^{i-1} a_{ik} a_{kj}. \tag{3.19}$$

We use the derived formula for Q_i to substitute it in Equation 3.15 for R_i,

$$\begin{aligned} R_i &= -\frac{1}{\Delta t a_{ii}} \left(Q^N - \left(-\frac{1}{\Delta t a_{ii}}(MP^N - MP_i) - \frac{c_i - a_{ii}}{a_{ii}} Q^N - \frac{\Delta t}{a_{ii}} \sum_{j=1}^{i-1} \tilde{a}_{ij} R_j \right) \right) - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} R_j \\ &= -\frac{1}{\Delta t^2 a_{ii}^2}(MP^N - MP_i) - \frac{c_i}{\Delta t a_{ii}^2} Q^N - \sum_{j=1}^{i-1} \left(\frac{\tilde{a}_{ij}}{a_{ii}^2} + \frac{a_{ij}}{a_{ii}} \right) R_j. \end{aligned} \tag{3.20}$$

This alternative way of computing R_i in DIRK-2-0 leads to the matrix-multiplication-free DIRK-2-0s scheme,

$$(M - \Delta t^2 a_{ii}^2 K) P_i = MP^N + \Delta t c_i Q^N + \Delta t^2 \textstyle\sum_{j=1}^{i-1} \bar{a}_{ij} R_j + \Delta t^2 a_{ii}^2 S_i \tag{3.21a}$$

$$R_i = -\frac{1}{\Delta t^2 a_{ii}^2}(MP^N - MP_i) - \frac{c_i}{\Delta t a_{ii}^2} Q^N - \textstyle\sum_{j=1}^{i-1} \left(\frac{\tilde{a}_{ij}}{a_{ii}^2} + \frac{a_{ij}}{a_{ii}} \right) R_j \tag{3.21b}$$

$$MP^{N+1} = MP^N + \Delta t Q^N + \Delta t^2 \textstyle\sum_{i=1}^{m} \bar{b}_i R_i \tag{3.21c}$$

$$Q^{N+1} = Q^N + \Delta t \textstyle\sum_{i=1}^{m} b_i R_i. \tag{3.21d}$$

Note that similar substitutions are not possible for explicit stages like they occur in ESDIRK methods, for example. This means that explicit stages in a DIRK-2-0s method always lead to matrix-vector multiplications. But these stages only require one such operation. In contrast, iterative linear system solvers for implicit DIRK-2-0 stages usually perform many matrix-vector multiplications.

3.3.2 Substitutions for DIRKN and DIRK-2-1

For the DIRKN and DIRK-2-1, avoiding the matrix multiplication by substitution is not feasible. The vector that is multiplied with the matrix K consists of a more complex sum of different vectors in these schemes. For these multiplications, no alternative expression can be derived from the Runge-Kutta equations.
Comparable substitutions are also not possible for ERKN because R_i occurs only once in the corresponding equations.

4 Theoretical comparison of the time integration schemes

In this chapter, the previously introduced integration schemes are compared. This also includes the issues of RKN-RK coefficient conversion and amplification of the linear system solver's error. The focus lies on complexity analysis to approximate and compare the costs for the integration schemes, conjugate gradient method, and multigrid preconditioner. This allows a theoretical prediction of when implicit methods could outperform an explicit alternative with the presented approach. Furthermore, different strategies for the initial guesses of the conjugate gradient method are introduced.
The conclusions in this chapter are derived from the equations, algorithms, and exemplary values without reliance on experimental data.

4.1 Summary of the time integration schemes

In this section, we collocate the previously derived equations of the time integration schemes DIRKN, DIRK-2-1, DIRK-2-0, and DIRK-2-0s applied to the acoustic wave equation in order to simplify their comparison. The explicit Runge-Kutta-Nyström scheme (ERKN) is also included. It can be regarded as a DIRKN scheme where the diagonal elements in the Butcher tableau are zero. We use it as a reference for the implicit schemes to compare against. Its time step sizes are limited by the CFL condition.

To improve the readability of the equations and help the comparison, we introduce new variables that simplify the equations but would necessarily be explicitly computed in efficient implementations. These are indicated by a hat index ($\wedge$).
Understanding the following notations and relationships between variables also helps when comparing the schemes. The source term S_i is equal to the value of the source signal at the time associated with its stage $S_i = S(t^N + c_i \Delta t), \quad \forall i = 1, ..., m$.
For the implicit methods, the diagonal entries in the Butcher tableau for $\bar{a}$ are equal to the squared diagonal elements of a, and thus used interchangeably, $\bar{a}_{ii} = a_{ii}^2, \ \forall i = 1, ..., m$. We are especially interested in methods with identical diagonal elements, $\bar{a}_{ii} = \bar{a}_{jj}$ and $a_{ii} = a_{jj}, \ \forall i, j = 1, ..., m$, since this causes the linear systems to be identical at all stages.
The definition of the variables Q and R at the stages or time steps varies between the integration schemes because they are either based on the first-order form of Equation 2.6 or 2.7. The variables either contain the matrix M, which consists of the squared inverses of the wave speeds, or not,

$$Q_{\text{DIRK-2-0}} \mathrel{\hat{=}} Q_{\text{DIRK-2-0s}} \mathrel{\hat{=}} MQ_{\text{ERKN}} \mathrel{\hat{=}} MQ_{\text{DIRKN}} \mathrel{\hat{=}} MQ_{\text{DIRK-2-1}} \tag{4.1}$$

$$R_{\text{DIRK-2-0}} \mathrel{\hat{=}} R_{\text{DIRK-2-0s}} \mathrel{\hat{=}} MR_{\text{ERKN}} \mathrel{\hat{=}} MR_{\text{DIRKN}} \mathrel{\hat{=}} MR_{\text{DIRK-2-1}}. \tag{4.2}$$

ERKN:

$$
\begin{aligned}
P_i &= P^N + c_i \Delta t Q^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} R_j && \text{(4.3a)}\\
R_i &= M^{-1} K P_i + M^{-1} S_i && \text{(4.3b)}\\
P^{N+1} &= P^N + \Delta t Q^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i R_i && \text{(4.3c)}\\
Q^{N+1} &= Q^N + \Delta t \sum_{i=1}^{m} b_i R_i && \text{(4.3d)}
\end{aligned}
$$

DIRKN:

$$
\begin{aligned}
\hat{P}_i &= P^N + c_i \Delta t Q^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} R_j && \text{(4.4a)}\\
(M - \Delta t^2 \bar{a}_{ii} K) R_i &= K \hat{P}_i + S_i && \text{(4.4b)}\\
P^{N+1} &= P^N + \Delta t Q^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i R_i && \text{(4.4c)}\\
Q^{N+1} &= Q^N + \Delta t \sum_{i=1}^{m} b_i R_i && \text{(4.4d)}
\end{aligned}
$$

DIRK-2-0:

$$
\begin{aligned}
(MP)_i^{\wedge} &= MP^N + c_i \Delta t Q^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} R_j && \text{(4.5a)}\\
(M - \Delta t^2 \bar{a}_{ii} K) P_i &= (MP)_i^{\wedge} + \Delta t^2 \bar{a}_{ii} S_i && \text{(4.5b)}\\
R_i &= K P_i + S_i && \text{(4.5c)}\\
MP^{N+1} &= MP^N + \Delta t Q^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i R_i && \text{(4.5d)}\\
Q^{N+1} &= Q^N + \Delta t \sum_{i=1}^{m} b_i R_i && \text{(4.5e)}
\end{aligned}
$$

DIRK-2-0s:

$$
\begin{aligned}
(MP)_i^{\wedge} &= MP^N + c_i \Delta t Q^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} R_j && \text{(4.6a)}\\
(M - \Delta t^2 \bar{a}_{ii} K) P_i &= (MP)_i^{\wedge} + \Delta t^2 \bar{a}_{ii} S_i && \text{(4.6b)}\\
R_i &= -\frac{1}{\Delta t^2 \bar{a}_{ii}} (MP^N - MP_i) - \frac{c_i}{\Delta t \bar{a}_{ii}} Q^N - \sum_{j=1}^{i-1} \left(\frac{\tilde{a}_{ij}}{\bar{a}_{ii}} + \frac{a_{ij}}{a_{ii}} \right) R_j && \text{(4.6c)}\\
MP^{N+1} &= MP^N + \Delta t Q^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i R_i && \text{(4.6d)}\\
Q^{N+1} &= Q^N + \Delta t \sum_{i=1}^{m} b_i R_i && \text{(4.6e)}
\end{aligned}
$$

DIRK-2-1

$$\Delta t\hat{P}_i = \Delta t c_j P^N + \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij} Q_j \tag{4.7a}$$

$$(M - \Delta t^2 \bar{a}_{ii} K) Q_i = MQ^N + K\Delta t\hat{P}_i + \Delta t \sum_{j=1}^{i} a_{ij} S(t_j^N) \tag{4.7b}$$

$$P^{N+1} = P^N + \Delta t \sum_{i=1}^{m} b_i Q_i \tag{4.7c}$$

$$\Delta t\hat{P}^{N+1} = \Delta t P^N + \Delta t^2 \sum_{i=1}^{m} \bar{b}_i Q_i \tag{4.7d}$$

$$MQ^{N+1} = MQ^N + K\Delta t\hat{P}^{N+1} + \Delta t \sum_{i=1}^{m} b_i S(t_j^N). \tag{4.7e}$$

4.2 Coefficients

While there is a multitude of diagonally implicit Runge-Kutta methods available, the subset fulfilling additional properties like unconditional stability and identical diagonal elements is relatively small. The number of DIRKN methods with such properties is even smaller because of their lesser popularity and more limited applicability.
It is possible to transform Runge-Kutta into Nyström coefficients, as we have seen in the DIRKN derivation (see Equation 2.14). But these indirect RKN methods do not use all degrees of freedom that Nyström integration schemes offer, which makes them potentially inferior to directly derived methods. Furthermore, it is not always obvious how properties of specific methods like stability translate from first-order to second-order ODEs.

An opposite problem exists with the new integration schemes. Since DIRK-2-0 and DIRK-2-0s are derived from Runge-Kutta equations and their own order conditions have not been derived yet, only the use of transformed Runge-Kutta coefficients is theoretically sound.
Despite this, we are interested in the performance of DIRKN coefficients with the DIRK-2-0(s) schemes since this would increase the number of available schemes, especially those with second-order ODE stability properties. An additional problem here is that DIRK-2-0s relies not only on the DIRKN coefficients $\bar{a}_{ij}, b_i, \bar{b}_i$ and c_i, but also on the DIRK coefficients a_{ij}, which are not part of DIRKN methods. The missing coefficients a_{ij} can be obtained by inverting the DIRK-to-DIRKN coefficient conversion (Equation 2.14) for SDIRKN methods,

$$b^{\mathrm{RK}} = b^{\mathrm{RKN}} \tag{4.8a}$$

$$c^{\mathrm{RK}} = c^{\mathrm{RKN}} \tag{4.8b}$$

$$a_{ii}^{\mathrm{RK}} = \sqrt{\bar{a}_{ii}^{\mathrm{RKN}}} \quad \forall i = 1, \ldots, m \tag{4.8c}$$

$$a_{ij}^{\mathrm{RK}} = \frac{1}{2a_{ii}^{\mathrm{RK}}} \left(\bar{a}_{ij}^{\mathrm{RKN}} - \sum_{k=j+1}^{i-1} a_{ik}^{\mathrm{RK}} a_{kj}^{\mathrm{RK}} \right) \quad \forall i, j = 1, \ldots, m. \tag{4.8d}$$

Because of the recursive nature of the last equation, the Runge-Kutta coefficients a_{ij}^{RK} have to be computed in an order that ensures that all required values are available. This order consists in computing the lower diagonals successively starting next to the main diagonal, each top to bottom (increasing i).
But this conversion only has a theoretical foundation for indirect SDIRKN methods. For direct

DIRKN methods, it still yields coefficients, but these do usually not constitute a valid Runge-Kutta method. Nevertheless, our numerical experiments (see Section 5.2) indicate that this is not a problem since the DIRK-2-0(s) schemes produce the same results as DIRKN with these coefficients if the linear systems are solved exactly. The fact that using the converted coefficients works is surprising because the transformation from DIRKN to DIRK-2-0 or DIRK-2-0s relies on equations from the underlying RK scheme. These results suggest that the order conditions, as well as the conditions for other properties like stability, for DIRK-2-0(s) might be the same as for DIRKN. But attempting to prove this or deriving their own conditions would be beyond the scope of this work.

4.3 Error analysis

Since DIRKN, DIRK-2-0, and DIRK-2-0s are derived from the same Runge-Kutta equations, they produce the same results when there are no error sources in the computations (see Section 5.2 for numerical validation). But this is not the case when an imprecise linear system solver like the conjugate gradient method is used. Where a linear system is solved in a scheme, an error ε_{CG} is introduced, which is controlled by the CG tolerance. This error impacts the time step results differently in the different schemes. The numerical experiments in Section 5.3 show that this affects the overall error. For higher CG tolerances, DIRK-2-0s amplifies the error significantly compared to DIRKN. The error amplification of DIRK-2-0 compared to DIRKN is small enough to not be visible in most of the shown numerical results.
The equations of the schemes provide some evidence for why this amplification varies between them. The way the values R_i are used in the equations that deliver the time step results is the same for all three schemes. Only the equations for the stages, and therefore for the computation of R_i, vary. For the following analysis, we assume that the previous time step results P^n and Q^n are exact, while the results of the previous stages, $R_j,\ j = 1, \ldots, i-1$, are not. We denote the variables being influenced by the CG error with an ε index, e.g. R_j^ε.

Each scheme solves one linear system with some right-hand side RHS per stage. These are influenced by the errors contained in the previous stages. For DIRK, the error in the right-hand side at stage i is:

$$\left(\text{RHS}_i^\varepsilon - \text{RHS}_i\right)_{\text{DIRKN}} = \Delta t^2 K \sum_{j=1}^{i-1} \bar{a}_{ij}(R_j^\varepsilon - R_j). \tag{4.9}$$

For DIRK-2-0 and DIRK-2-0s, it is:

$$\left(\text{RHS}_i^\varepsilon - \text{RHS}_i\right)_{\text{DIRK-2-0(s)}} = \Delta t^2 \sum_{j=1}^{i-1} \bar{a}_{ij}(R_j^\varepsilon - R_j). \tag{4.10}$$

The errors both contain the factor Δt^2 which should reduce the impact of the right-hand side error. Therefore, we neglect the difference between the two, which consists of the presence of the discrete Laplace operator K in the equation for DIRKN. How these errors affect the result of the linear systems is not simple to determine. Since the right-hand side errors likely do not contribute much to the overall errors and the error differences between the schemes, we include it in the overall CG error ε^{CG}, which also depends on the used tolerance.

For DIRKN, the value of R_i is then only affected by the CG error ε^{CG},

$$\left(R_i^\varepsilon - R_i\right)_{\text{DIRKN}} = \varepsilon^{\text{CG}}. \tag{4.11}$$

For DIRK-2-0, it is determined by the discrete Laplacian of the CG error,

$$\left(R_i^{\varepsilon} - R_i\right)_{\text{DIRK-2-0}} = K\varepsilon^{\text{CG}}. \tag{4.12}$$

Similarly to above, the only difference between the two consists of the presence of the discrete Laplace operator K but in the opposite equation and without an additional factor. The Δx contained in K as the factor $\frac{1}{\Delta x^2}$ is usually relatively large ($\Delta x > 1m$) and therefore does not have an amplifying effect. The other components of K still have the potential to amplify the error value because CG errors are usually not smooth. Therefore, the sum of its second derivatives in each dimension (Laplacian) could be large compared to its absolute value. A limit can be given as the sum over all absolute weights in the finite-difference stencil times the largest expected error value $\varepsilon_{\text{CG}}^{\max}$. For the two-dimensional second-order finite differences this limit is $\frac{8}{\Delta x^2}\varepsilon_{\text{CG}}^{\max}$, which does not indicate a particularly strong error amplification. For large enough Δx the matrix could even reduce the errors. Note that this limit is proportional to the number of stencil points and therefore increases with higher orders or dimensions.
In theory, the multigrid preconditioner could additionally smooth the high-frequency error components, which could reduce the errors further. But the results of Section 5.3 do not show a large difference between using and not using the preconditioner. The similar error levels of DIRKN and DIRK-2-0 confirm that the application of the matrix K does not have a large error amplifying effect. The potential of additional smoothing iterations applied to the MGPCG result for error reduction could be investigated, but are beyond the scope of this work.

The error for the stage result for DIRK-2-0s differs from the one for DIRK-2-0 because of the substitution of the discrete Laplace operator and assumes the following form:

$$\left(R_i^{\varepsilon} - R_i\right)_{\text{DIRK-2-0s}} = \frac{1}{\Delta t^2 \bar{a}_{ii}} M\varepsilon^{\text{CG}} - \sum\nolimits_{j=1}^{i-1} \left(\frac{\tilde{a}_{ij}}{\bar{a}_{ii}} + \frac{a_{ij}}{a_{ii}}\right)\left(R_j^{\varepsilon} - R_j\right). \tag{4.13}$$

In order to compare DIRK-2-0 and DIRK-2-0s here, we have to first compare the effect of K and $\left(\frac{1}{\Delta t^2 a_{ii}^2} M\right)$ on the error. Based on the reasoning introduced in Section 4.4.4 (Equations 4.30, 4.28a and 4.29a), we can infer that the value of $\frac{1}{\Delta x^2}$ contained as a factor in K lies in the interval

$$\frac{1}{\Delta x^2} \in \left[\frac{{c_{\text{CFL}}}^2}{v_{\max}^2 \Delta t^2}, \frac{n_{\text{dim}}}{v_{\min}^2 \Delta t^2}\right]. \tag{4.14}$$

Since the diagonal matrix M contains the squared inverse wave speeds at the grid points, its multiplication is comparable to a multiplication with some value in the interval $\left[\frac{1}{v_{\max}^2}, \frac{1}{v_{\min}^2}\right]$. Figure 5.11 that the value of a_{ii}^2 is contained in the interval $a_{ii}^2 \in [0.03, 1.7]$ for many methods. Therefore, the range of $\left(\frac{1}{\Delta t^2 a_{ii}^2} M\right)$ can be estimated as

$$\left(\frac{1}{\Delta t^2 a_{ii}^2} M\right) \in \left[\frac{0.5882}{v_{\max}^2 \Delta t^2}, \frac{33.3334}{v_{\min}^2 \Delta t^2}\right]. \tag{4.15}$$

While these are rough estimates, they still back up the assumption that $\left(\frac{1}{\Delta t^2 a_{ii}^2} M\right)$ and $\frac{1}{\Delta x^2}$ are of the same order of magnitude. When neglecting the impact of the weighted sum of the neighboring points contained in K and only considering the factor $\frac{1}{\Delta x^2}$, we can further claim that the multiplication

application of K to the CG error for DIRK-2-0 has a similar effect as the multiplication of the CG error with $\left(\frac{1}{\Delta t^2 a_{ii}^2} M\right)$ for DIRK-2-0s. This leads to the assumption that the main difference regarding error amplification between DIRK-2-0 and DIRK-2-0s stems from the sum of the errors from the previous stages $\left(R_j^{\varepsilon} - R_j\right)$. The corresponding coefficients $\left(\frac{\tilde{a}_{ij}}{a_{ii}^2} + \frac{a_{ij}}{a_{ii}}\right)$ vary in value but do not change the order of magnitude of the errors' contributions. Therefore, this additional term can potentially cause large error accumulations, especially for schemes with many stages.

4.4 Complexity analysis

In order to make informed decisions regarding the time integration schemes and linear system solver, a quantification of their computational cost is necessary.

4.4.1 Time integration scheme costs

Table 4.1: Operation counts for m-stage schemes. The number of finite difference stencil points is denoted with n_{sp}.

Operations	ERKN	DIRKN	DIRK-2-0	DIRK-2-0s	DIRK-2-1
Linear system solves	0	m	m	m	m
Matrix-vector multiplications	m	m	m	0	$m+1$
Vector sums, subtractions or element-wise multiplications	$0.5m^2 + 3.5m$	$0.5m^2 + 2.5m + 1$	$0.5m^2 + 2.5m + 1$	$1m^2+5m+1$	$1m^2+2m+1$
Scalar-vector multiplications	$0.5m^2 + 2.5m - 1$	$0.5m^2 + 2.5m$	$0.5m^2 + 2.5m$	$1m^2 + 4m$	$1m^2 + 2m$
Total vector operations, M-V-mult. in $2n_{\text{sp}}$ V-op. (w/o LS solver)	$1m^2 + (6 + 2n_{\text{sp}})m - 1$	$1m^2 + (5 + 2n_{\text{sp}})m + 1$	$1m^2 + (5 + 2n_{\text{sp}})m + 1$	$2m^2+9m+2$	$2m^2 + (4 + 2n_{\text{sp}})m + 1 + 2n_{\text{sp}}$
... with $m = 3, n_{\text{sp}} = 5$	56	55	55	47	71
... with $m = 3, n_{\text{sp}} = 9$	80	79	79	47	103
... with $m = 5, n_{\text{sp}} = 5$	104	101	101	97	131
... with $m = 5, n_{\text{sp}} = 9$	144	141	141	97	179

Table 4.1 shows the numbers of different operations that each integration scheme requires for each time step, which depends on the used method's number of stages m. The variable n_{sp} denotes the number of finite difference stencil points and is equal to the number of spatial dimensions n_{dim} times the FD order n_{order} plus one for the standard finite difference method,

$$n_{\text{sp}} = n_{\text{dim}} n_{\text{order}} + 1. \tag{4.16}$$

For these numbers, the following assumptions are used:
The acoustic pressure P^N of each grid point at every iteration is not computed by DIRK-2-0 and DIRK-2-0s. This saves one element-wise vector multiplication that would be used to determine it, $P^N = M^{-1}(MP^N)$. Similarly, Q^N is not computed from MQ^N in DIRK-2-1.
Furthermore, we assume that the source is local, e.g. in a single point, or only active for a short time, which allows us to neglect the cost of all source terms. Compared to a source that is permanently active on the whole domain, this saves between m and $4m$ vector operations, depending on the scheme.
Additionally, $\Delta t Q^N$ is stored instead of Q^N, which reduces the number of required scalar-vector operations by one for all schemes. An exception is DIRK-2-1, which does not require $\Delta t Q^N$. But the same savings can be achieved for it by storing $\Delta t P^N$ instead of P^N.

Should the used diagonally implicit Runge-Kutta coefficients include an explicit stage, the costs differ. Solving a linear system is not required in such a stage. Additionally, DIRKN and DIRK-2-0 require one vector operation less if the first stage is explicit (with $c_1 = 0$), and one vector operation more for other explicit stages. If stage s is explicit, the vector operation count of DIRK-2-0s changes by $(-2s - 2 + 2n_{\text{sp}})$. The term $2n_{\text{sp}}$ is introduced here because the matrix-vector multiplication of the explicit stage cannot be avoided.

Table 4.1 also contains approximations of the overall costs excluding the linear system solver costs. These costs are more complex to determine and are handled in the next section. To arrive at an overall cost approximation the different operations must be weighted according to their individual costs. We assume the cost of vector-vector and scalar-vector operations of the vector with N entries to be the same at N floating point operations (FLOPS). For the purpose of our analysis, we consider N FLOPS to be equivalent to one 'vector operation'. All occurring vectors have N entries, which corresponds to the number of grid points. The matrix-vector multiplications always involve the sparse $N \times N$ finite difference matrix K. The number of entries in this matrix per row is equal to the number of finite-difference stencil points n_{sp}. We treat one such multiplication as $2n_{\text{sp}}$ vector operations or $2n_{\text{sp}}N$ FLOPS. The formulas for the total number of vector operations required by each scheme can be found in the table as functions of m and n_{sp}. Since the schemes are difficult to compare based on the formulas alone, the table also contains values for exemplary parameters. These are 3 and 5 stages for 5 and 9 stencil points, which correspond to second- and fourth-order finite differences in two dimensions.

The table shows that DIRK-2-1 is more expensive to use than the alternatives. DIRKN and DIRK-2-0 require an identical number of operations. Without the linear system solver cost, DIRKN also has a similar cost as ERKN per time step. For two-stage methods, they even share the same cost here, independent of the finite differences used. But ERKN usually requires smaller time steps than the implicit schemes and DIRKN the additional use of a linear system solver. DIRK-2-0s is the only scheme whose costs without the linear system solver do not explicitly depend on the number of stencil points. This is because the substitutions eliminated the matrix-vector multiplication. Note that the quadratic term is weighted twice as much compared to DIRKN and DIRK-2-0. DIRK-2-0s thus becomes less expensive relative to the other two if the number of stages decreases or the number of stencil points increases.

The schemes' costs without the contribution of the linear system solvers are visualized in Figure 4.1 to illustrate their relative differences. They are plotted as planes over the two parameters which are the number of stages m and the number of stencil points n_{sp}.

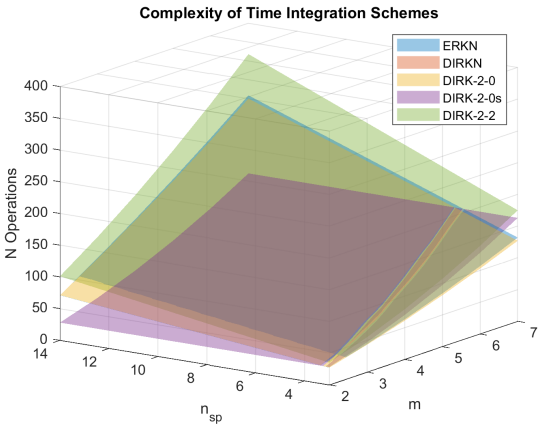

Figure 4.1: FLOP costs per time step of the different time integration schemes over the number of stages m and FD stencil points n_{sp} based on the values from Table 4.1. Linear system solver costs are not included. Note that because of the identical costs of DIRKN and DIRK-2-0 only one of the planes is visible.

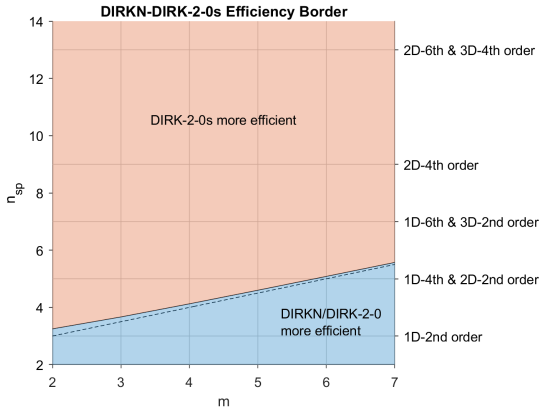

Figure 4.2: Comparison of DIRKN and DIRK-2-0s complexity assuming equal accuracy and linear system solver cost based on the values from Table 4.1. Examples for standard finite difference stencils are given on the right for their corresponding number of stencil points. The dashed line is the approximation solid line with $n_{sp}(m) \approx \frac{m}{2} + 2$.

The two identical planes corresponding to DIRKN and DIRK-2-0 are close to parallel to the one of ERKN, with the differences increasing slightly for larger m and n_{sp}. They intersect for two-stage methods with ERKN being slightly more expensive for more stages. Note that equal time step size and accuracy are assumed here and the linear system solver costs are excluded. For a total cost comparison see Section 4.4.4.
The plane of DIRK-2-1 is of similar shape but increases more rapidly with any of the two parameters increasing. It does not intersect the planes of any other methods, making it the most expensive option for all realistic parameter values.
The cost of DIRK-2-0s is independent of n_{sp}, which causes its plane to intersects with the ones of ERKN, DIRKN, and DIRK-2-0. This means that whether DIRKN and DIRK-2-0 or DIRK-2-0s are more efficient depends on the two parameters.

Figure 4.2 shows which scheme is more efficient for which parameter configuration in more detail. The border in the parameter space is derived by comparing their costs. DIRKN or DIRK-2-0 is more efficient than DIRK-2-0s if

$$1m^2 + (5 + 2n_{sp})m + 1 < 2m^2 + 9m + 2 \tag{4.17}$$

$$\Leftrightarrow \quad n_{sp} < \frac{1}{2m}\left(m^2 + 4m + 1\right). \tag{4.18}$$

This border can be approximated with $n_{sp}(m) = \frac{1}{2m}\left(m^2 + 4m + 1\right) \approx \frac{m}{2} + 2$, which is included in the diagram as the dashed line.
It should be noted here that this complexity analysis is only an approximation. While the numbers

are of relatively high accuracy, the actual cost and border can vary and also depends on the implementation.
What this analysis shows is that there are realistic stencil point and stage numbers where their costs are similar. In most cases, DIRK-2-0s seems to be the better choice, according to the metric we use here. Its advantage over DIRKN increases for more dimensions, higher finite-difference orders, and fewer stages.

This analysis neglects the impact of the schemes on accuracy and linear system solver cost. These factors can change the results significantly. A discussion of the efficiency of the schemes that include them can be found in the following chapter.

4.4.2 Linear system solver cost

The multigrid preconditioned conjugate gradient method (MGPCG) is used as a solver. The implementation used in the experiments requires the following operations with $n_{\text{it-CG}}$ being the number of PCG iterations:

- $(n_{\text{it-CG}} + 1)$ matrix-vector multiplications,
- $3n_{\text{it-CG}}$ vector-vector operations,
- $3n_{\text{it-CG}}$ scalar-vector multiplications,
- $(n_{\text{it-CG}} + 2)$ L_2-norms,
- $2n_{\text{it-CG}}$ dot products, and
- $n_{\text{it-CG}}$ preconditioner runs.

The matrix-vector multiplications involve the finite difference matrix again and are therefore equivalent to $2n_{\text{sp}}$ vector operations. The norms and dot products are treated as two vector operations.
The total number of vector operations required by the unpreconditioned conjugate gradient method, therefore, amounts to

$$C_{\text{CG}}^{\text{1it}} = \left(2n_{\text{sp}} + 12\right) n_{\text{it-CG}} + 2n_{\text{sp}} + 4. \tag{4.19}$$

To assess the computational complexity of the preconditioner, we first only consider the operations on the finest grid. On every level multigrid requires

- n_{sm} smoothing iterations, which consist of pre- and post-smoothing iterations $n_{\text{sm}} = n_{\text{pre-sm}} + n_{\text{post-sm}}$,
- 1 matrix-vector multiplication,
- 2 vector-vector operations,
- 1 restriction, and
- 1 prolongation.

We assume the matrix multiplication to be equivalent to $2n_{\text{sp}}$ vector operations, which corresponds to the Galerkin condition not being used. The restriction and prolongation operators, which are based on full weighting, require $2\left(\frac{n_{\text{order}}+1}{2}\right)^{n_{\text{dim}}}$ vector operations each. When using the damped Jacobi method as a smoother, one smoothing iteration costs

- 1 matrix multiplication, which is again equivalent to $2n_{\text{sp}}$ vector operations,

- 3 vector-vector, and
- 1 scalar-vector operations.

Therefore, the total number of vector operations of one multigrid preconditioner iteration on the finest level amounts to

$$C_{\text{MG-L0}} \approx \left(n_{\text{sm}} \left(2n_{\text{sp}} + 4 \right) + 4 \left(\frac{n_{\text{order}} + 1}{2} \right)^{n_{\text{dim}}} + 2n_{\text{sp}} + 2 \right). \tag{4.20}$$

Considering that the algorithm does roughly the same on all grid levels with $\frac{1}{2^{n_{\text{dim}}}}$-th the problem size, we approximate the overall cost of applying the preconditioner as

$$\begin{aligned} C_{\text{MG}} &= \sum_{l=0}^{\infty} \left(\frac{1}{2^{n_{\text{dim}}}} \right)^{l} C_{\text{MG-L0}} = \frac{2^{n_{\text{dim}}}}{2^{n_{\text{dim}}} - 1} C_{\text{MG-L0}} \\ &= \frac{2^{n_{\text{dim}}}}{2^{n_{\text{dim}}} - 1} \left(n_{\text{sm}} \left(2n_{\text{sp}} + 4 \right) + \left(\frac{n_{\text{order}} + 1}{2} \right)^{n_{\text{dim}}} + 2n_{\text{sp}} + 2 \right). \end{aligned} \tag{4.21}$$

Approximations in this step include the infinite number of levels and the absence of special treatment on the coarsest level. Because the problem size, and therefore the cost of additional levels, decays exponentially with the level number l, these costs are small enough to be neglected. If the Galerkin condition is used, additional costs occur here because of the fill-in in the finite-difference stencil on the finer levels. These additional costs are quantified in Section 4.4.3.

The total cost contribution of MGPCG (without Galerkin condition) to a scheme with m implicit stages is approximately equivalent to

$$C_{\text{MGPCG}} = m \left(n_{\text{it-CG}} \left(\frac{2^{n_{\text{dim}}}}{2^{n_{\text{dim}}} - 1} \left(n_{\text{sm}} \left(2n_{\text{sp}} + 4 \right) + 4 \left(\frac{n_{\text{order}} + 1}{2} \right)^{n_{\text{dim}}} + 2n_{\text{sp}} + 2 \right) + 2n_{\text{sp}} + 12 \right) + 2n_{\text{sp}} + 4 \right) \tag{4.22}$$

vector operations with $n_{\text{sp}} = n_{\text{dim}} \cdot n_{\text{order}} + 1$.
This equation is linear in the number of implicit stages m, Jacobi smoothing iterations n_{sm}, and PCG iterations $n_{\text{it-CG}}$. The dimensionality and finite difference order contribute non-linearly. Since they determine the number of stencil points, the equation is also not linear in n_{sp}. Also note that in practice there is a non-linear relationship between the parameter and the PCG iteration counts which usually cannot be chosen.

Single-run MGPCG costs, which correspond to the costs per stage, are given in Table 4.2 as terms linear in the number of MGPCG iterations $n_{\text{it-CG}}$ for a couple of exemplary scenarios. The costs are relatively high, especially when compared to the previously determined scheme costs. Increases in each parameter lead to significant increases in the costs.
Since the cost of a multigrid run is relatively high, it is worth checking at which iteration count reduction the additional effort of preconditioning pays off. For this, we compare the costs of a preconditioned and unpreconditioned conjugate gradient method run. This leads to the following ratio of iteration counts which has to be surpassed for the multigrid preconditioner to improve efficiency:

$$C_{\text{CG}}^{\text{1it}} \overset{!}{>} C_{\text{MGPCG}}^{\text{1it}} \quad \Leftrightarrow \tag{4.23}$$

$$\frac{n_{\text{it-CG}}}{n_{\text{it-MGPCG}}} \overset{!}{>} 1 + \frac{1}{2n_{\text{sp}} + 12} \frac{2^{n_{\text{dim}}}}{2^{n_{\text{dim}}} - 1} \left(n_{\text{sm}} \left(2n_{\text{sp}} + 4 \right) + 4 \left(\frac{n_{\text{order}} + 1}{2} \right)^{n_{\text{dim}}} + 2n_{\text{sp}} + 2 \right). \tag{4.24}$$

Table 4.2: Exemplary numbers of vector operations required by one MGPCG run for various parameter values based on Equation 4.22. Additionally, the iteration count ratios of CG to MGPCG that have to be exceeded for the preconditioning to pay off are given.

n_{dim}	n_{order}	n_{sm}	MGPCG vector operations	$n_{\text{it-CG}}/n_{\text{it-MGPCG}}$ limit
2	2	2	87.33 $n_{\text{it-CG}}$ + 9	3.97
2	2	4	124.67 $n_{\text{it-CG}}$ + 9	5.67
2	4	2	148.67 $n_{\text{it-CG}}$ + 13	4.56
2	4	4	207.33 $n_{\text{it-CG}}$ + 13	6.91
3	2	2	100.86 $n_{\text{it-CG}}$ + 11	3.88
3	2	4	142.00 $n_{\text{it-CG}}$ + 11	5.46
3	4	2	210.00 $n_{\text{it-CG}}$ + 16	5.53
3	4	4	278.57 $n_{\text{it-CG}}$ + 16	7.33

Values for the previously considered scenarios are also given in Table 4.2. In these cases, the preconditioning must reduce the CG iteration counts by a factor between $\frac{1}{4}$ and $\frac{1}{8}$, depending on the dimensionality of the problem, the finite difference order, and the number of smoothing iterations.

Table 4.3: Examples for the total number of vector operations required by DIRKN, DIRK-2-0 and DIRK-2-0s, including their MGPCG costs.

m	n_{dim}	n_{order}	n_{sp}	$n_{\text{it-CG}}$	n_{sm}	C_{MGPCG}	$C_{\text{DIRKN}} = C_{\text{DIRK-2-0}}$	$C_{\text{DIRK-2-0s}}$	$\frac{C_{\text{DIRK-2-0s}}-C_{\text{DIRKN}}}{C_{\text{DIRKN}}}$
3	2	2	5	3	2	828	883	875	-0.91%
3	2	4	9	3	2	1404	1483	1451	-2.16%
5	2	2	5	3	2	1380	1481	1477	-0.27%
5	2	4	9	3	2	2340	2481	2437	-1.77%

A comparison of the overall costs of DIRKN, DIRK-2-0, and DIRK-2-0s is given in Table 4.3. We see that the costs of solving the linear systems are relatively large compared to the other costs of the schemes. This diminishes the differences between the methods to only a few percent.
The relative differences in cost between the methods $\frac{C_{\text{DIRK-2-0s}}-C_{\text{DIRKN}}}{C_{\text{DIRKN}}}$ are also visualized in Figure 4.3. It shows that DIRKN and DIRK-2-0 improve relative to DIRK-2-0s with an increasing number of stages m. The colored lines indicate the zero crossings of the respective planes. For the given parameter range, the other two only manage to outperform DIRK-2-0s in the 2D second-order case for $m > 5.83$. The relative differences become more pronounced at low iteration counts, while they become negligible for larger iteration numbers.

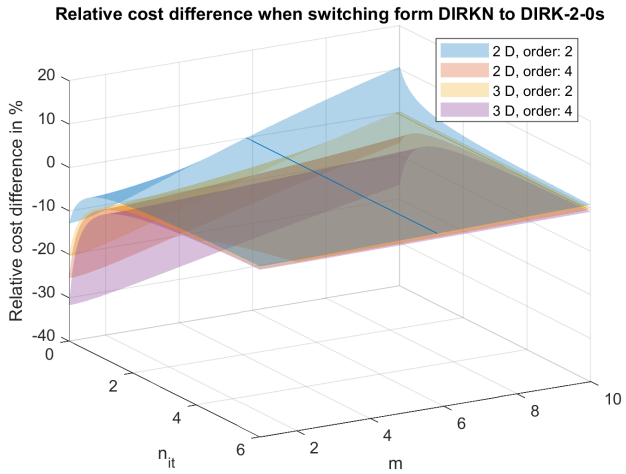

Figure 4.3: Relative cost differences between DIRKN (or DIRK-2-0) and DIRK-2-0s $\frac{C_{\text{DIRK-2-0s}} - C_{\text{DIRKN}}}{C_{\text{DIRKN}}}$.

4.4.3 Costs related to the Galerkin condition

Using the Galerkin condition for the coarse grid system matrices in the multigrid preconditioner leads to a fill-in in the corresponding stencil. The number of stencil points increases from $(n_{\text{dim}} n_{\text{order}} + 1)$ to $(n_{\text{order}} + 1)^{n_{\text{dim}}}$. The finest grid is not impacted by this. The additional operations required because of this phenomenon are quantified in this section.
Multigrid performs the same operations for every grid point on every level, except for the finest level. Therefore, we can determine the total cost increase by multiplying the additional costs per grid point by the number of all grid points on all levels except the finest. A level has $\frac{1}{2^{n_{\text{dim}}}}$-th of the grid points of the previous level because of the grid ratio of two. The sum of the number of grid points on all levels except the finest can be approximated by

$$\sum_{l=1}^{\infty} \left(\frac{N}{2^{n_{\text{dim}}}} \right)^l = \frac{N}{2^{n_{\text{dim}}} - 1}. \tag{4.25}$$

This equation uses the simplification of an infinite number of levels again (cf. Equation 4.21). The sum is equal to $\frac{1}{3}N$ in two and $\frac{1}{7}N$ in three dimensions. On each level, the part of the multigrid cost per level (see Equation 4.20) which depends on the stencil points is $2n_{\text{sp}}(n_{\text{sm}} + 1)$ times the number of grid points. Thus, the difference between using and not using the Galerkin condition per CG iteration in FLOPS is:

$$\frac{N}{2^{n_{\text{dim}}} - 1} \cdot 2 \left(n_{\text{order}} n_{\text{dim}} + 1 - (n_{\text{order}} + 1)^{n_{\text{dim}}} \right) (n_{\text{sm}} + 1). \tag{4.26}$$

Examples for the resulting cost differences between a MGPCG iteration with and without the Galerkin condition are given in Table 4.4. The iteration costs are based on the values from Table 4.2. These examples show that whether the Galerkin condition is used or not has a significant impact on the costs. In the case of 2D second-order finite differences, it increases the cost of one MGPCG iteration by approximately 9%. For higher dimensions or finite difference orders, this contribution increases further. These additional costs occur whether the Galerkin condition is applied to the whole matrix or only partially to the matrix M or K since the fill-in occurs for each. The Galerkin

Table 4.4: Approximated MGPCG iteration costs in vector operations (N FLOPS) and the impact of the Galerkin condition on it. The cost increases when using the Galerkin condition are given as absolute and relative values.

n_{dim}	n_{order}	w/o Galerkin	w/ Galerkin	Abs. increase	Rel. increase
2	2	87	95	8	9%
2	4	149	181	32	21 %
3	2	101	118	17	17 %
3	4	210	306	96	46 %

condition also has some additional storage requirements, but they are not considered here.
Since the cost of MGPCG contributes most of the total time integration scheme cost (see Table 4.3), the relative increase in the overall cost should be similar to those listed here.
All previous and following MGPCG costs do not include the cost increase caused by the Galerkin condition unless stated otherwise.

4.4.4 Cost comparison of explicit and implicit schemes

The better stability properties of implicit time integration methods allow larger time steps than are possible for explicit alternatives. Therefore, we need to derive step sizes for both before we can compare the previously derived computational costs of ERKN and the implicit schemes.

Explicit and implicit time step sizes

We consider a wave source with a fixed frequency f in a heterogeneous medium, which is discretized with a uniform grid. The same analysis is also valid for more complex oscillations with a broader frequency spectrum. In these cases, f is the highest relevant constituent frequency of the source.
To guarantee a sufficient resolution of all oscillations up to this frequency f at all wave speeds, we base the spatial and temporal resolution on the parameter number of points per minimum wavelength N_{ppmw}. In space this means setting the grid spacing Δx equal to the minimum occurring wavelength λ_{min} divided by the desired number of points per minimum wavelength N_{ppmw},

$$\Delta x = \frac{1}{\sqrt{n_{\text{dim}}}} \frac{\lambda_{\text{min}}}{N_{\text{ppmw}}} = \frac{v_{\text{min}}}{\sqrt{n_{\text{dim}}} \cdot f \cdot N_{\text{ppmw}}}. \tag{4.27}$$

The square root of the number of dimensions $\sqrt{n_{\text{dim}}}$ is included to ensure that the desired resolution is also achieved along the diagonals of the grid cells. On the right, we used the relationship of the wavelength to the wave speed, $\lambda_{\text{min}} = \frac{v_{\text{min}}}{f}$.
If we assume that the frequencies do not change in the domain during the simulated period, we can

achieve the same resolution in time by setting the number of time steps of the size Δt per period $T = \frac{1}{f}$ equal to N_{ppmw},

$$\frac{T}{\Delta t} \stackrel{!}{=} N_{\text{ppmw}} \quad \Leftrightarrow \tag{4.28a}$$

$$\Delta t \stackrel{!}{=} \frac{1}{f \cdot N_{\text{ppmw}}}. \tag{4.28b}$$

But in practice, frequencies larger than the source frequency can occur through the superposition of waves, especially in heterogeneous media, or because of numerical dispersion as mentioned in Section 1.3.1. Time steps that are too large cause inaccuracies that change the wavefield over time and can make signals unrecognizable, as shown in the numerical experiment of Section 5.2. Therefore, it is often beneficial to choose smaller time steps. We introduce an additional parameter $c_f \leq 1$ for this purpose, which has to be tuned to the scenario it is used in and the desired level of accuracy,

$$\frac{T}{\Delta t} \stackrel{!}{=} \frac{1}{c_f} N_{\text{ppmw}} \quad \Leftrightarrow \tag{4.29a}$$

$$\Delta t \stackrel{!}{=} \frac{c_f}{f \cdot N_{\text{ppmw}}}. \tag{4.29b}$$

Choosing the time step size like this without additional constraints is only possible for unconditionally stable implicit time integration methods. Otherwise, the stability of the numerical scheme is not guaranteed. For explicit methods the Courant-Friedrichs-Lewy (CFL) condition [CFL67] states, that the time step size Δt must be smaller than the spatial resolution divided by the maximum wave speed v_{max} multiplied with a problem-specific constant $c_{\text{CFL}} \in (0, 1]$, that decreases with an increasing number of dimensions or the order of the spatial discretization scheme,

$$\Delta t \leq c_{\text{CFL}} \cdot \frac{\Delta x}{v_{\text{max}}}, \quad c_{\text{CFL}} \in (0, 1]. \tag{4.30}$$

With the above definition for Δx, this results in the following limit for the time step size:

$$\Delta t \leq \frac{1}{f \cdot N_{\text{ppmw}}} \cdot \frac{v_{\text{min}}}{v_{\text{max}}} \cdot \frac{c_{\text{CFL}}}{\sqrt{n_{\text{dim}}}}. \tag{4.31}$$

The equivalent temporal resolution is:

$$\frac{T}{\Delta t} = N_{\text{ppmw}} \left(\frac{\sqrt{n_{\text{dim}}}}{c_{\text{CFL}}} \frac{v_{\text{max}}}{v_{\text{min}}} \right). \tag{4.32}$$

Comparison with the desired resolution (Equation 4.29a) shows that they differ by the factor:

$$\left(\frac{c_f \sqrt{n_{\text{dim}}}}{c_{\text{CFL}}} \frac{v_{\text{max}}}{v_{\text{min}}} \right) \geq 1. \tag{4.33}$$

Since this factor describes the ratio of the required to the desired number of time steps, we refer to it as the oversampling factor. It varies between different scenarios because of its dependence on the contrast of the wave speed in the medium and the number of dimensions. But it is also affected by the order of the spatial discretization through the value c_{CFL}.

Implicit time integration schemes can achieve better stability properties which lead to them not suffering from the limitations of the CFL condition. The oversampling factor is equal to the number of time steps that have to be computed by an explicit method for one step of a stable implicit method. It is therefore a good indicator of how much savings are possible in theory by switching to implicit methods.
The factor is proportional to the wave speed contrast. It is therefore more likely for implicit methods to outperform implicit ones in scenarios with high wave speed contrasts.

This analysis is in large parts copied from notes of Longfei Gao [Gao19] with some modifications.

Implicit-explicit cost ratios and wave speed contrasts

Table 4.5: Examples for the total number of vector operations for DIRKN or DIRK-2-0 and ERKN. The wave speed contrast $\frac{v_{\max}}{v_{\min}}$ is computed according to Equation 4.36 with $c_{\mathrm{CFL}} = 1$ and $c_f = 1$. For $c_f = \frac{1}{3}$, the contrast values are three times larger.

m	n_{dim}	n_{order}	n_{sp}	$n_{\text{it-CG}}$	n_{sm}	C_{MGPCG}	C_{DIRKN}	C_{ERKN}	$C_{\mathrm{DIRKN}}/C_{\mathrm{ERKN}}$	Contrast
3	2	2	5	3	2	828	883	56	15.77	11.15
3	2	4	9	3	2	1404	1483	80	18.54	13.11
5	2	2	5	3	2	1380	1481	104	14.24	10.07
5	2	4	9	3	2	2340	2481	144	17.23	12.18

With the previously determined estimate of the costs of solving the linear systems of Section 4.4.2, the costs of explicit and implicit approaches can be compared. Table 4.5 lists the cost of DIRKN and ERKN in the previously used scenarios (Table 4.3). These were calculated under the assumption that the explicit and implicit schemes use methods with the same number of stages. Furthermore, these comparisons are only useful when their accuracies are similar. The numerical experiments in Section 5.6 show that this is not always realistic. The results are also highly dependent on the required CG iterations and the parameters of the solver. But we will use this simplified analysis here as a reference point. Because of the high MGPCG cost, the cost ratios of DIRK-2-0 and DIRK-2-0s are similar to those of DIRKN for equal parameters and therefore omitted.
The implicit schemes are about 14 to 19 times more expensive per time step than the explicit alternative in the included scenarios. Therefore, their time steps need to be larger than the explicit ones by this factor in order to be equally computationally expensive.
The previously derived time step sizes (Equation 4.30 and 4.29b) are not chosen freely, but depend on the scenario they are used in. Their ratio is the oversampling factor (Equation 4.33), which is proportional to the wave speed contrast in the simulation domain. The time step size ratio required for DIRKN and ERKN to have similar costs, therefore, corresponds to a wave speed contrast.
For an implicit method to be more efficient than an explicit one, its cost per time step C_{implicit} times the number of time steps per second, which is equal to the inverse of the step size $\frac{1}{\Delta t_{\mathrm{implicit}}}$, must be smaller than the corresponding value for the explicit method,

$$\frac{C_{\mathrm{implicit}}}{\Delta t_{\mathrm{implicit}}} \overset{!}{\leq} \frac{C_{\mathrm{explicit}}}{\Delta t_{\mathrm{explicit}}}. \tag{4.34}$$

Note that differences in accuracy are not considered here.
This equation can be reordered to yield the cost ratio of the previous table. We also substitute the time step sizes with the previously mentioned formulas,

$$\frac{C_{\text{implicit}}}{C_{\text{explicit}}} \overset{!}{\leq} \frac{\Delta t_{\text{implicit}}}{\Delta t_{\text{explicit}}} = \frac{c_f \sqrt{n_{\text{dim}}}}{c_{\text{CFL}}} \frac{v_{\text{max}}}{v_{\text{min}}}. \tag{4.35}$$

From this, we derive a formula for the value which the wave speed contrast has to exceed for implicit methods to be competitive,

$$\frac{c_{\text{CFL}}}{c_f \sqrt{n_{\text{dim}}}} \frac{C_{\text{implicit}}}{C_{\text{explicit}}} \overset{!}{\leq} \frac{v_{\text{max}}}{v_{\text{min}}}. \tag{4.36}$$

These values are also given for the exemplary scenarios in Table 4.5. Since the constants c_{CFL} and c_f are highly problem-dependent, they are not included in these numbers (set to 1). Both values are lesser than or equal to one, so a low c_{CFL} value reduces the required contrast value while a low c_f value increases it.
While wave speed contrasts of 15 and above occur in seismic modeling, there are many scenarios in which this threshold is not exceeded.
This analysis shows that whether switching from explicit to implicit can save computational effort is highly dependent on the use-case and the actual cost of solving the linear systems.

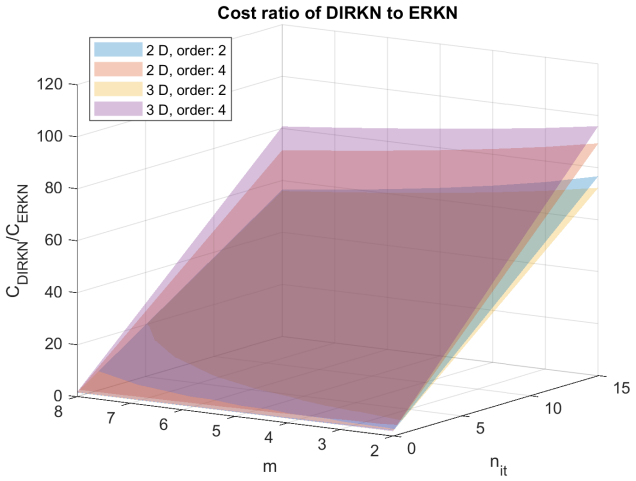

Figure 4.4: Ratio of DIRKN to ERKN cost

Figure 4.4 shows the ratio of the DIRKN to the ERKN cost over the number of CG iterations and stages for two and three dimensions and finite difference orders two and four. The ratio increases linearly with the iteration count and decrease with the number of stages m. The latter effect is smaller at lower iteration numbers. The differences between the two- and three-dimensional results are relatively small. Higher finite-difference orders mostly increase the slope in the direction of the iteration axis. These results illustrate the importance of decreasing the number of CG iterations to make implicit methods competitive in more scenarios.
The ratios displayed in Figure 4.4 can be approximated reasonably well with bilinear interpolation. Simplified formulas are provided in Table 4.6. They interpolate between the points $(n_{\text{it}}, m) = (0, 2), (0, 8), (15, 2)$ and $(15, 8)$. Simplifications were made to shorten the equations. The formulas should not be used for more than ten stages.

Table 4.6: $C_{\text{DIRKN}}/C_{\text{ERKN}}$ ratio approximation using bilinear interpolation between the points $(n_{\text{it}}, m) = (0, 2), (0, 8), (15, 2)$ and $(15, 8)$

n_{dim}	n_{order}	Bilinear approximation of $\left(\frac{C_{\text{DIRKN}}}{C_{\text{ERKN}}}\right)$ as functions of m and n_{it}
2	2	$1.88 + 5.43\ n_{\text{it-CG}} - 0.22\ m\ n_{\text{it-CG}}$
2	4	$1.93 + 6.22\ n_{\text{it-CG}} - 0.19\ m\ n_{\text{it-CG}}$
3	2	$1.91 + 5.05\ n_{\text{it-CG}} - 0.18\ m\ n_{\text{it-CG}}$
3	4	$1.95 + 6.60\ n_{\text{it-CG}} - 0.17\ m\ n_{\text{it-CG}}$

4.5 Initial guesses

Accurate initial guesses can reduce the required amount of CG iterations further. Since the implicit schemes DIRK-2-0(s) and DIRKN solve for different variables, they also require different initial guess (IG) strategies. In general, one has more options for P_i in DIRK-2-0(s) than for its second derivative R_i in DIRKN. This is because, at the time steps, there are only values computed for the pressure and its first temporal derivative, P^N and Q^N, but none for the second temporal derivative R. The values of P^N and Q^N are more relevant for estimations of P_i than R_i. But multiple approaches can still be derived for DIRKN as well.
We introduce multiple initial guess strategies in this section and determine their computational cost. Their performance is evaluated in the following chapter. As a reference solution, a zero vector is used as the initial guess for all schemes.
While arbitrarily accurate estimates are possible, it is important to ensure that the additional costs of computing an initial guess do not outweigh the savings. We focus on simple strategies with low additional computational costs. Note that more advanced strategies also require more storage, especially in the case of DIRK-2-0(s), but the focus here remains on the computational complexity. While we do not explore this approach, one should keep in mind that initial guess strategies are essentially predictors. If one would remove the linear system solver and use their results directly the overall time stepping scheme is a predictor-corrector method. As outlined in Section 1.3.2, some researchers have obtained good results with such methods (e.g. [YWL12]). This may motivate further investigations into more accurate and expensive initial guess strategies used with or without a linear solver but is beyond the scope of this work.

4.5.1 Different initial guesses for P_i^N in DIRK-2-0(s)

Since the P-values do not change drastically between time steps or stages, it is likely beneficial to initialize the data with a previous result. One option is to simply use the last time-step result P^N. If this value is not computed at every time step, an additional element-wise vector multiplication is required to compute it from MP^N.
While stage results are less accurate than the ones at time steps, they are often closer to the current stage in time. Because temporal proximity is likely more important than accuracy for the quality of an initial guess, we consider a strategy that determines the closest stage or time-step result of the current and the previous time step.
Additionally, the strategy of using the previous result of the same stage is included. This would be a

good strategy if the error at the stages would be systematic, meaning the error at a specific stage would deviate from the exact solution similarly at different time steps.
The derivative Q of MP is also available at the time steps and allows for more sophisticated initial guesses. Among these, we consider an explicit Euler step, which uses these values of the previous time step, and Hermit interpolation/extrapolation, which uses the two values at the two previous time steps.
Additionally, we propose an approach based on central differences. It uses the derivative of the time step Q^N and the value of the stage result that has an equal distance in time from the latest time step as the current stage to compute but in the opposite direction. Thus, it only has a theoretical basis for methods with symmetric stage time coefficients, meaning $c_j = 1 - c_{m+1-j} \quad \forall j = 1, \dots, m$, but could be applied to any method.
The different strategies including their respective formulas and costs in vector operations C_{IG} are given in Table 4.7.

Table 4.7: Initial guess strategies for P_i^N in DIRK-2-0(s) with their respective formulas and costs in vector operations.

#	Name	C_{IG}	Equation
1	No IG	0	0
2	Time step	1	$P^N = M^{-1}MP^N$
3	Closest result	0 or 1	P^N, P^{N-1}, P_j^N or $P_j^{N-1} \quad \forall j = 1, \dots, m$
4	Same stage	0	P_i^{N-1}
5	Forward Euler	1	$M^{-1}\left(c_i \Delta t Q^N + MP^N\right)$
6	Hermite interpolation	8	$M^{-1}(c_i^2(2c_i+3)MP^{N-1} + (1-c_i^2(2c_i+3))MP^N + c_i^2(1+c_i)\Delta t Q^{N-1} + c_i(1+c_i)^2\Delta t Q^N)$
7	Central differences	4	$2c_i \Delta t M^{-1} Q^N + M^{-1} MP_{m+1-i}^{N-1}$

4.5.2 Initial guess strategies for R_i^N in DIRKN

For DIRKN, one needs to approximate the second temporal derivative R (or Laplacian) of the acoustic pressure P. This value is not explicitly computed at the time steps and its derivatives are not part of the integration scheme. The only values that are available at the time steps are the sums $R_P^N := \sum_{i=1}^m \bar{b}_i R_i$ and $R_Q^N := \sum_{i=1}^m b_i R_i$. While they could be viewed as an approximation of the actual value at the time step R^N, they are better interpreted as correction terms in the time step equations. They differ from the best possible approximation of R^N to achieve higher algebraic orders than a Taylor-style expansion using an approximation of R^N would provide.
Nevertheless, we examine the suitability of R_P^N and R_Q^N as initial guesses.
Additionally, we consider an approximation of R^N using backward differences with the latest time step results for Q.
Similar to DIRK-2-0(s), we also include the previous value of the same stage and the time step (R_Q^N) or stage result that is closest in time. In an attempt to increase accuracy, we use linear inter-/extrapolation with R_Q^N and R_Q^{N-1}.

Table 4.8: Initial guess strategies for R_i^N in DIRKN

#	Name	C_{IG}	Equation
1	No IG	0	0
2	Time step P	0	$R_P^N = \sum_{i=1}^m \bar{b}_i R_i^{N-1}$
3	Time step Q	0	$R_Q^N = \sum_{i=1}^m b_i R_i^{N-1}$
4	Derivative approx. Q	2	$\frac{1}{\Delta t}(Q^N - Q^{N-1})$
5	Same stage	0	R_i^{N-1}
6	Closest stage	0	R_j^N or $R_j^{N-1} \quad \forall j = 1, \ldots, m$
7	Lin. extrapolation R_Q	3	$(1 + c_i)R_Q^N + c_i R_Q^{N-1}$

4.5.3 Relative costs of the initial guess strategies

The number of vector operations saved when using an IG compared to using zeros is equal to the difference in CG iterations $\Delta n_{\text{CG-it}}$ multiplied with the cost of one CG iteration C^1_{MGPCG} and the number of stages m,

$$\Delta n_{\text{CG-it}} m C^1_{\text{MGPCG}}. \tag{4.37}$$

The savings through initial guesses can be quantified using the following formula, with $n^{\text{IG}}_{\text{it-CG}}$ and $n^{\text{noIG}}_{\text{it-CG}}$ denoting the CG iteration count with and without initial guesses, respectively:

$$m \left(\left(n^{\text{noIG}}_{\text{it-CG}} - n^{\text{IG}}_{\text{it-CG}} \right) C^1_{\text{MGPCG}} - C_{\text{IG}} \right) \overset{!}{\geq} 0. \tag{4.38}$$

An initial guess strategy is worth its cost if this value is greater than or equal to zero. This is equivalent to the difference in iterations with and without initial guesses being greater than the initial guess cost divided by the MGPCG iteration cost,

$$\Delta n^{\text{IG}}_{\text{CG-it}} := n^{\text{noIG}}_{\text{it-CG}} - n^{\text{IG}}_{\text{it-CG}} \overset{!}{\geq} \frac{C_{\text{IG}}}{C^1_{\text{MGPCG}}}. \tag{4.39}$$

This boundary is visualized in Figure 4.5 for four different scenarios. They include two- and three-dimensional cases with finite-difference orders of two and four. All use two smoother iterations in the multigrid preconditioner. These parameters influence the results through the cost of the MGPCG iteration. The larger the costs of one iteration, the smaller the difference in iterations needs to be for an initial guess strategy to be worth its cost. Because of the generally high cost of an MGPCG iteration, all initial guesses pay off at relatively small iteration reductions. The largest minimum required difference of the cases considered here corresponds to the Hermite interpolation for DIRK-2-0(s) with a cost of eight vector operations and is equal to 0.0916 iterations in the two-dimensional, second-order scenario.

Similarly, one can compare two different IG strategies. For example, Hermite interpolation is superior to the froward Euler method in the case of two dimensions and second order finite differences for DIRK-2-0(s) if

$$n^{\text{IG4}}_{\text{it-CG}} - n^{\text{IG8}}_{\text{it-CG}} \geq \frac{C_{\text{IG8}} - C_{\text{IG4}}}{C_{\text{CG-it}}} = \frac{8 - 1}{87.33} = 0.0802. \tag{4.40}$$

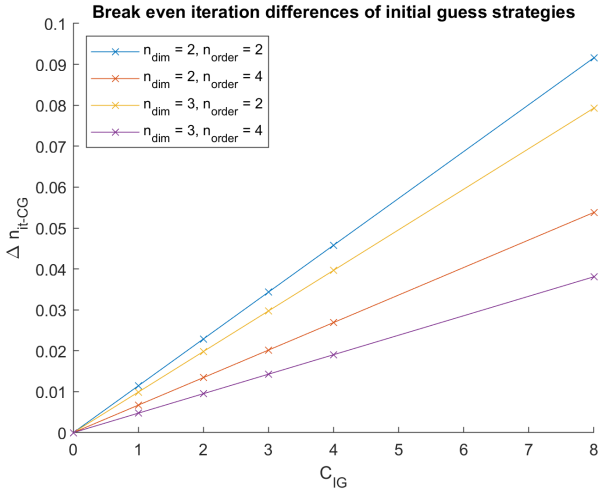

Figure 4.5: Break even iteration differences of initial guess strategies. The x-axis contains the costs of initial guess strategies in vector operations (N FLOPS). The y-axis shows the average difference in iteration counts of the conjugate gradient method when an initial guess strategy is used versus when the values are initialized with zeros. The lines highlight the iteration differences a strategy has to exceed at a specific cost in order to reduce the overall computational cost.

5 Numerical experiments

The numerical experiments in this chapter provide further insight into the performance of the different time integration schemes. The common scenario of the experiments, including the domain, source signal, used Runge-Kutta (-Nyström) methods, and multigrid parameters, is specified in the first section. The experiments include investigations into the behavior of the time integration schemes DIRKN, DIRK-2-0, and DIRK-2-0s with various coefficients and other parameters. This includes a test with a direct solver, the conjugate gradient method with and without the multigrid preconditioner, the different initial guess strategies, and the preconditioner with and without the Galerkin condition. The focus lies on the effect that these and other design choices have on the accuracy and cost of the schemes. The last experiment uses the knowledge gained to choose promising configurations and provides a detailed comparison of their costs and errors. This also allows the identification of the most efficient configurations at different levels of accuracy.

5.1 Experiment setup

5.1.1 Simulation domain

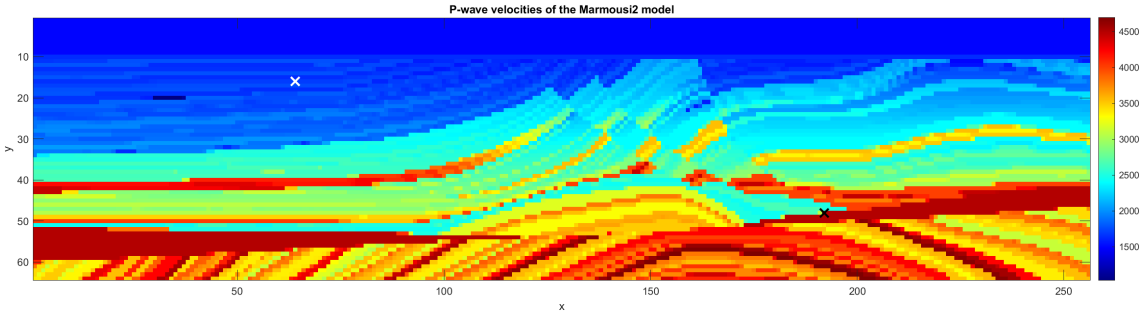

Figure 5.1: P-wave velocities of the Marmousi2 data set in m/s, truncated and rescaled to a 256×64 grid. The grid resolution is $\Delta x = \Delta y = 55.56m$.

All experiments use the same simulation domain and wave speed distribution, except for some rescaling of the velocities in the last experiment. The used configuration is based on the P-wave velocities from the popular AGL Elastic Marmousi data set, which is also known as Marmousi2 [MWM06]. It is a heterogeneous model that contains large velocity changes in all directions. These discontinuities make the model challenging, but representative of real-world scenarios. To reduce the computational effort required for the simulations we resample the velocities from the original 701×2721 grid on a $64 \times 256 = 2^6 \times 2^8$ grid. This is achieved by cropping the medium from its original size of 17km $\times$ 3.5km to a 255:63 $\approx$ 4:1 aspect ratio of 14.167km $\times$ 3.5km, selecting the middle section. The new grid points are obtained through bilinear interpolation with the original

grid. This leads to a spatial resolution of $\Delta x = \Delta y = 55.56m$. The aspect ratio of 4:1 also enables the direct application of the multigrid preconditioner with a grid ratio of two.
This spatial resolution is relatively coarse. But since the reference signal for the error computations is computed with the same spatial discretization, the errors and artifacts caused by this do not contribute to the measured error.
The resulting model has a mean velocity of $2671.29\frac{\mathrm{m}}{\mathrm{s}}$ and a wave speed contrast of $\frac{v_{\max}}{v_{\min}} = \frac{4700\frac{\mathrm{m}}{\mathrm{s}}}{1028\frac{\mathrm{m}}{\mathrm{s}}} = 4.57$. While absorbing boundary conditions would be more realistic for seismic scenarios, periodic boundary conditions are used because of their simplicity.
As an initial condition, the acoustic pressure P^0 and its derivatives are set to zero. The waves are introduced through a source term and originate from a single point source located at $(16, 64) = (\frac{2^6}{4}, \frac{2^8}{4})$, which is marked in Figure 5.1 with a white x. The source value, which is added to the second temporal derivative of the acoustic pressure, changes according to the Ricker wavelet

$$S(t) = \pi^2 f^2 (t - t_0) \left(4\pi^2 f^2 (t - t_0)^2 - 6\right) \; e^{-\pi^2 f^2 (t-t_0)^2} \tag{5.1}$$

with the frequency $f \approx 0.726906Hz$ and delay $t_0 = \frac{1}{f}$. The frequency is calculated through its relationship with the spatial resolution Δx (see Equation 4.27),

$$f = \frac{v_{\min}}{\sqrt{n_{\dim}} \cdot N_{\text{ppmw}} \cdot \Delta x}. \tag{5.2}$$

The number of points per minimum wavelength N_{ppmw} must be chosen large enough to guarantee a good representation of the waves so no information is lost. But it also needs to be small enough to limit computational costs. We choose a value of $N_{\text{ppmw}} = 18$ which was suggested by [Gao19] and seems to fulfill these demands.

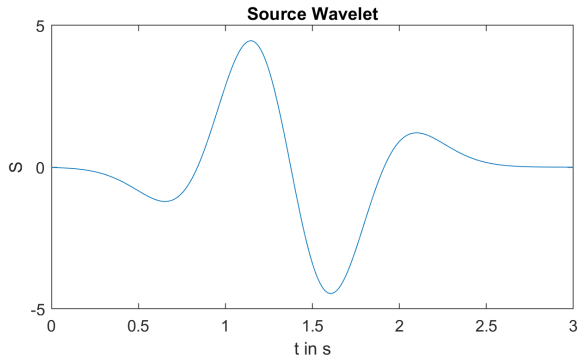

Figure 5.2: Source wavelet. The source term is added to the Laplacian of the acoustic pressure.

The source oscillates for approximately $t \in [t_0 - \frac{1}{f}, t_0 + \frac{1}{f}]$ and is close to zero outside this interval. It is displayed in Figure 5.2. The frequency f is used to determine the time step size of the implicit methods in the experiments. The default time step size is chosen in accordance with the analysis in Section 4.4.4 with the previously used resolution of 18 points per minimum wavelength, $N_{\text{ppmw}} = 18$ and time step size reduction factor of one-third, $c_f = \frac{1}{3}$,

$$\Delta t = \frac{c_f \cdot T}{N_{\text{ppmw}}} = \frac{c_f}{f \cdot N_{\text{ppmw}}} \approx 0.025476s. \tag{5.3}$$

A virtual receiver is located at the point $(48, 192) = (\frac{3}{4} \cdot 2^6, \frac{3}{4} \cdot 2^8)$ in the grid, which is marked by a black x in Figure 5.1. The signal recorded by it is the basis for the error computations. Only using

the one-dimensional recorded data simplifies the comparison of different schemes and methods.
The reference solution for the error is produced by a DIRKN run using the RK18 method (see Table 5.1) with a direct solver for the linear systems. It uses the same spatial grid and time-step size, which is one-fourth the size of the other methods. We use RK18 because of its high order of 5 and good performance (see Section 5.2). With the reduced time step size it is more accurate than all other configurations that use the implicit time integration schemes and does not introduce a bias for certain methods. DIRKN is chosen because it is the only already established scheme of the three used time integration schemes.
As an error measure, the mean of the squared signal differences at those time steps that are common to both runs is used. This means that the signal at each time step of the configuration, which is being evaluated, is compared to every fourth time step of the reference solution.
The experiments simulate the wave propagation for eight seconds, which is sufficient for the source signal to reach the receiver and to produce a couple of peaks, based on which the different configurations can be compared.

To ensure that numerical results are not specific to this model, they have been verified qualitatively with simulations on a homogeneous grid. It has the same number of grid points, spatial resolution, source, receiver, and simulated time period. The wave speed is chosen as $2671\,\frac{\mathrm{m}}{\mathrm{s}}$ which corresponds to the mean of the other domain. The results are not included here to maintain brevity.

5.1.2 DIRK and DIRKN methods

The different methods used in the experiments are listed in Table 5.1. They include 10 Runge-Kutta-Nyström and 21 Runge-Kutta coefficients. All are diagonally implicit with identical diagonal elements (SDIRK) except for explicit stages in some Runge-Kutta methods (e.g. ESDIRK).
Only highly-stable methods are included, which means A-, strong A- (sA), or L-stability for the Runge-Kutta methods and unconditional stability (ucs), P-, R-, or RL-stability for the Nyström methods. Note that there are different definitions of P-stability as discussed in [ACM06]. They introduced their own definition, but the majority of publications use the original one. For the purposes of this work, differentiation between them does not seem necessary.
The coefficients are transformed from RK to RKN and vice versa in order to make them available for the different time integration schemes they are used with. All used time integration schemes require RKN coefficients, but DIRK-2-0s also needs the RK equivalents. Note the problem with RKN to RK conversion, which is outlined in Section 4.2.
The preservation of algebraic order and stability properties is not ensured for the transformed coefficients and left for further investigations. Checking whether all methods actually fulfill their claimed order and stability properties might also be useful but is beyond the scope of this work.

5.1.3 MGPCG setup

As a linear system solver, the conjugate gradient method with a multigrid preconditioner is used. Initial guess strategies, other than initializing with zeros, and tolerances vary between the experiments. The preconditioner is used with a grid ratio of two and six grid levels, which corresponds to a 8×2 grid on the coarsest level. It is solved with a direct solver. The grid transfer operators are full weighting for restriction and its transpose for prolongation. If not stated otherwise

Table 5.1: DIRK and DIRKN methods.
Abbreviations: sym.: symmetric; symp.: symplectic; x-disp.: x-th order dispersive; x-diss.: x-th order dissipative; ESDIRK: SDIRK with explicit first stage; SDIRK: DIRK with identical diagonal elements. Other comments serve the identification of the methods within the sources. The numbers in parentheses following the orders of the RK methods denote their stage orders.

Code	Order (stage o.)	Stability	Source	Comments
RKN1	3	R	[SFB90]	$a_{11} = \frac{800}{5329}$
RKN2	3	R	[SFB90]	$a_{11} = \frac{289}{329}$
RKN3	4	RL	[SFB90]	
RKN4	4	R	[SFB90]	
RKN5	4	P	[SFB90]	
RKN6	4	P	[ACM06]	sym., symp.
RKN7	4	P	[FGR01]	sym., symp., 6-disp.
RKN8	4	P	[FGR01]	symp., 8-disp.
RKN9	2	P	[HS89a]	4-disp.
RKN10	2	ucs	[HS89a]	4-disp.
RK1	3 (1)	A	[FGR97]	4-disp., 5-diss.
RK2	3 (1)	A	[FGR97]	4-disp., 7-diss.
RK3	3 (2)	A	[CS79]	ESDIRK
RK4	4 (2)	A	[CS79]	ESDIRK
RK5	5 (1)	A	[CS79]	
RK6	6 (1)	A	[CS79]	last stage explicit
RK7	4 (1)	A	[Ale77]	4-disp.
RK8	3 (1)	A	[Ale77]	4-disp., 3-disp.
RK9	2 (1)	sS	[Ale77]	
RK10	3 (1)	s	[Ale77]	
RK11	4 (2)	L	[KC16]	ESDIRK4(3)6L[2]SA
RK12	3 (2)	L	[KC16]	ESDIRK3(2)5L[2]SA
RK13	3 (2)	L	[KC16]	ESDIRK3(2I)5L[2]SA
RK14	4 (2)	L	[KC16]	ESDIRK4(3I)6L[2]SA
RK15	4 (1)	L	[KC16]	SDIRK4()5L[1]SA
RK16	5 (2)	L	[KC16]	ESDIRK5(3)6L[2]SA
RK17	5 (1)	L	[KC16]	SDIRK5()5L[1]
RK18	5 (2)	L	[KC16]	ESDIRK5(4)7L[2]SA
RK19	3 (1)	A	[HS89b]	6-disp., 3-diss.
RK20	3 (1)	A	[HS89b]	8-disp., 3-diss.
RK21	2 (1)	A	[KC16; QZ92; Zha92]	symp.

the Galerkin condition is used to obtain the coarse system matrices with these operators. One pre- and one post-smoothing iteration are performed on each level (except the coarsest) by the damped Jacobi methods with a damping factor of $\frac{2}{3}$.

5.2 Direct solver

In this first experiment, the results of simulations with the three time integration schemes DIRKN, DIRK-2-0, and DIRK-2-0s used in combination with each of the 10 Nyström and 21 Runge-Kutta methods are examined. An exact linear system solver is used to allow comparison of the different methods without the influence of the error introduced by the conjugate gradient method.

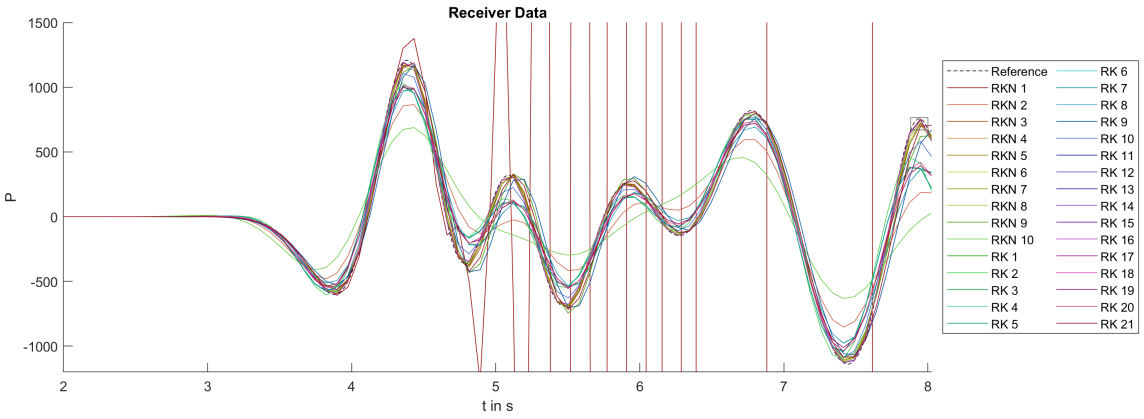

Figure 5.3: Signal at the virtual receiver of DIRKN in combination with different sets of coefficients, a direct solver, and a larger time step ($c_f = 1$)

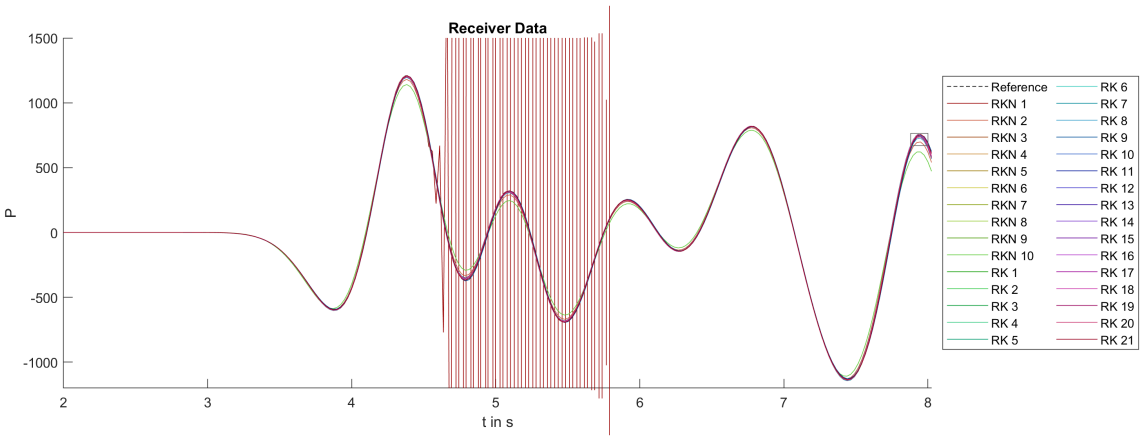

Figure 5.4: Signal at the virtual receiver of DIRKN in combination with different sets of coefficients, a direct solver and a reduced time step size ($c_f = \frac{1}{3}$)

Figure 5.3 shows the signal recorded at the receiver of all methods used with DIRKN and a time step factor of $c_f = 1$. Most signals follow the reference solution roughly, but few reproduce it accurately. This indicates that the time steps, which are chosen based on the desired resolution

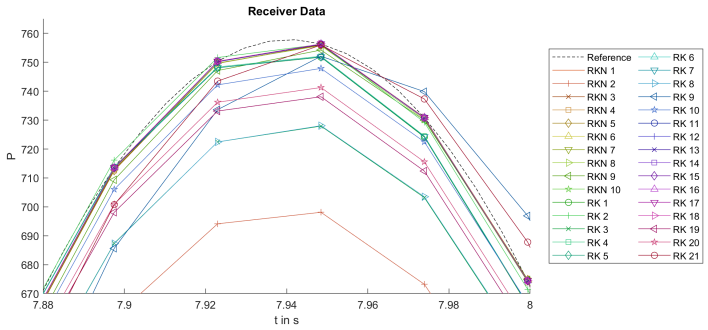

Figure 5.5: Signal of different methods with a direct solver (detailed view of the last positive peak) ($c_f = \frac{1}{3}$)

defined in Equation 4.28a, are too large. Therefore, we reduce the time step size by the factor $c_f = \frac{1}{3}$. This increases the accuracy significantly for all stable methods, as Figure 5.4 shows. The time step factor must be chosen according to the desired accuracy. For our purposes, the value of $c_f = \frac{1}{3}$ seems adequate and is used as a default value in all following simulations.
The RKN1 method is unstable, despite its supposed R-stability. RKN10 on the other hand tends to damp or dissipate the oscillations significantly, which is especially evident for the larger time step. The performance of the other methods can be examined better with the help of a more zoomed-in view. Figure 5.5 shows the last positive peak of this time series at 7.49s in more detail.
While most of the methods are approximately in-phase with the reference solution, the signals of RK9 and RK21 are delayed. A shift in the opposite direction is also possible, as RK2 demonstrates. These phase shifts are larger and become significant for other methods too when using the larger time step.
Regarding the amplitude, the stable methods rarely exceed the reference signal by much. Rather, dampening or dissipation seems to have a larger deteriorating impact on accuracies. This is likely related to the stability properties since dampening on oscillations can help achieve stability. Relevant in this context is also the definition of P-stability, which allows no dissipation, and of RL-stability which requires dissipation. While the P-stable methods are not among the ones with the strongest dissipation, they are not among the most accurate ones either.

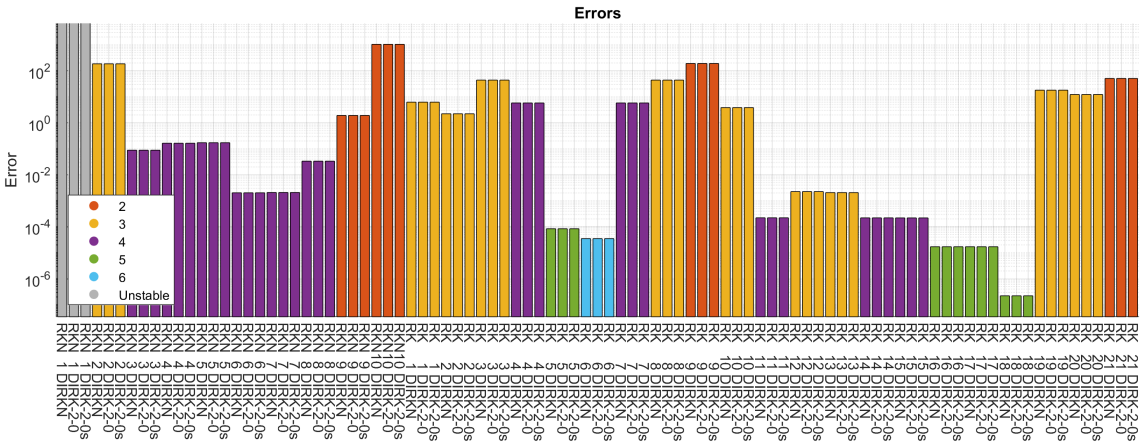

Figure 5.6: Errors of different methods used with each of the tree schemes and a direct solver (method orders are color coded)

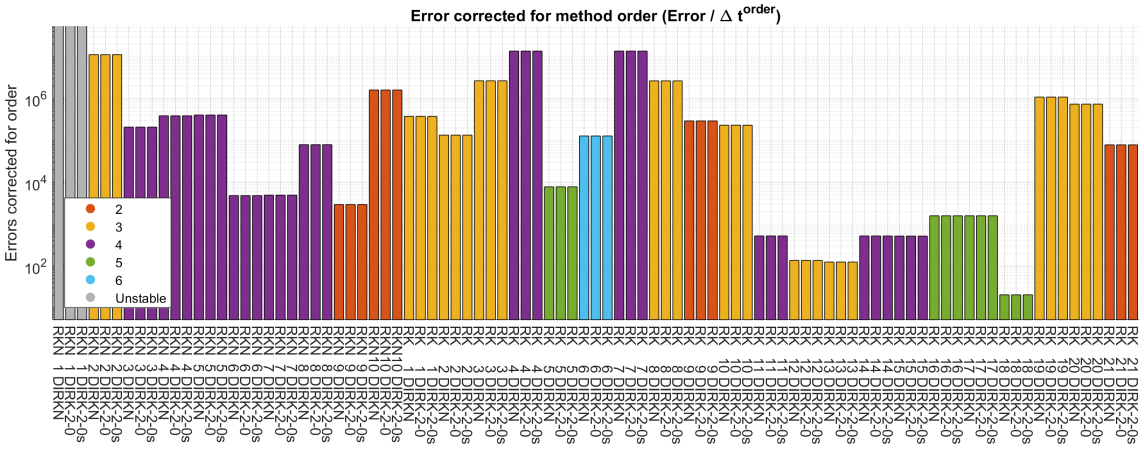

Figure 5.7: Errors of different methods used with each of the tree schemes and a direct solver corrected for the methods' orders (method orders are color coded). The correction consists of multiplying the error values by the inverse of the time step width to the power of the respective methods' orders.

The overall accuracy can be examined more easily using the error values based on the signals' mean squared error to the reference signal over the whole time series. Figure 5.6 shows these errors for all combinations of methods and time integration schemes.
The first thing to notice is the good agreement of the error values of DIRKN, DIRK-2-0, and DIRK-2-0s for each method. This is to be expected since the schemes are analytically identical and there is no source of errors except for small numerical artifacts.
That the schemes also produce almost identical errors when used with RKN coefficients shows that the transformation of DIRKN coefficients into DIRK-2-0s coefficients (which are essentially Runge-Kutta coefficients) works at least for the used methods. In the tests, no detrimental effects of their usage are noticeable.

Certain differences in error values are to be expected when comparing methods of different orders. An attempt to correct for this is given in Figure 5.7. The error values are scaled with the inverse of the time step size to the power of the methods' orders. This reduces the differences but does not come close to eliminating them. The uncorrected numbers already show that one cannot reliably make predictions about a method's accuracy based on its algebraic order alone. Their accuracies in the corrected error measure, therefore, also do not correlate with their algebraic orders.

A better way of comparing the methods is based on the number of operations required by a method to achieve a certain error level since this is more relevant in practical applications. The total cost of an eight second simulation run $C_{8\mathrm{s}}^{\mathrm{total}}$ is equal to the number of time steps $n_{\Delta t} = \frac{8\mathrm{s}}{\Delta t}$ times the cost of one iteration $C_{\Delta t}^{\mathrm{total}}$,

$$C_{8\mathrm{s}}^{\mathrm{total}}(\varepsilon) = \frac{8\mathrm{s}}{\Delta t_{\varepsilon}} C_{\Delta t}^{\mathrm{total}}. \tag{5.4}$$

To obtain results with various costs and accuracies, the experiment was repeated with c_f-values of $\frac{1}{32}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1 and 2, which are a factor in the formula determining the time step size (conf. Equation 4.29b).
When using the direct solver, the cost of solving a linear system is the same for all methods. Its exact value is not relevant here because we consider it in isolation. Using an arbitrary value scales the cost of each method equally and does not change the relative performance differences between

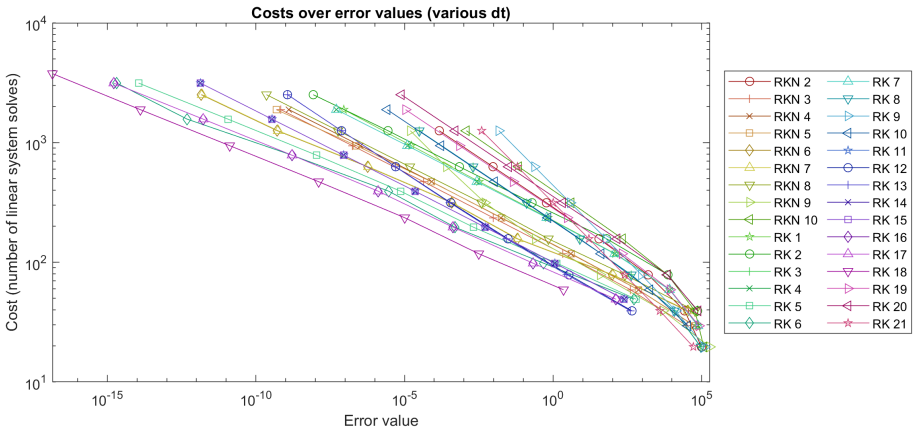

Figure 5.8: Costs over error. Based on DIRKN simulations with $c_f = \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1$ and 2. Lower errors of one method correspond to smaller time steps.

them. We set it equal to one here, which corresponds to a time step cost equal to the number of implicit stages of the use method, $C_{\Delta t}^{\text{total}} := m_{\text{implicit}}$.
The resulting costs of all methods over their error values are given in Figure 5.8. This plot was produced with the DIRKN values, but the other two schemes produce identical charts. For all methods, smaller time steps lead to lower errors and higher costs. Regarding the results in a plot with two logarithmic axes, the values of most methods can be approximated well by a straight line with a negative slope. While not all methods have approximately the same slope, most are close to parallel. This especially means that methods that are cheap in this metric at one error level are also cheap at the other error levels. The most efficient method is RK18, followed by RK17. While RK18 is the method used as a reference, this result is independent of the method used as a reference. The fact that the errors of the other methods are reduced consistently when decreasing the time step size and do not stall also indicates this. RK18 was chosen for the reference solution of all experiments for this reason.
Note that the behavior of the different methods here does not necessarily translate to the conjugate gradient method since it introduces the CG tolerance as an additional parameter, which influences both cost and accuracy.

Which properties of RK or RKN methods cause them to perform better or worse here is impossible to answer without an in-depth analysis. Comparing RK and RKN methods is difficult since information on how properties of RK methods translate to properties of second-order time integration schemes like the DIRKN is hard to find, if it exists at all. The most relevant examples are the different stability categories of first and second-order integration schemes. But the problem also extends to symmetry, symplecticity, dispersive order, dissipative order, and truncation coefficients. A thorough analysis of these connections and their importance to the acoustic wave equation would be interesting but is beyond the scope of this work. We limit ourselves to the search for patterns in the experimental results.
Good stability properties are essential when dealing with the acoustic wave equation. Some methods that are not at least A- or R-stable were included in earlier experiments and all proved to be unstable. While A-stability seems to guarantee good behavior here, R-stability of DIRKN methods is not always sufficient, as the example of RKN1 shows.
It is much more difficult to derive causal relations for other properties. The number of the tested

methods, but also the number of all published highly stable diagonally implicit methods, is not large enough to allow the derivation of statistically meaningful correlations. In our data set, larger numbers of implicit stages seem to be beneficial for the overall error (not regarding the cost), which is not surprising since this results in more degrees of freedom that can be used to achieve higher orders or other desirable properties. Higher-order methods tend to deliver more accurate results, but there are also many exceptions. The symmetric or symplectic methods perform well in the experiments, but since these are only two and three methods, respectively, with the symmetric ones also being symplectic, this may be coincidental. A stage order of two, which is recommended in [KC16], does not seem to be the determining factor for the accuracy here. The methods with higher dispersive or dissipative order or the stiffly accurate property also do not outperform the others consistently.

5.3 Conjugate gradient method with multigrid preconditioning

In the previous experiment, the use of the direct solver leads to the different integration schemes producing identical results because of their analytical equivalence. Using the conjugate gradient method instead introduces differences between them since the error from the inexact solving of the linear systems influences their respective time step results differently. In this experiment, we examine how different CG tolerances impact the receiver signal accuracy and the number of CG iterations for each scheme. CG tolerances of 10^{-1}, 10^{-2}, ..., and 10^{-7} are used.
The conjugate gradient method is used without preconditioner and initial guesses first to isolate the effects of using CG and to establish a baseline that the further improvements can be compared against. Then the multigrid preconditioner is added to demonstrate its impact on error and iterations.

5.3.1 Conjugate gradient method without preconditioner

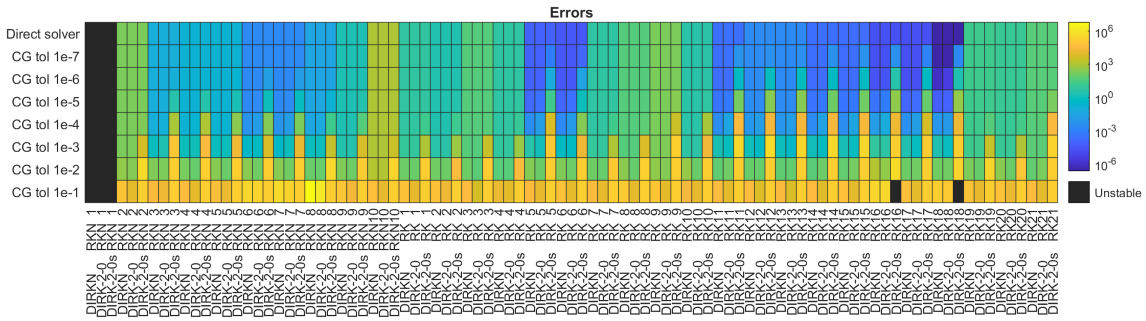

Figure 5.9: Errors of different methods and time integration schemes at different CG tolerances

Figure 5.9 shows the errors of all combinations of methods and time integration schemes for all CG-tolerances. We observe that DIRKN and DIRK-2-0 produce similar errors, with DIRK-2-0 in some cases being a little less accurate. DIRK-2-0s on the other hand often produces significantly larger errors with all methods. These differences between DIRK-2-0s and the other two decrease with decreasing CG tolerances. At a tolerance of 10^{-7}, the differences are not discernable in the plot for any methods except for the most accurate third of them. If a method generally produces larger errors, the differences become negligible at higher tolerances, compared to more accurate methods.

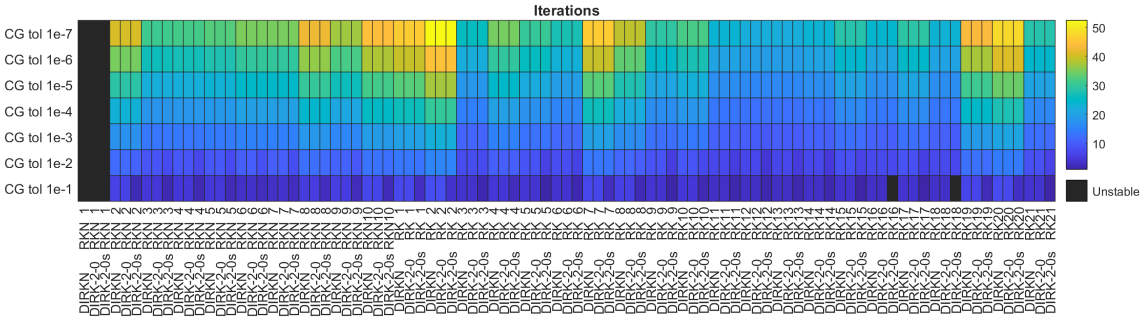

Figure 5.10: Iteration counts of different methods and time integration schemes with different CG tolerances

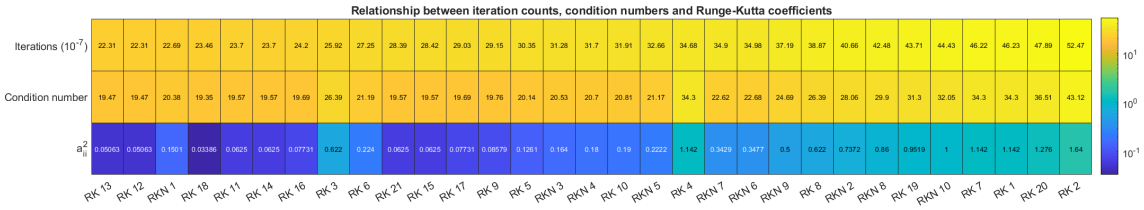

Figure 5.11: Relationship between CG iteration counts, condition numbers of the linear systems' matrices, and the diagonal Runge-Kutta coefficients. The iteration counts are obtained using the unpreconditioned conjugate gradient method with a tolerance of 10^{-7}. Note that the coefficients a_{ii} are identical for all i for every method included here (except for explicit stages). Sorted by iteration counts.

The corresponding average CG iteration counts are shown in Figure 5.10. The three schemes require approximately the same average number of iterations for each method. Some differences only start to occur at the highest tolerances, where the large errors prevent meaningful comparisons. The similarity in iteration counts is likely because the convergence of the conjugate gradient method is only dependent on the condition number of the system matrix. Since the schemes share the same system matrices $\left(M - \bar{a}_{ii}\Delta t^2 K\right)$, their condition numbers, which are equal to the ratio of the largest to the smallest absolute eigenvalue, are also identical. Because the $\bar{a}_{ii}$ varies between methods, their corresponding convergence properties also vary. Figure 5.11 illustrates this relationship. While there are some exceptions, it seems to be true in most cases that lower values for $\bar{a}_{ii}$ lead to lower condition numbers, which leads to lower iteration counts. Since changes in Δt have a similar effect on the system matrix as changes in $\bar{a}_{ii}$, smaller time steps may have a beneficial impact on the condition number as well. Larger Δx could have the same effect since a factor of $\frac{1}{\Delta x^2}$ is included in the matrix K.

Naturally, the iteration counts decrease with an increase in CG tolerance. This is approximately linear over the logarithmic tolerance for all methods, except at the highest tolerances, with decreases between 3 and 7.5 iterations for each tenfold increase in tolerance. The amount of the reduction seems to be related to the absolute iteration counts. The decrease of iterations per tenfold increase in tolerance for each method can be approximated by multiplying the iteration count for a tolerance of 10^{-4} by 0.2487 (standard deviation: 0.0033).

We see that the errors of DIRK-2-0s at a fixed tolerance are larger than those of DIRKN and DIRK-2-0 for all methods or similar. Meanwhile, the iteration counts do not show significant differences between the three schemes. For larger tolerances, the errors increase for all combinations of coefficients and schemes, but the increases for DIRK-2-0s are especially large. This leads to the conclusion that DIRK-2-0s amplifies the errors of the linear system solver more than the other two. DIRK-2-0 and DIRKN amplify the CG method's errors much less and to a similar amount. This means that in order to produce results of the same accuracy, DIRK-2-0s requires much smaller CG tolerances and therefore more iterations compared to DIRK-2-0 or DIRKN.

The iteration counts of up to 53 are too high for this implicit approach to be competitive. This motivates the use of multigrid as a preconditioner.

5.3.2 Conjugate gradient method with multigrid preconditioner

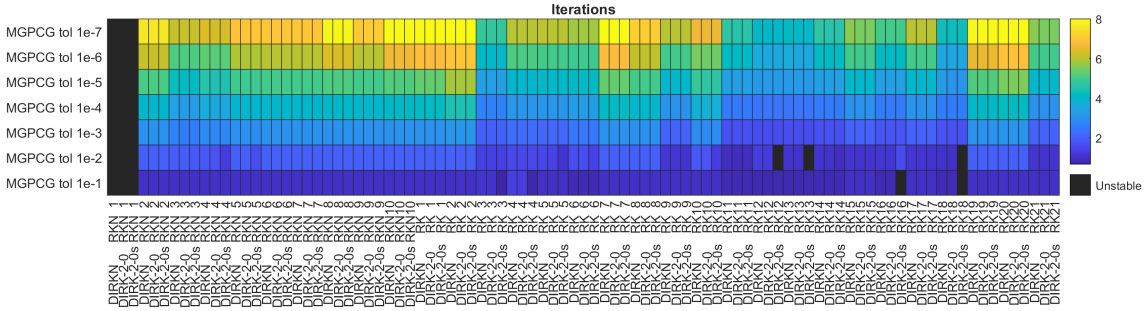

Figure 5.12: Iteration counts of MGPCG at different CG tolerances

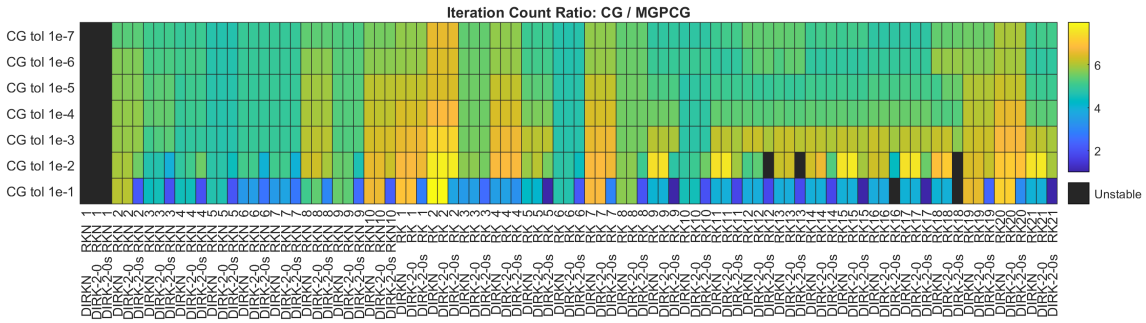

Figure 5.13: Ratio of the number of CG to MGPCG iterations at different CG tolerances

As mentioned previously, we use a multigrid preconditioner with 6 grids, a direct solver on the coarsest level, one pre- and one post-smoothing iteration with damped Jacobi (damping factor $\frac{2}{3}$). This leads to a large reduction in CG iterations and reduces the previous upper limit of 53 to 8 iterations. The iteration counts are given in Figure 5.12.

The absolute decreases in iterations with an increase in tolerances now deviate from the linear relationship of the previous case. The mean iteration decrease with a tenfold increase in the CG tolerance, normalized with the 10^{-4} iteration counts, is similar as before at 0.2801, but the standard deviation has almost increased tenfold to a value of 0.0305.

The overall pattern of the iteration count plot is similar to the one without the preconditioner. Only the small difference between DIRK-2-0s and the other two schemes that previously occurred for

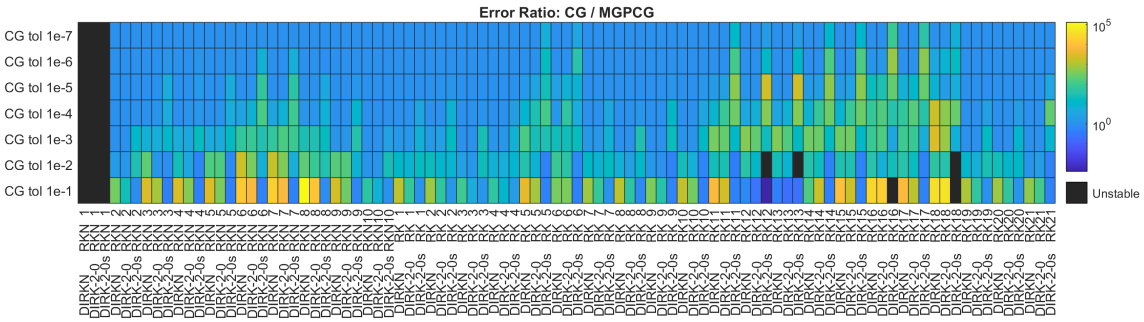

Figure 5.14: Ratio of CG to MGPCG errors at different CG tolerances. The most common color (medium blue) corresponds to a ratio of approximately 1, meaning that there is no difference between CG and MGPCG.

many methods at high tolerances is now limited to few instances.
To analyze the benefit of using the preconditioner, we consider the ratio of the unpreconditioned to the preconditioned iteration counts, which is visualized in Figure 5.13. These values vary greatly, especially at high tolerances. But for a tolerance of 10^{-3} and below the iterations are reduced by a factor of at least 4.6, which is more than the required 3.97 (see Table 4.2) for a reduction in overall complexity.
At higher tolerances, values below this threshold also occur. In these cases, it may seem beneficial to not use the multigrid preconditioner. But one must also consider the impact of the preconditioner on accuracy.
The ratio of the unpreconditioned to the preconditioned error values are given in Figure 5.14. The absolute values are omitted since they produce a picture similar to the plot of the error values of the case without a preconditioner. The plot shows that the differences to the previous result are rather small in the majority of the used configurations. But there is also a significant portion of the results where the use of the preconditioner improves accuracy. Increases of the errors on the other hand are rare.
The accuracy improvements are in most cases larger for DIRK-2-0s, which is likely related to its larger absolute error values. Without it, the error improvements per CG iteration are small and the final result of a CG run is likely close to the specified tolerance. The better accuracy is therefore likely caused by more accurate CG results because of larger improvements per iteration.
The preconditioner affects DIRK-2-0 similarly to DIRKN. This questions the smoothing hypothesis of Section 4.3, which states that the smoothing of the error in the multigrid preconditioner has a positive effect on the error components that the matrix K is applied to.

5.4 Initial guess strategies

In this section, the different initial guess strategies introduced in the Section 4.5 are tested experimentally. According to the previous analysis, only a reduction of the MGPCG iterations by 0.1 is required for it to pay off (see Figure 4.5). This especially means, that in terms of computational complexity, only the absolute iteration reduction of the different strategies is relevant here. Our goal is to identify initial guess strategies that work well for many methods and not to compare different methods in detail.

The previous experiment shows that it might be beneficial to utilize different CG tolerances for the three time integration schemes because of the error amplification of DIRK-2-0s. These observations lead us to evaluate the impact of the initial guesses for MGPCG with tolerances of 10^{-4} and 10^{-7}.

5.4.1 DIRKN initial guesses

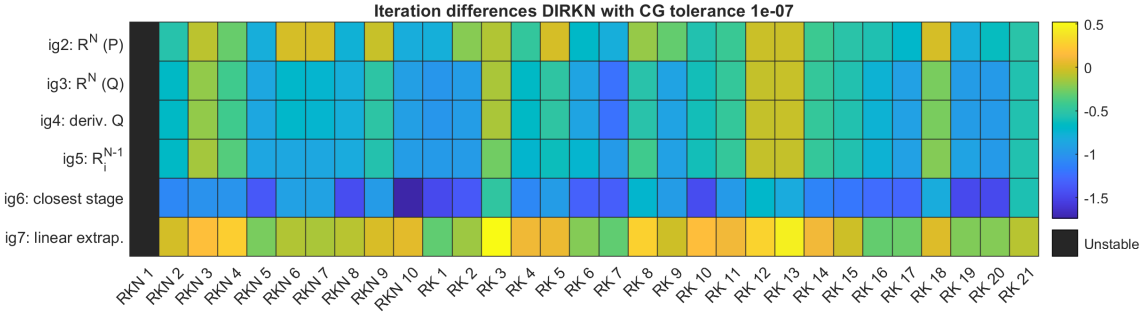

Figure 5.15: Average iteration count changes over all time steps of the respective initial guess strategies compared to using zero. The data is based on simulations using DIRKN and MGPCG with a tolerance of 10^{-7}. For definitions of the initial guess strategies, see Table 4.8.

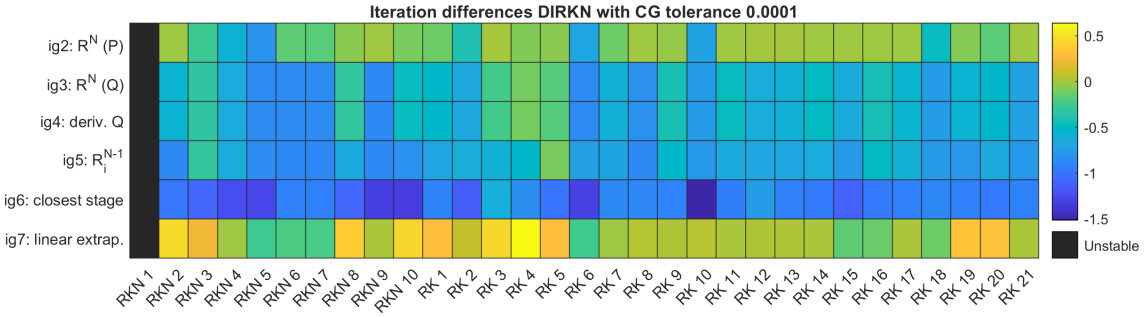

Figure 5.16: Average iteration count changes over all time steps of the respective initial guess strategies compared to using zero. The data is based on simulations using DIRKN and MGPCG with a tolerance of 10^{-4}.

The iteration changes when using the different initial guesses compared to using zero with DIRKN at both tolerances are visualized in Figures 5.15 and 5.16. While the iteration counts improve in most scenarios, there are also many where the iterations increase or are close to their original value. The absolute values lie in a similar range at both tolerances, approximately between -1.75 and +0.65. The corresponding relative values lie between -0.40 and 0.21 for the tolerance 10^{-4} and between -0.26 and 0.12 for 10^{-7}. The relative iteration differences indicate that the effect of initial guesses is greater at higher tolerances, positive and negative. This is not caused by differences in absolute iteration count reductions, but the lower overall iteration counts in the case of 10^{-4}. One would expect this to occur since initial guesses only determine the starting point of a CG run and potentially replace the first iterations, but do not influence the iterations that are needed to increase the accuracy from the tolerance of 10^{-4} to 10^{-7}.
It is easier to compare the initial guess strategies when their corresponding iteration counts are ranked

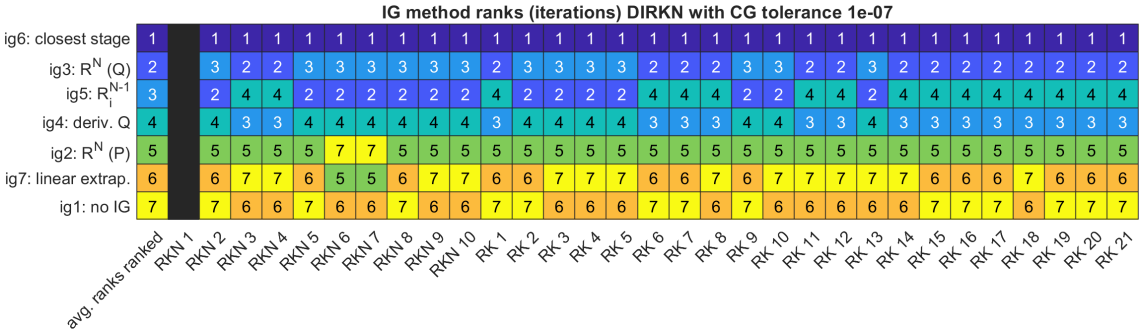

Figure 5.17: Initial guess strategies ranked according to average iteration counts for each method (1: lowest iteration count, lower is better). The rows are reordered according to average rank. The data is based on simulation using DIRKN and MGPCG with a tolerance of 10^{-7}.

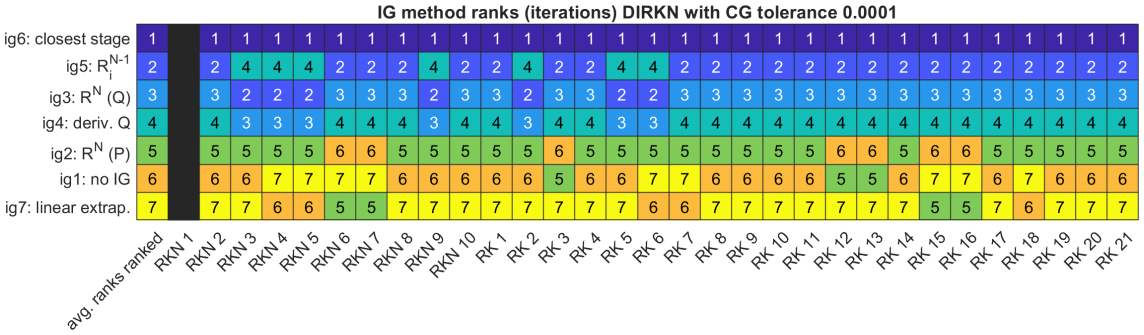

Figure 5.18: Initial guess strategies ranked according to average iteration counts for each method (1: largest iteration count, lower is better). The rows are reordered according to average rank. The data is based on simulation using DIRKN and MGPCG with a tolerance of 10^{-4}.

for each method, as depicted in Figures 5.17 and 5.18. Here, a lower rank number corresponds to a greater reduction of iteration counts. The strategies are sorted by their average ranks over all methods.

Using the closest stage value as initial guess yields the smallest iteration counts for all methods at both tolerances. For the higher tolerance of 10^{-4}, the absolute change achieved with the closest stage strategy compared to using zeros varies from -1.52 to -0.61 and the relative from -39.8% to -21.2%. At the low tolerance (10^{-7}), the absolute difference lies between -1.75 and -0.49 and the relative between -25.1% and -10.0%. The strategy ranking second-best varies between the two tolerances. At 10^{-7}, it is the R approximation at the time step used for the update of Q^N, which ranks second for about half the methods and third for the other half. This strategy is only the third-best at 10^{-4}. For this tolerance, using the result of the same stage of the previous time step is in second place, while being third at the higher tolerance. In both cases, the approximation of the derivative of Q is the fourth-best. The other two strategies and the initialization with zeros do not manage to reach a rank of four or higher for any method and tolerance. This also shows that the R approximation for the time step computation of P^N is a worse approximation for R_i than the one for Q^N.

The error values are barely impacted by the initial guesses here (not shown). At a tolerance of 10^{-7}, they are almost identical and at 10^{-4}, the values only vary slightly for the methods with small errors.

These variations appear to be random and not correlated to the iteration counts or their reduction. They are likely the result of natural variations within the tolerance of the CG method.
The fact that there are hardly any good approximations for the derivatives of R available prevents more advanced initial guessing strategies. Consistently, the best results are obtained by utilizing values of previously computed stages that are close in time. This strategy has the added benefit of not requiring additional calculations but requires more storage than a strategy based on time step values. For further investigations, interpolation between multiple nearby stage values may be a good start.

5.4.2 DIRK-2-0 initial guesses

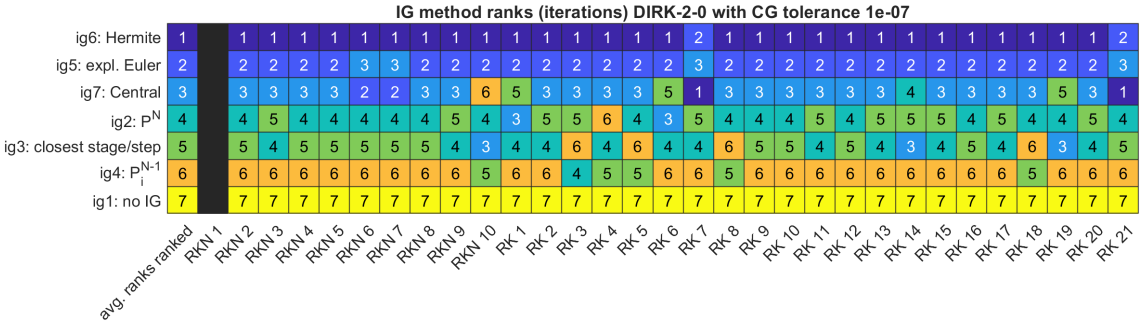

Figure 5.19: Initial guess strategies ranked according to average iteration counts for each method (1: lowest iteration count, lower is better). The rows are reordered according to average rank. The data is based on simulation using DIRK-2-0 and MGPCG with a tolerance of 10^{-7}.

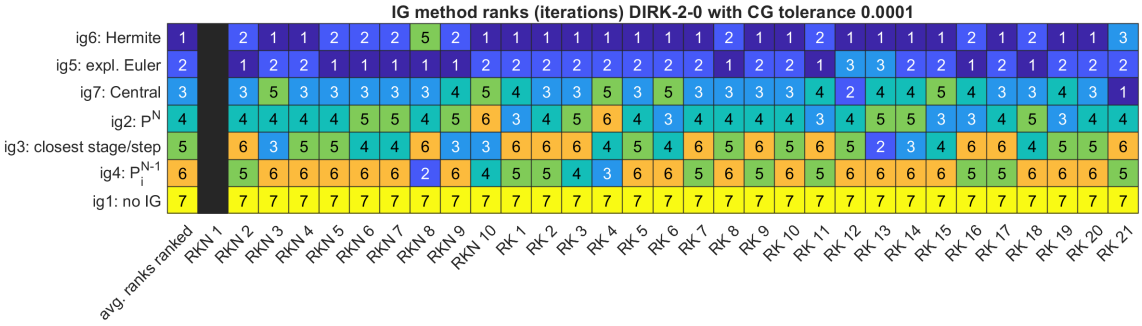

Figure 5.20: Initial guess strategies ranked according to average iteration counts for each method (1: lowest iteration count, lower is better). The rows are reordered according to average rank. The data is based on simulation using DIRK-2-0 and MGPCG with a tolerance of 10^{-4}.

For DIRK-2-0 at a tolerance of 10^{-4} the range of the change in iteration counts of all methods reaches from -2.22 to -0.24, which corresponds to relative changes between -58.7% and -9.2%. At 10^{-7} the values lie between -3.27 and -0.09 iterations, or between -60.6% and -1.9%. The ranks of the initial guess strategies for each method are given in Figures 5.19 and 5.20. The average ranks of the strategies are the same at both tolerances. They are, in the order of smallest to greatest iteration counts: Hermite interpolation, forward Euler method, central differences (as defined in Section 4.5),

using P^N, closest step or stage result, using the result of the same stage of the previous time step, and using zeros as the consistently worst option.
While the ranks of the strategies vary between methods at 10^{-7}, they are less mixed than at the higher tolerance of 10^{-4}. In the former case, Hermite interpolation is the best option for all but two methods. For the latter, it is the best option for all except one third where the forward Euler method is the best option. But a further comparison of the methods is difficult since the absolute and relative iteration reductions vary greatly between methods.
Hermite interpolation reduces the iterations by amounts between -3.27 and -1.95 or by -60.6% to -25.0% at 10^{-7} and between -2.22 and -0.78 or by -58.7% to -19.1% at 10^{-4}.
The third-best initial guesses are ones based on central differences. This is surprising since it only has a theoretical basis for methods that have stages that are symmetrical in time, meaning $\forall i = 1, ..., m : c_i \overset{!}{=} c_{m+1-i}$. This is only the case for RKN 6, 7, 9 and RK 7, 8, 21. For these, the strategy tends to produce better ranks than for the others, but not in all cases.

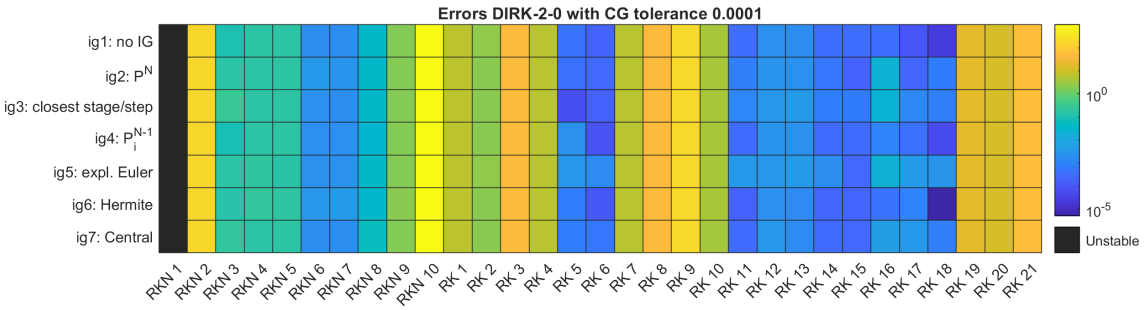

Figure 5.21: Errors of different initial guesses with DIRK-2-0 and MGPCG at a tolerance of 10^{-4}

The errors of the simulations with a CG tolerance of 10^{-4} are visualized in Figure 5.21. Similar to the results for DIRKN, the errors are almost identical compared to using zero as initial guess when the low tolerance is used and some variations start to occur at higher tolerances for the more accurate methods. These differences are larger than for DIRKN and affect more methods. This hints at a slight error amplification by DIRK-2-0 compared to DIRKN. In the equivalent plot for DIRKN with the tolerance of 10^{-4} (not shown), no error differences are visible between the different initial guess strategies.

5.4.3 DIRK-2-0s initial guesses

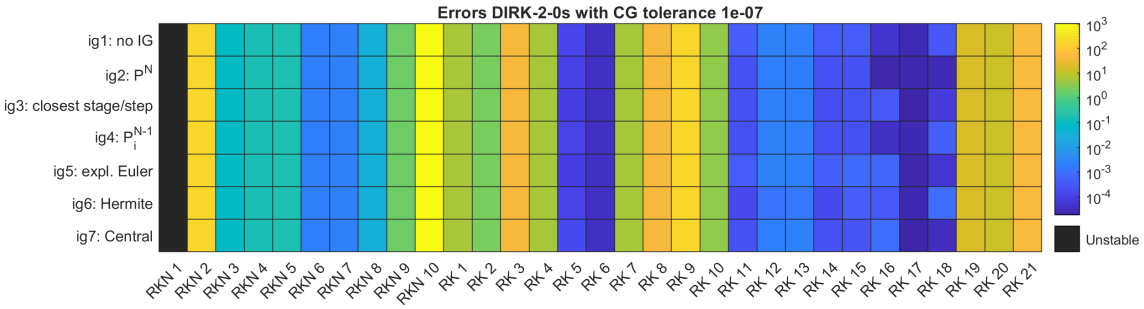

Figure 5.22: Errors of different initial guesses with DIRK-2-0s and MGPCG at a tolerance of 10^{-7}

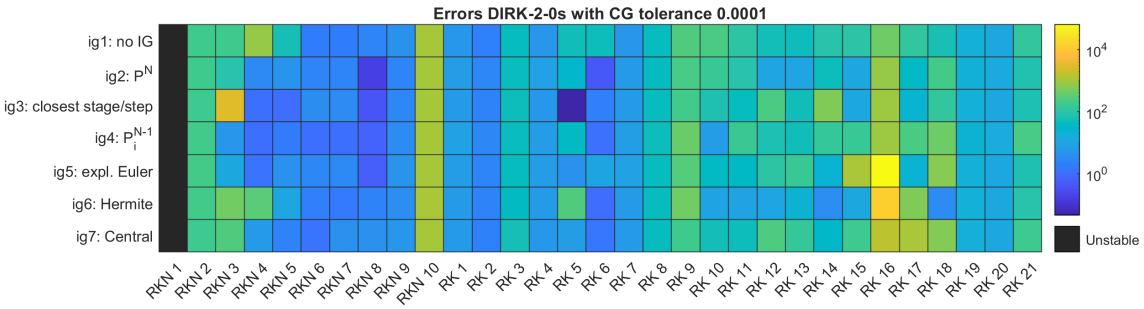

Figure 5.23: Errors of different initial guesses with DIRK-2-0s and MGPCG at a tolerance of 10^{-4}

For DIRK-2-0s, small differences in accuracy start to occur even at a tolerance of 10^{-7} for some methods, as Figure 5.22 shows. At 10^{-4}, the errors do not follow the same pattern for the different methods anymore, are much larger, and vary greatly between the different initial guesses. This is shown in Figure 5.23. This means that one would have to take the error differences between the initial guess strategies into account when evaluating their performance. The added complexity of this approach is beyond the scope of this analysis. Additionally, the large errors make using DIRK-2-0s with higher tolerances unattractive in general. Therefore, we only consider the results of the simulations with the lower tolerance of 10^{-7}.

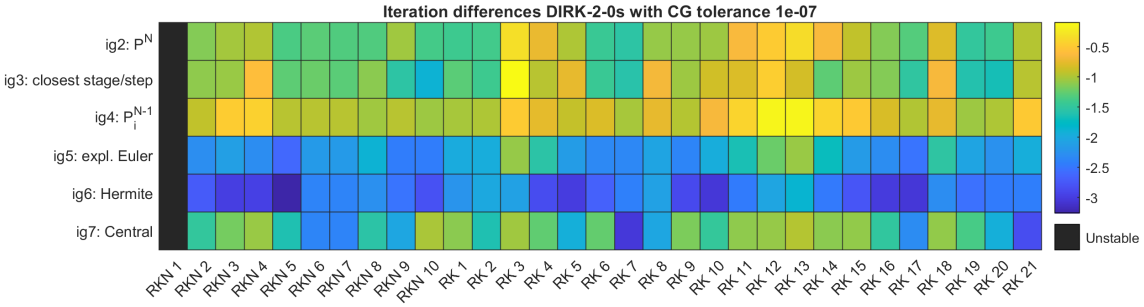

Figure 5.24: Average iteration count changes over all time steps of the respective initial guess strategies compared to using zero. The data is based on simulations using DIRK-2-0s and MGPCG with a tolerance of 10^{-7}.

Figure 5.24 visualizes the amount the MGPCG iteration numbers change when using the respective initial guesses compared to using zero. In all scenarios, using an initial guess strategy improves the iteration counts. The absolute amounts vary between -3.27 and -0.09 and the relative reductions between -60.3% and -1.9%.
The ranks of the initial guess strategies for each method, which are shown in Figure 5.25, are mostly incidental to those of DIRK-2-0 with a tolerance of 10^{-7}. Hermite interpolation changes the iterations by amounts between -3.27 and -1.86 or by -60.3% to -25.0% relative to using zeros. This is similar to the DIRK-2-0 results. These similarities are likely the result of the linear systems, left- and right-hand side, being computed with identical equations in both schemes.

Overall, one can see that for DIRK-2-0 and DIRK-2-0s simply using a result close to the next evaluation is not as effective as trying to predict it with extrapolation or explicit time-stepping techniques. Therefore, the more effective strategies are also the more expensive ones. But in the

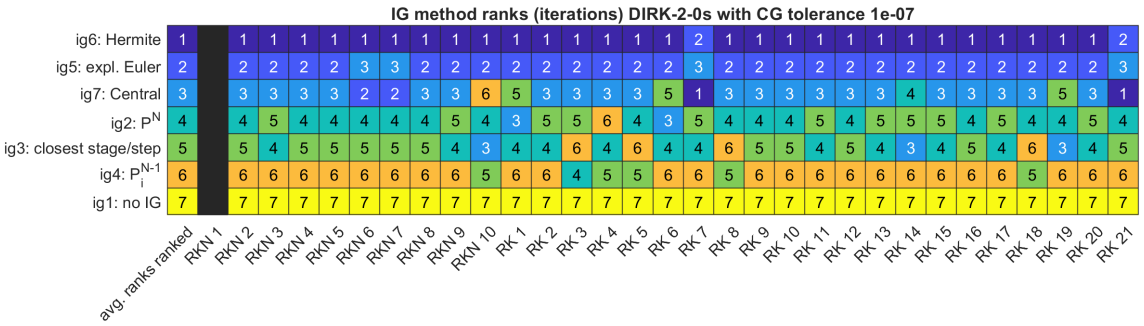

Figure 5.25: Initial guess strategies ranked according to average iteration counts for each method (1: lowest iteration count, lower is better). The rows are reordered according to average rank. The data is based on simulation using DIRK-2-0s and MGPCG with a tolerance of 10^{-7}.

light of the high PCG iteration costs of the examined strategies, this hardly matters. The iteration reductions using Hermite interpolation of at least 1.86 in the case of DIRK-2-0s and 1.95 in the case of DIRK-2-0 are well above the break-even boundary of 0.1 iterations.

5.4.4 Concluding remarks on initial guess strategies

Overall, it is more difficult to derive good initial guesses for R, the second derivative of P, compared to for P itself, as in for DIRK-2-0(s). The reduction in iterations for DIRKN is significantly smaller compared to those for the other two, even with the best strategies. Because of the similarity of DIRK-2-0 and DIRK-2-0s, the different initial guesses perform alike in both cases.
The best strategies for DIRKN simply use previously computed values for R, preferable to those close in time. Any attempts of using predictions with orders higher than zero produce worse results here. For DIRK-2-0(s) the contrary is true. Hermite interpolation and the forward Euler method work best, while simply using previously computed values leads to worse results. They are still better than initializing with zeros, but worse than the higher-order strategies.

Depending on the tolerance, the impact of initial guesses on the accuracy of the methods also has to be considered. If the CG tolerance is chosen low enough, the errors are independent of the initial guesses. Larger ones create a range within which the linear system solutions can lie and therefore also create an admissible range for the overall error. Initial guesses change the values that are assumed in this range. But it does not appear to be the case that certain strategies consistently improve the accuracy in these cases. Therefore the accuracy of results produced with larger tolerances has a certain random component. This is especially true for DIRK-2-0s, where the errors of the conjugate gradient method are amplified the most. The location of the threshold where this range becomes significant depends on the used method and scheme. DIRKN seems to be the most robust regarding high tolerances, followed by DIRK-2-0. Of the different methods, the range starts to appear earlier for those that are of higher orders and more accurate. For less accurate methods, these variations only become significant at tolerances where they reach an order of magnitude similar to the overall error.

The absolute reduction of the computational effort needed to solve a linear system by initial guesses is similar at all tolerances. But the relative reduction is highest for high tolerances. This means that effort invested in finding good initial guess strategies leads to smaller rewards if high accuracy is desired.

5.5 Impact of the Galerkin condition on iteration counts

The motivations for and against using the Galerkin condition are outlined in Section 4.4.3. It also explains the possibility of only applying it to either the matrix M or K, which constitute the linear system matrix $(M - \Delta t^2 a_{ii}^2 K)$. These different options are also referred to as coarsening strategies. In this section, we test the different options to analyze their impact on the errors and iteration counts. Note that all previous results use the Galerkin condition.
We only consider a tolerance of 10^{-7} to limit the artifacts stemming from the error amplification of DIRK-2-0s. Additionally to the multigrid preconditioner, the previously determined best initial guesses are used. These are the closest stage result for DIRKN and Hermite interpolation for DIRK-2-0 and DIRK-2-0s.

The error values are hardly impacted by the different coarsening strategies and therefore not shown here. Only some slight variations occur for the higher-order, more accurate methods. This is likely related to the admissible range given by the CG tolerance and therefore similar to the previously mentioned impact of the different initial guesses on the accuracy.

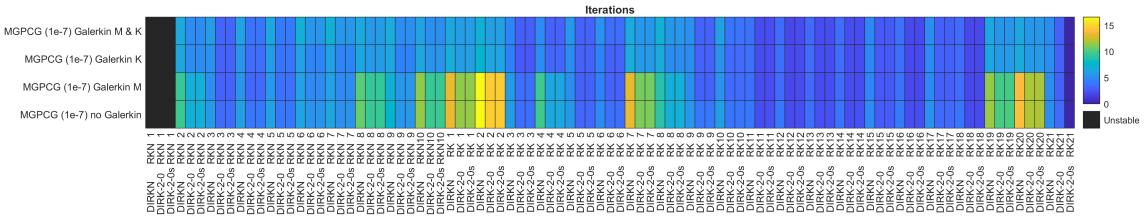

Figure 5.26: Iteration counts for all combinations of methods and time integration schemes calculated using MGPCG with a tolerance of 10^{-7} and the four different coarsening strategies

Figure 5.26 visualizes the iteration counts of all methods and schemes for the four different ways to apply, or not apply, the Galerkin condition. We see that for approximately two-thirds of the methods the iterations are not affected much by differences in coarsening strategies. But for the other methods, there is a significant increase in iteration numbers for all three time integration schemes if the Galerkin condition is not used on the finite difference matrix K. Their values up to 2.3 times larger. The differences between using the Galerkin condition for both M and K and using it only for K are negligible. The same is true for the differences between using it only for M and not using it at all. Thus, only not using the Galerkin condition on the finite difference matrix K has significant negative effects for some methods. The changes are often larger for DIRKN than for the other two schemes. But since it seems to require more iterations in all scenarios, the relative difference is similar in most cases. There are also configurations where the iteration counts are reduced when the Galerkin condition is not used. But these effects are small compared to the other increases.

A potential reason why this only occurs for the matrix K is that not applying the Galerkin condition to K changes the relationship between the points. This ignores the changes in the finite-difference stencil, which would otherwise occur. These are fill-ins, e.g. a 2D second-order 5-point stencil becomes a 9-point, and smaller differences between the coefficients. Not applying the Galerkin condition to M is equivalent to simply using a worse method for averaging the wave speeds on the coarse grids. While this may provide starting points for potential further investigations, it does not explain why this increase only happens for some methods.
The iteration count increases for most methods are large enough to prevent them from being competitive with the others. Table 4.4 of the previous chapter shows the cost of one MGPCG iteration with and without Galerkin condition for the 2D, second-order case. Their ratio is $\frac{87}{95}$. Therefore, the iteration count using Galerkin condition must be larger than $\frac{87}{95} \approx 0.915789$ times the iteration count not using it, for the latter case to be more efficient. The configurations which do not become less efficient when the Galerkin condition is dropped are given in Figure 5.27.
Note that for implementations using the Galerkin condition for K , storing the coarse matrices is not necessary. The coarse stencil can be determined analytically – at least for the periodic boundary conditions, restriction, and prolongation operator used here.

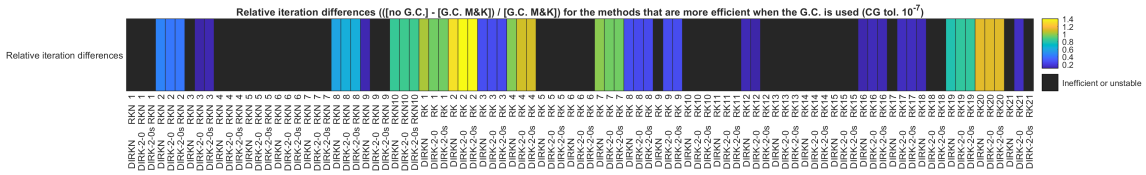

Figure 5.27: Relative iteration differences (values without Galerkin condition (M & K) minus the ones with it (M & K), divided by the ones with it) of configurations where the Galerkin condition for both M and K is more efficient computed with a CG tolerance of 10^{-7}. The blacked-out configurations are more efficient when using the Galerkin condition at least for K (or in the case of RKN1 unstable).

5.6 Comparison of overall efficiency

In the previous experiments, different parameters for the time integration of the acoustic wave equation are examined. In this section, the goal is to identify the configurations that are the most efficient using the knowledge of the other experiments. Efficiency refers to the computational cost (FLOPS) a configuration requires to produce results of a certain accuracy. Furthermore, an explicit method is employed as a reference point and provides insight into whether explicit and implicit methods using the presented approach can achieve similar efficiency. Note Section 4.4.4 already shows that it is unrealistic for an implicit method to outperform an explicit one for media with lower wave speed contrasts (see Table 4.5). Therefore, we consider the contrast to level the playing field. Equation 4.36 can be used to determine the contrast at which an explicit and an implicit method have equal costs. But this formula depends on the cost of the used implicit time integration scheme, which depends on empirical parameters. Furthermore, it does not consider the accuracy of both methods. This is why the different configurations and explicit methods are compared based on experimental results here.
To represent the explicit time integration schemes, the popular 4-stage, 5th order ERKN method

from [HNW93] (p.285) is chosen. This method is not necessarily representative of other explicit methods and possibly not the best option for this application. A search for well-performing ERKN methods is beyond the scope of this work. But the comparisons with this method already allow some general conclusions.

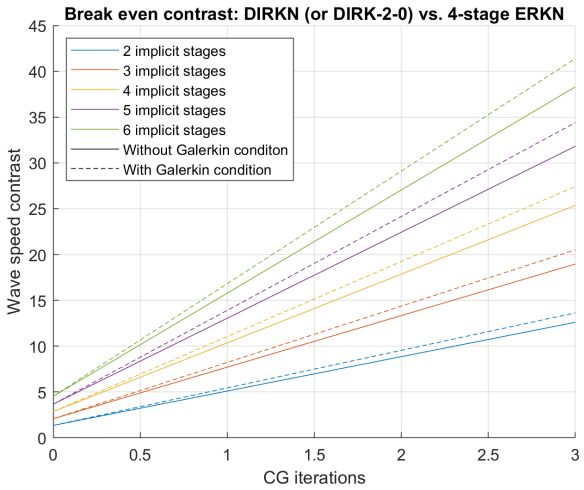

Figure 5.28: Wave speed contrasts over MGPCG iteration counts at which DIRKN requires the same amount of operations as a 4-stage ERKN method based on Equation 4.36. This contrast depends on the number of stages and whether the Galerkin condition is used. The values for DIRK-2-0 and DIRK-2-0s are identical or almost identical. If the contrast in a medium is higher at a fixed iteration count, implicit methods are more likely to outperform the ERKN method. Differences in accuracy are not taken into account.

For the implicit method, the same time step size as before with $c_f = \frac{1}{3}$ is used. While the results of varying time steps like in Figure 5.8 would be interesting, including this dimension would increase the complexity of this comparison drastically and is therefore omitted. Other variables influencing their costs, like the number of stages, iteration counts, and whether the Galerkin condition is used, vary between the methods. The required contrasts in all these different cases are given in Figure 5.28. It only shows the values for DIRKN and DIRK-2-0, but the values for DIRK-2-0s are almost identical because of the relatively high costs of MGPCG. The required contrast is proportional to the number of iterations. The corresponding slopes are proportional to the number of implicit stages since this determines the number of linear systems that have to be solved. When the Galerkin condition is used, the required contrast at a fixed number of iterations is also higher because of the higher MGPCG cost.

To increase the contrast, the wave speeds are scaled linearly with a constant minimal velocity so that the relationship between the spatial resolution, the source frequency, and the minimal guaranteed resolution N_{ppmw} is maintained (see Equation 4.27). This means that increasing the contrast increases all wave speeds except the minimum and thus reduces the realism of the simulated scenario. Therefore, we choose a contrast of 'only' 25, which is a big increase from the original 4.57, but only fulfills the theoretical boundary in scenarios with relatively few iterations or few stages. Note that the selection of methods does not include a method with six implicit stages and most of the 5-stage methods include explicit stages as well.

At the chosen contrast of 25, the ERKN method remains stable with a CFL factor of $C_{\text{CFL}} = 0.8$

during the 8s simulation time. For longer time spans or higher contrasts, smaller values may be required.
The previously determined best initial guess strategies are used for the implicit methods, namely the closest stages for DIRKN and Hermite interpolation for DIRK-2-0 and DIRK-2-0s. The Galerkin condition is used only for all combinations of methods and schemes where this is more efficient in the previous experiment (see Figure 5.27). To gain insight into the impact of changing CG tolerances, results are included of simulations with tolerances of $10^{-8}, 10^{-7}, ..., 10^{-1}$.

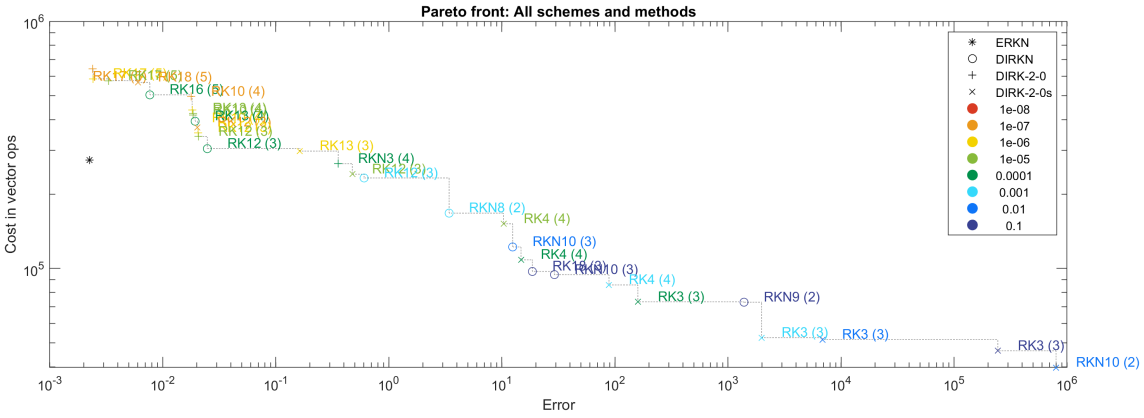

Figure 5.29: Pareto frontier of most efficient combinations of methods and time integration schemes. Markers refer to time integration schemes, colors to CG tolerances, labels to the methods, and the appended numbers in parentheses to their respective orders.

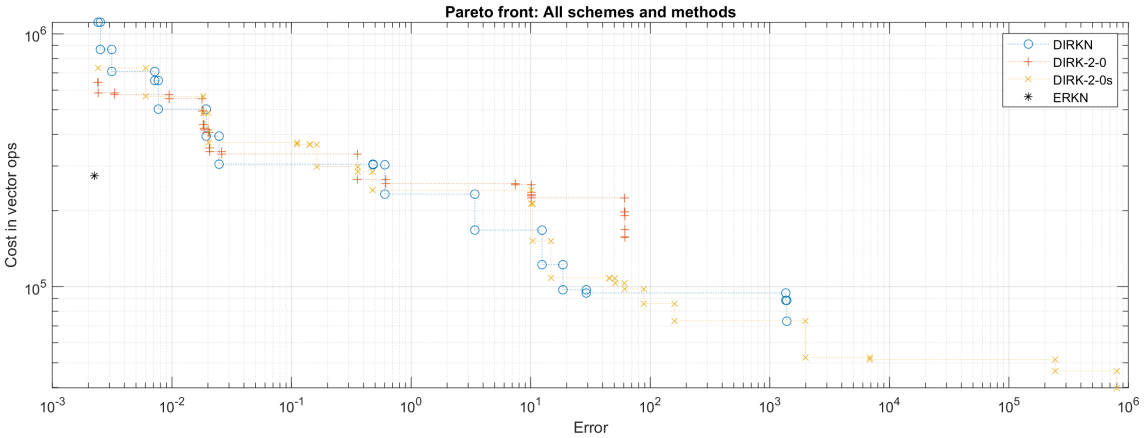

Figure 5.30: Separate Pareto frontiers for the three time integration schemes

One result is objectively better than another if it has lower costs or a lower error while the other variable remains fixed. The most efficient methods, therefore, form a Pareto frontier in the cost-error diagram, which is given in Figure 5.29. Each point has a marker corresponding to the used time integration scheme and color indicating the used CG tolerance. Additionally, they are labeled with the used method and the method's order in parentheses.
No configuration based on an implicit method manages to outperform the ERKN method. Some implicit results have similar errors and many have lower costs, but none come close to it in terms

of both cost and error. More on the comparison of explicit and implicit methods follows after an analysis of the implicit results.
The Pareto frontier of the implicit methods is not dominated by a single configuration but contains many different ones. Nevertheless, some trends are visible. At the lower end of the error spectrum, all three schemes have points on the frontier. For medium errors, DIRK-2-0 markers are not present. The upper end of the spectrum is entirely populated by DIRK-2-0s results.
The differences between the three time integration schemes can be understood better with the help of Figure 5.30. It shows individual Pareto frontiers for all three schemes. The range of errors which is covered by these curves varies. While the most accurate results are similar in terms of cost and error, only DIRK-2-0s produces cheap, less accurate results. The frontier of DIRK-2-0 has the smallest range in terms of both error and cost, while DIRKN lies in the middle. Note that the upper section of the error bandwidth is likely unattractive because of its large inaccuracy, even if it is cheaper to compute. In error ranges where all three have results, the fronts mostly lie close together. The only exception is the medium error range, where DIRK-2-0 becomes less efficient.
There is also a correlation between error size and CG tolerances in the Pareto front. Higher tolerances are favorable at larger errors and smaller ones at smaller errors. But the different tolerances still overlap widely. There are no results with a tolerance of 10^{-8}, which indicates that, for the used configurations, tolerances smaller than 10^{-7} do not improve accuracy or efficiency. If one has the choice between two similar-performing methods, one with a high and one with a low tolerance, the former is likely the worse choice since it is less reliable in its accuracy. This is because the admissible range of the CG results is larger and the accuracy within it can be considered as random.
There are also more higher-order methods present at lower errors and vice versa. But there are still many cases where methods of lower-order produce smaller errors than higher-order ones. This is consistent with the previous experiments, which showed that the accuracy of a method cannot be predicted well based on its algebraic order alone.
Many methods only have a single point on the front and if one has multiple, they are usually grouped relatively close together. This shows that there is no method that is the best option at all levels of accuracy.

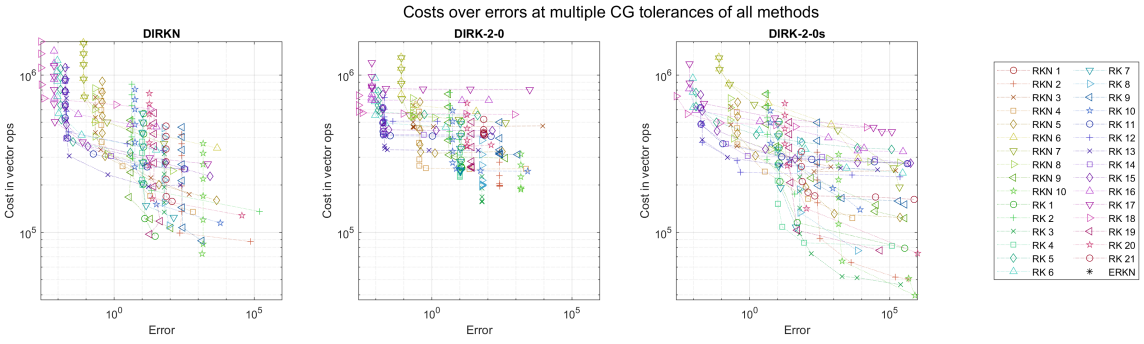

Figure 5.31: Costs of over errors at multiple CG tolerances (10^{-8}, 10^{-7}, ..., 10^{-1}) of all methods. Tolerance levels are not indicated, but higher cost or smaller errors for the same method indicate the use of lower tolerances in almost all cases.

We have already seen that lowering the CG tolerance only has beneficial effects up to a threshold of approximately 10^{-7}. Figure 5.31 provides further information on how changes in tolerance have an impact on the costs and errors. It contains the results of all runs with the points belonging to

the same method highlighted. The different tolerances are not indicated, but results of the same method with higher cost usually belong to lower tolerances. Here, we can see that increasing the tolerance starting from a low value usually first decreases the cost while the error stays the same. This means that error sources other than the linear system solver are dominating the overall error. Most methods reach a point where the error starts to increase, meaning the CG error becomes large enough to increase the overall error. The tolerance at which this happens varies greatly between methods and the schemes. Each method used with each scheme starts at similar error values for low tolerances. While the error stays consistent over many tolerances for DIRKN and DIRK-2-0, the error amplification of DIRK-2-0s makes it more sensitive to lower error tolerances, leading to error increases at lower tolerances.
Once DIRK-2-0 reaches the tolerance where the errors increase, the costs hardly decrease further. The other schemes usually still reduce the costs once the errors start increasing. This makes DIRK-2-0 less attractive for low accuracy uses.
The plots also show why no method dominates the Pareto front. The errors of all methods have individual lower bounds that are determined by factors other than the CG tolerance. When the tolerance is chosen low enough for the errors to increase, the cost reductions are usually too small for the method to be competitive with some other methods that are inherently less accurate. This leads to different methods being the most efficient at different error levels or operation counts.

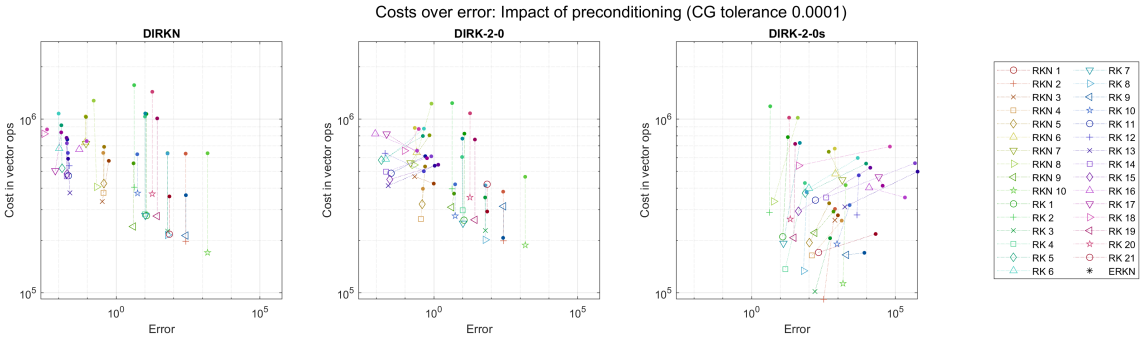

Figure 5.32: Impact of the preconditioner on costs and errors with all methods at a CG tolerance of 10^{-4}. The results using the preconditioner for all methods are identified based on their different markers and colors. The filled dots indicate the results without preconditioning and are connected to results using the preconditioner of the same method.

Figure 5.32 illustrates the effect of using the multigrid preconditioner on efficiency. For a fixed tolerance of 10^{-4} it shows the costs and errors of all methods and all schemes with and without preconditioning. The results that are produced without a preconditioner are marked with a filled dot, while the others have the same markers as previously used.
The changes when using the preconditioner vary in amount and kind. While almost all results improve significantly, there are also a few cases where the use of the preconditioner is detrimental to the result in either error or costs, but it is relatively rare. Many configurations, especially those using DIRKN, remain at a similar error value but reduce their costs significantly. But many also improve their error values additionally to improving their costs. This is more common for the more accurate results of DIRK-2-0 and the less accurate results of DIRK-2-0s. Increases of cost or error also occur, but are rare and often improve the other value.

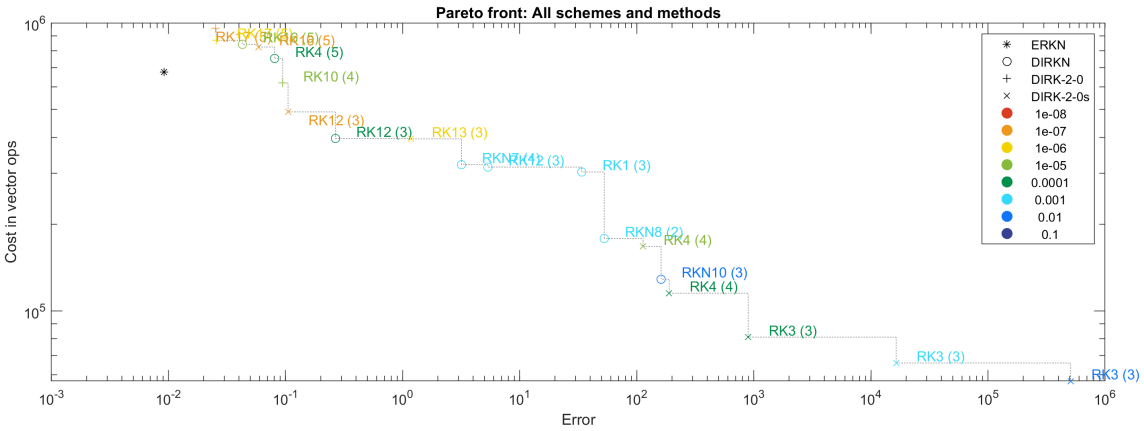

Figure 5.33: Pareto front of the most efficient methods and schemes for a wave speed contrast of 50.

Since increasing the contrast to 25 does not cause the implicit methods to outperform the explicit reference method one might want to increase it even further. Figure 5.33 shows the Pareto front and ERKN result for a contrast of 50. This requires a reduction of the CFL constant to 0.65. The implicit methods still cannot match the combination of cost and accuracy of the ERKN method. While its cost has increased relative to the Pareto front, it has also gotten more accurate.
One would predict EKRN to outperform the implicit time integration schemes at higher contrasts because this forces smaller time steps for explicit methods, while the ones of the implicit schemes remain unaffected. But increasing the contrast and therefore the wave speed does affect implicit methods as well. While they do not become unstable, their accuracy still diminishes. This might be related to the occurrence of frequencies higher than the source's, which are no longer represented accurately. It can therefore be necessary to decrease the time step size below the desired resolution to maintain accuracy.
Furthermore, increasing the contrast and therefore the wave speed also increases the cost of implicit methods. The increase is not as big as for the explicit method, but still noticeable. It stems from increases in the required CG iterations, which are likely caused by larger changes between time steps. In return, this means that decreasing the implicit time step size does not increase the cost proportional to the number of time steps. This leads to smaller differences between time steps and fewer required CG iterations.
The smaller time step size of explicit methods causes them to be more accurate than the competition. While increasing the contrast or decreasing the time step sizes to maintain stability for longer simulation times increases its cost, it also increases its accuracy.

It could possible to construct scenarios where some of the used implicit time integration schemes outperform explicit ones, but the experiments of this section suggest that the efficiency of explicit time integration methods should not be underestimated even in scenarios with high contrasts.

6 Conclusion and ideas for future work

6.1 Summary and conclusion

This work examines diagonally implicit Runge-Kutta-based approaches to the implicit time integration of the acoustic wave equation.
For the spatial discretization, finite differences are used but this approach is compatible with other discretization methods as well.
The time steps of explicit integration schemes are limited by the CFL condition. This can make them computationally expensive, especially for high wave speed contrasts in the simulated domain. For A-stable Runge-Kutta methods, which are necessarily implicit, on the other hand, the time step size is not limited in a similar way. Diagonally implicit Runge-Kutta (DIRK) methods allow A-stability while being much less computationally demanding than fully implicit methods. A direct application of Runge-Kutta methods to the acoustic wave equation in first-order form is inefficient because of the resulting large linear systems. A better alternative is to derive second-order integration schemes by combining and rearranging the original Runge-Kutta formulas. The most popular way of doing this is the diagonally implicit Runge-Kutta Nyström (DIRKN) scheme.
In this work, we derive two alternative second-order diagonally implicit time integration schemes, which are called DIRK-2-1 and DIRK-2-0. They can also be applied to differential equations other than the acoustic wave equation, but may not have the discussed properties for them. The new schemes are similar to the diagonally implicit Runge-Kutta Nyström (DIRKN) scheme, which can also be derived from the DIRK equations. The difference between the schemes is the variable they solve their systems of equations for. DIRK-2-0 solves them for the desired variable itself, DIRK-2-1 for the first temporal derivative, and DIRKN (or DIRK-2-2) for the second temporal derivative. It should be noted that Runge-Kutta Nyström methods include not only rearranged Runge-Kutta methods but are technically generalizations of them. The same is likely to be true for the newly derived schemes as well. One of the new time integration schemes, DIRK-2-1, is computationally and storage-wise too expensive to compete with the others and therefore not considered in our analysis. The more promising one, called DIRK-2-0, can be rearranged further in order to substitute matrix-vector multiplications. This has the potential to decrease the required operation per time step. This rearranged scheme is called DIRK-2-0s and is compared with DIRKN and DIRK-2-0 in this work.
In order for the implicit schemes to be competitive relative to explicit methods, the occurring linear systems must be solved as efficiently as possible. To that end, we use the conjugate gradient method (CG) with a multigrid preconditioner (MGPCG) and initial guess strategies. This approach can reduce the costs significantly, but introduces many parameters that need to be adjusted to optimize performance. We focus on singly diagonally implicit Runge-Kutta and Runge-Kutta-Nyström coefficients, meaning that the diagonal elements in the butcher tableau are all identical, but additionally allow explicit stages. These methods are well-suited for our use case because they yield linear systems with identical system matrices at all Runge-Kutta stages. This simplifies the solving

process. These matrices are also identical between the three different integration schemes.
Complexity analysis is used to quantify the costs of the different time integration schemes and the linear system solver cost. The schemes DIRKN and DIRK-2-0 have identical costs. The comparison with DIRK-2-0s is more complicated because its costs do not directly depend on the number of finite-difference stencil points like the ones of the other two do. Whether it is more efficient than the others at identical linear system solver costs depends on the number of stencil points and Runge-Kutta stages. DIRK-2-0s becomes more competitive for larger numbers of stencil points, as well as for fewer stages at lower stencil point counts.
However, the linear system solving costs are much larger than those of the time integration schemes themselves and therefore dominate the overall cost. They are highly dependent on the number of Runge-Kutta stages, the number of dimensions, the used finite difference order, the multigrid setup, and especially the amount of CG iterations required to achieve the desired tolerance. This dependency necessitates the use of numerical data for a proper comparison of different schemes and methods.

For the numerical experiments, a down-sampled version of the popular Marmousi2 model provides the wave speeds in the domain. It is a realistic geological model that contains many layers and multiple faults. Its wave speed contrast is 4.57. The waves are generated by a Ricker wavelet with a frequency of $0.73Hz$ at a single point source. Periodic boundary conditions are used because of their simple implementation. The errors are calculated based on the signal recorded at a single virtual receiver over the 8s simulation time. The time steps are chosen based on a specified temporal resolution of the waves. Ten DIRKN and 21 RKN sets of coefficients belonging to unconditionally stable methods are used and their results are compared against each other.

Because the three time integration schemes are derived from the same equations, they produce identical results when used with the same coefficients, and the linear systems are solved exactly. But there are large differences in the accuracy between the different methods. Although methods of higher algebraic order tend to produce lower errors than such with lower orders, one cannot predict the performance of a method based on its order alone. Multiple second-order methods outperform multiple third-order ones and other third-order methods outperform fourth-order ones. If there is a determining factor for the performance of the methods, we could not identify it reliably from the tested subset. The low number of published highly stable singly diagonally implicit Runge-Kutta (-Nyström) methods in the literature, especially with additional properties, makes the search for such factors difficult.

We used DIRKN coefficients with DIRK-2-0 and DIRK-2-0s and obtained good results with them. This might be considered surprising because DIRK-2-0 and DIRK-2-0s are derived from Runge-Kutta formulas and DIRKN coefficients usually cannot be converted into valid Runge-Kutta coefficients. Proving that DIRKN coefficients are compatible with DIRK-2-0(s) and deriving order conditions for DIRK-2-0(s) would be a logical continuation of this project.

Methods that are not at least A- or R-stable seem to be not stable enough for this application and are therefore not included in the selection of methods for the experiments. Whether A- and R-stability are sufficient stability criteria for our application in general is also not clear. One allegedly R-stable DIRKN method proved to be unstable in our experiments. These are also aspects that could be investigated further.

When the linear systems are solved with inexact solvers like CG, the three schemes perform differently. Compared to DIRKN, DIRK-2-0 performs similarly, but DIRK-2-0s amplifies the error of the linear system solver significantly. This amplification is likely because DIRK-2-0s utilizes the results of all previous stages which provides the possibility of error buildup over the stages. Because of this, DIRK-2-0s often requires lower CG tolerances to achieve a comparable level of accuracy. This can offset theoretical advantages in efficiency.

The multigrid preconditioner reduces the required CG iterations significantly and therefore reduces the overall cost in most cases. Especially for DIRK-2-0 and DIRK-2-0s, it also reduces the errors significantly in some cases. There are also combinations of integration schemes and methods where the preconditioner increases the cost or error, but these occurrences are relatively rare.

The use of initial guess strategies has the potential to reduce the iteration counts further. Because of the high MGPCG iteration cost, even more-complex strategies only have to reduce the iterations slightly to improve the overall operation counts. Since the linear systems are solved for the acoustic pressure in DIRKN and for its Laplacian in DIRK-2-0 and DIRK-2-0s, they require different initial guess strategies. For DIRKN, we obtain the best results by using the values of the previously computed stage result that are closest in time. For DIRK-2-0(s), Hermite interpolation (or extrapolation) using the last two computed time step results produces the highest iteration reductions among the examined strategies. A forward Euler step from the latest time step result also works well. For further improvements, interpolation between stage results seems promising for DIRKN and higher-order explicit time stepping schemes for DIRK-2-0(s).

Not using the Galerkin condition in the multigrid preconditioner can reduce the MGPCG iteration cost significantly. This does not lead to significant changes in error values and in many cases, the iteration counts also remain unaffected. But for some methods, the preconditioner loses its effectiveness, which makes the use of the Galerkin condition more efficient in terms of overall computational costs. What causes the absence of the Galerkin condition to affect only some methods is not obvious and could be of interest in further investigations.

The question of which combination of integration scheme, method, and CG tolerance works best is not trivial to answer. When plotting their costs over their respective errors, they form a Pareto front. Depending on the desired level of accuracy, different combinations are the best option. The best performing configurations of all three time integration schemes lie relatively close together in most cases. None of them are inherently less accurate or more expensive than the other. But differences between them still exist. In the experiments, DIRK-2-0s was the only method producing low-cost results with correspondingly large errors. DIRK-2-0 on the other hand only has points on the Pareto frontier at lower errors.
High tolerances tend to be more desirable for lower accuracies, and lower tolerances for high accuracy simulations, but the overlaps are relatively large. It should be noted though that high tolerances of 10^{-3} and 10^{-4} already produce good results for most use-cases and tolerances smaller than 10^{-7} only increase cost without improving accuracy.
Higher-order methods are necessary for high accuracy demands, but besides that, it is hard to make predictions.
The experiments show no well-performing general-purpose method. Instead, the methods usually only have points close to or on the Pareto front in a narrow error range. Which is the best choice at a certain level of accuracy must likely be determined experimentally.

To establish a reference point for their performance we use a popular fifth-order ERKN method. Even with all the discussed improvements, none of the implicit methods get close to the combined cost and error value of the explicit one even when the wave speed contrast is raised three-fold. Larger wave speed contrasts reduce the maximum allowable time step size for explicit methods, which is given by the CFL condition. This increases the cost of explicit methods and therefore should make the implicit ones more competitive because their time step sizes are not directly limited by the wave speed contrast. But creating scenarios in which implicit methods outperform the explicit competition is not as simple as increasing the contrast. While finding implicit methods that are better than some explicit methods in terms of cost or accuracy is often easy, the efficiency of explicit methods is difficult to match. Smaller time steps automatically lead to higher accuracy for explicit methods. If the implicit time steps become much larger than the CFL condition allows for explicit methods, the accuracy of implicit methods additionally diminishes. This makes it unrealistic for implicit methods based on the presented approach to compete with explicit alternatives in many scenarios.
The situation might be different for more accurate finite difference methods or its alternatives outlined in Section 1.3.1, but this is a topic left for future investigations.

6.2 Ideas for future work

As we have seen, the presented approach is not efficient enough to compete with explicit methods yet. But multiple areas for possible improvement come to mind.
The largest portion of the computational cost is caused by the linear system solver. While the conjugate gradient method itself does not leave room for optimization, the multigrid preconditioner can probably be improved. As a smoother, we use the damped Jacobi method, but the Gauss-Seidel method should theoretically provide better results at the same operation count. While it requires much more CPU time in the implementation used here, this is likely not the case for others.
Additionally, one could experiment with asymmetric multigrid preconditioning, especially with just one pre- or post-smoothing iteration. Table 4.2 shows that reducing the overall smoothing iterations from four to two reduces the cost significantly and preliminary experiments also showed no large differences in performance. Switching to just one iteration would decrease this cost further. A potential problem here is that multigrid as an operator should be symmetric when used with the conjugate gradient method. But [BDK15] for example still achieve good results with this approach in other applications.

The experiments have demonstrated the potential of initial guesses to reduce the overall complexity. The use of more sophisticated methods could reduce the required number of MGPCG iterations further. Because one iteration is quite expensive, even more-complicated methods, such as explicit Runge-Kutta methods for DIRK-2-0(s), should be considered. Interesting in this context is also the m-step Runge-Kutta method used by [YWL12] and its variant used with linear multistep methods in [YWD10]. The method is used in the former paper to predict Runge-Kutta stage values, which would be similar to DIRKN. In the latter paper, it is used to approximate time step values, which is similar to approximating P_i in DIRK-2-0(s). Note that these methods are used by the authors to avoid linear systems altogether and not as initial guess strategies. Instead, the approximations are used directly in the Runge-Kutta time step equations. This converts the time integration scheme they are used with into an explicit method. Yang and colleagues seem to obtain good results using this approach. While this is similar to using an explicit Runge-Kutta method from the start, using

the m-step Runge-Kutta methods apparently can lead to better stability properties [YWL12].
A possible intermediary approach would be using explicit methods to only approximate the implicit dependencies in the stage equations, which would turn it into a predictor-corrector scheme. This is equivalent to using the value of an initial guess strategy as a final result and omitting the MGPCG solver. How these approaches perform would have to be tested experimentally.

In the literature overview of Section 1.3, finite difference methods and splitting algorithms leading to tridiagonal linear systems are mentioned. It would be interesting to see if using such approaches could decrease computational costs.

While a discussion of numerical dispersion is not included in this work, reducing it also has the potential to further improve the presented approach. Using the standard finite difference formulas with the coarse spatial resolution of the discussed experiments likely produces much dispersion. We attempt to circumvent this problem by computing the reference solutions on the same grid. Using other spatial discretization methods that effectively suppress numerical dispersion like NADM (see Section 1.3.1) could increase the accuracy and potentially allow coarser grids in space and time without detrimental effects on the accuracy. This especially means that we could choose a smaller resolution parameter N_{ppmw}, which leads to larger spatial resolutions Δx and therefore decreases the overall computational cost. But note that this applies to explicit and implicit methods equally. Whether the improvements in accuracy could also support larger time steps for implicit methods, meaning larger values for the factor c_f in our equations, would have to be tested.

While switching the discretization method to one with less numerical dispersion would likely have a positive effect, using spatially implicit finite difference methods should also be considered. The authors of [CS12] show that if the time integration is implicit it can be possible to implement spatially implicit finite differences without additional costs. This has the potential of reducing dispersion and can also be combined with methods that only require the solving of tridiagonal linear systems.

Of these suggestions, the predictor-corrector approach and its comparison to the approaches of [YWD10; YWL12] are especially attractive because they are simple to implement and promise large cost reductions. A combination with a more accurate spatial discretization method (like NADM) should be considered as well. Whether this approach could lead to methods that can compete with explicit methods in a wide range of scenarios is impossible to predict without experimental data.

A Eigenvalues of the evolution matrix

The following proof was adopted from unpublished notes of Longfei Gao [Gao19] with minor modifications.

Neglecting the source term, the semi-discrete acoustic wave equation in first-order form (see Equation 2.7) can be written in the following block form

$$\frac{d}{dt}\begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} 0 & M^{-1} \\ K & 0 \end{bmatrix}\begin{bmatrix} P \\ Q \end{bmatrix}, \tag{A.1}$$

where 0 stands for zero matrix blocks of the appropriate size. We will refer to the block matrix as the evolution matrix of the system since its properties characterize the solution evolution. Here, we demonstrate that all eigenvalues of the evolution matrix are on the imaginary axis.

Since the following similarity transformation holds,

$$\begin{bmatrix} I & 0 \\ 0 & M^{-1} \end{bmatrix}\begin{bmatrix} 0 & M^{-1} \\ K & 0 \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} = \begin{bmatrix} 0 & I \\ M^{-1}K & 0 \end{bmatrix},$$

we have that $\begin{bmatrix} 0 & M^{-1} \\ K & 0 \end{bmatrix}$ and $\begin{bmatrix} 0 & I \\ M^{-1}K & 0 \end{bmatrix}$ share the same eigenvalues. Denoting $M^{-1}K$ as B, next we consider the relationship between the eigen-pairs of matrix B and that of matrix $\begin{bmatrix} 0 & I \\ B & 0 \end{bmatrix}$.

Suppose vector $\begin{bmatrix} u^T & w^T \end{bmatrix}^T$ is an eigenvector of $\begin{bmatrix} 0 & I \\ B & 0 \end{bmatrix}$ corresponding to eigenvalue α, we have

$$\begin{bmatrix} 0 & I \\ B & 0 \end{bmatrix}\begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} w \\ Bu \end{bmatrix} = \begin{bmatrix} \alpha u \\ \alpha w \end{bmatrix}.$$

From the first block row, we have relation $w = \alpha u$ and from the second block row, we now have $Bu = \alpha w = \alpha^2 u$. In other words, $\{\alpha^2; u\}$ is an eigen-pair of B. The matrix M is diagonal with positive entries and K is symmetric negative semi-definite and thus only has non-positive real eigenvalues. The matrix $B = M^{-1}K$ also only has non-positive real eigenvalues because it is similar to $M^{-\frac{1}{2}}KM^{-\frac{1}{2}}$, which is also symmetric non-positive definite. Therefore, α can only be purely imaginary numbers (which include 0).

Bibliography

[ACM06] I. Alonso-Mallo, B. Cano, M. Moreta. “Stability of Runge–Kutta–Nyström methods”. In: *Journal of Computational and Applied Mathematics* 189.1-2 (May 2006), pp. 120–131. DOI: 10.1016/j.cam.2005.01.005 (cit. on pp. 18, 57, 58).

[AKB74] R. M. Alford, K. R. Kelly, D. M. Boore. “Accuracy of finite-difference modeling of the acoustic wave equation”. In: *GEOPHYSICS* 39.6 (Dec. 1974), pp. 834–842. ISSN: 0016-8033. DOI: 10.1190/1.1440470 (cit. on p. 8).

[Ale77] R. Alexander. “Diagonally Implicit Runge–Kutta Methods for Stiff O.D.E.’s”. In: *SIAM Journal on Numerical Analysis* 14.6 (Dec. 1977), pp. 1006–1021. DOI: 10.1137/0714068 (cit. on p. 58).

[AM+80] A. Alekseev, B. Mikhailenko, et al. “The solution of dynamic problems of elastic wave propagation in inhomogeneous media by a combination of partial separation of variables and finite-difference methods”. In: *Journal of geophysics* 48.1 (1980), pp. 161–172 (cit. on p. 10).

[BC83a] D. C. Booth, S. Crampin. “The anisotropic reflectivity technique: anomalous reflected arrivals from an anisotropic upper mantle”. In: *Geophys J Int* 72.3 (Mar. 1983), pp. 767–782. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.1983.tb02832.x (cit. on p. 10).

[BC83b] D. C. Booth, S. Crampin. “The anisotropic reflectivity technique: theory”. In: *Geophysical Journal of the Royal Astronomical Society* 72.3 (Mar. 1983), pp. 755–766. ISSN: 0016-8009. DOI: 10.1111/j.1365-246X.1983.tb02831.x (cit. on p. 10).

[BDK15] H. Bouwmeester, A. Dougherty, A. V. Knyazev. “Nonsymmetric Preconditioning for Conjugate Gradient and Steepest Descent Methods 1”. In: *Procedia Computer Science* 51 (2015), pp. 276–285. DOI: 10.1016/j.procs.2015.05.241 (cit. on pp. 26, 84).

[Bou96] M. Bouchon. “The discrete wave number formulation of boundary integral equations and boundary element methods: A review with applications to the simulation of seismic wave propagation in complex geological structures”. In: *pure and applied geophysics* 148.1 (1996), pp. 3–20. ISSN: 1420-9136. DOI: 10.1007/BF00882052 (cit. on p. 10).

[BR97] J. O. Blanch, J. O. A. Robertsson. “A modified Lax-Wendroff correction for wave propagation in media described by Zener elements”. In: *Geophys J Int* 131.2 (Nov. 1997), pp. 381–386. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.1997.tb01229.x (cit. on p. 8).

[Car94] J. M. Carcione. “The wave equation in generalized coordinates”. In: *Geophysics* 59.12 (1994), pp. 1911–1919 (cit. on p. 9).

[CFL67] R. Courant, K. Friedrichs, H. Lewy. “On the Partial Difference Equations of Mathematical Physics”. In: *IBM Journal of Research and Development* 11.2 (Mar. 1967), pp. 215–234. DOI: 10.1147/rd.112.0215 (cit. on p. 48).

[CH99] J. Carcione, H. Helle. “Note: Numerical Solution of the Poroviscoelastic Wave Equation on a Staggered Mesh”. In: *Journal of Computational Physics* 154 (1999), pp. 520–527 (cit. on p. 8).

[Cha78] C. H. Chapman. “A new method for computing synthetic seismograms”. In: *Geophys J Int* 54.3 (Sept. 1978), pp. 481–518. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.1978.tb05491.x (cit. on p. 10).

[Che93] X. Chen. “A systematic and efficient method of computing normal modes for multilayered half-space”. In: *Geophysical Journal International* 115.2 (Nov. 1993), pp. 391–409. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.1993.tb01194.x (cit. on p. 10).

[Con93] N. h. Cong. “A-stable diagonally implicit Runge-Kutta-Nyström methods for parallel computers”. In: *Numerical Algorithms* 4.2 (1993), pp. 263–281. ISSN: 1572-9265. DOI: 10.1007/BF02144107 (cit. on p. 17).

[CS11] C. Chu, P. L. Stoffa. “Derivation and numerical analysis of implicit time stepping schemes”. In: SEG Technical Program Expanded Abstracts 0. Society of Exploration Geophysicists, Jan. 2011, pp. 2987–2991. DOI: 10.1190/1.3627815. URL: https://doi.org/10.1190/1.3627815 (cit. on p. 13).

[CS12] C. Chu, P. L. Stoffa. “Implicit finite-difference simulations of seismic wave propagation”. In: *Geophysics* 77.2 (Feb. 2012), T57–T67. ISSN: 0016-8033. DOI: 10.1190/geo2011-0180.1 (cit. on pp. 13, 14, 85).

[CS79] G. J. Cooper, A. Sayfy. “Semiexplicit A-stable Runge-Kutta methods”. In: *Mathematics of Computation* 33.146 (May 1979), pp. 541–541. DOI: 10.1090/s0025-5718-1979-0521275-1 (cit. on p. 58).

[Dab86] M. A. Dablain. “The application of high-order differencing to the scalar wave equation”. In: *GEOPHYSICS* 51.1 (Jan. 1986), pp. 54–66. ISSN: 0016-8033. DOI: 10.1190/1.1442040 (cit. on p. 8).

[Dah63] G. G. Dahlquist. “A special stability problem for linear multistep methods”. In: *BIT Numerical Mathematics* (1963) (cit. on p. 18).

[Dah78] G. Dahlquist. “On accuracy and unconditional stability of linear multistep methods for second order differential equations”. In: *BIT Numerical Mathematics* 18.2 (1978), pp. 133–136. ISSN: 1572-9125. DOI: 10.1007/BF01931689 (cit. on p. 17).

[DK06] M. Dumbser, M. Käser. “An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes – II. The three-dimensional isotropic case”. In: *Geophys J Int* 167.1 (Oct. 2006), pp. 319–336. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.2006.03120.x (cit. on p. 9).

[EJ91] K. Eriksson, C. Johnson. “Adaptive Finite Element Methods for Parabolic Problems I: A Linear Model Problem”. In: *SIAM Journal on Numerical Analysis* 28.1 (1991), pp. 43–77. ISSN: 0036-1429. URL: http://www.jstor.org/stable/2157933 (cit. on p. 9).

[ESS82] S. H. Emerman, W. Schmidt, R. A. Stephen. “An implicit finite-difference formulation of the elastic wave equation”. In: *GEOPHYSICS* 47.11 (Nov. 1982), pp. 1521–1526. DOI: 10.1190/1.1441302 (cit. on pp. 8, 10, 11, 13).

[FGR01] J. M. Franco, I. Gómez, L. Rández. “Four-stage symplectic and P-stable SDIRKN methods with dispersion of high order”. In: *Numerical Algorithms* 26.4 (2001), pp. 347–363. ISSN: 1572-9265. DOI: 10.1023/A:1016629706668 (cit. on p. 58).

[FGR97] J. Franco, I. Gómez, L. Rández. “SDIRK methods for stiff ODEs with oscillating solutions”. In: *Journal of Computational and Applied Mathematics* 81.2 (July 1997), pp. 197–209. DOI: 10.1016/s0377-0427(97)00056-3 (cit. on p. 58).

[Gao19] L. Gao. “Unpublished notes and oral communication”. Cooperation during an Internship at KAUST, KSA. 2019 (cit. on pp. 7, 17, 30, 32, 49, 56, 87).

[GDCK19] L. Gao, D. C. Del Rey Fernández, M. Carpenter, D. Keyes. “SBP–SAT finite difference discretization of acoustic wave equations on staggered block-wise uniform grids”. In: *Journal of Computational and Applied Mathematics* 348 (2019), pp. 421–444. ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2018.08.040. URL: https://www.sciencedirect.com/science/article/pii/S0377042718305272 (cit. on p. 8).

[GMK08] M. Galis, P. Moczo, J. Kristek. “A 3-D hybrid finite-difference—finite-element viscoelastic modelling of seismic wave motion”. In: *Geophysical Journal International* 175.1 (2008), pp. 153–184 (cit. on p. 10).

[GSS06] M. J. Grote, A. Schneebeli, D. Schötzau. “Discontinuous Galerkin Finite Element Method for the Wave Equation”. In: *SIAM J. Numer. Anal.* 44.6 (Jan. 2006), pp. 2408–2431. ISSN: 0036-1429. DOI: 10.1137/05063194x (cit. on p. 9).

[Hel68] D. V. Helmberger. “The crust-mantle transition in the Bering Sea”. In: *Bulletin of the Seismological Society of America* 58.1 (Feb. 1968), pp. 179–214. ISSN: 0037-1106 (cit. on p. 10).

[HH99] K. Honglu, X. Hexi. “Differentiator series solution of linear differential ordinary equation”. In: *Applied Mathematics and Mechanics* 20.8 (1999), pp. 880–887 (cit. on p. 15).

[HNW93] E. Hairer, S. Norsett, G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Vol. 8. Jan. 1993. DOI: 10.1007/978-3-540-78862-1 (cit. on pp. 14, 18, 23, 75).

[HS89a] P. J. V. der Houwen, B. P. Sommeijer. “Diagonally Implicit Runge–Kutta–Nyström Methods for Oscillatory Problems”. In: *SIAM Journal on Numerical Analysis* 26.2 (Apr. 1989), pp. 414–429. DOI: 10.1137/0726023 (cit. on pp. 18, 58).

[HS89b] P. J. V. der Houwen, B. P. Sommeijer. “Phase-Lag Analysis of Implicit Runge-Kutta Methods”. In: *SIAM Journal on Numerical Analysis* 26.1 (Feb. 1989), pp. 214–229. DOI: 10.1137/0726012 (cit. on p. 58).

[HW96] E. Hairer, G. Wanner. *Solving Ordinary Differential Equations II*. Springer Berlin Heidelberg, 1996. DOI: 10.1007/978-3-642-05221-7 (cit. on p. 17).

[HYM20] X. He, D. Yang, X. Ma. “A Weighted Runge-Kutta Discontinuous Galerkin Method for 3D Acoustic and Elastic Wave-Field Modeling”. In: *COMMUNICATIONS IN COMPUTATIONAL PHYSICS* 28.1 (2020), pp. 372–400 (cit. on pp. 9, 15, 16).

[HYMQ20] X. He, D. Yang, X. Ma, C. Qiu. "A modified numerical-flux-based discontinuous Galerkin method for 2D wave propagations in isotropic and anisotropic media". In: *Geophysics* 85.5 (Sept. 2020), T257–T273. ISSN: 0016-8033. DOI: 10.1190/geo2019-0485.1 (cit. on p. 9).

[HYW15] X. He, D. Yang, H. Wu. "A weighted Runge-Kutta discontinuous Galerkin method for wavefield modelling". In: *Geophys J Int* 200.3 (Mar. 2015), pp. 1389–1410. ISSN: 0956-540X. DOI: 10.1093/gji/ggu487 (cit. on pp. 8, 9, 15, 16).

[IIP16] R. Itzá, U. Iturrarán-Viveros, J. O. Parra. "Optimal implicit 2-D finite differences to model wave propagation in poroelastic media". In: *Geophys J Int* 206.2 (Aug. 2016), pp. 1111–1125. ISSN: 0956-540X. DOI: 10.1093/gji/ggw180 (cit. on p. 17).

[IMR95] H. Igel, P. Mora, B. Riollet. "Anisotropic wave propagation through finite-difference grids". In: *GEOPHYSICS* 60.4 (July 1995), pp. 1203–1216. ISSN: 0016-8033. DOI: 10.1190/1.1443849 (cit. on p. 8).

[KB82] D. D. Kosloff, E. Baysal. "Forward modeling by a Fourier method". In: *GEOPHYSICS* 47.10 (Oct. 1982), pp. 1402–1412. ISSN: 0016-8033. DOI: 10.1190/1.1441288 (cit. on p. 9).

[KC16] C. A. Kennedy, M. H. Carpenter. "Diagonally implicit Runge-Kutta methods for ordinary differential equations". In: *A Review. NASA Report. Langley research center. Hampton VA* 23681 (2016), p. 162 (cit. on pp. 58, 63).

[KD06] M. Käser, M. Dumbser. "An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes – I. The two-dimensional isotropic case with external source terms". In: *Geophys J Int* 166.2 (Aug. 2006), pp. 855–877. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.2006.03051.x (cit. on p. 9).

[KHI94] Y. Kondoh, Y. Hosaka, K. Ishii. "Kernel optimum nearly-analytical discretization (KOND) algorithm applied to parabolic and hyperbolic equations". In: *Computers & Mathematics with Applications* 27.3 (1994), pp. 59–90. ISSN: 0898-1221. URL: https://www.sciencedirect.com/science/article/pii/0898122194900477 (cit. on p. 9).

[KK59] F. C. Karal Jr, J. B. Keller. "Elastic wave propagation in homogeneous and inhomogeneous media". In: *The Journal of the acoustical society of america* 31.6 (1959), pp. 694–705 (cit. on p. 10).

[KL07] S. Kim, H. Lim. "High-order schemes for acoustic waveform simulation". In: *Applied Numerical Mathematics* 57.4 (2007), pp. 402–414. ISSN: 0168-9274. DOI: https://doi.org/10.1016/j.apnum.2006.05.003. URL: https://www.sciencedirect.com/science/article/pii/S0168927406001012 (cit. on p. 11).

[KL73] R. L. Kuhlemeyer, J. Lysmer. "Finite element method accuracy for wave propagation problems". In: *Journal of Soil Mechanics & Foundations Div* 99.Tech Rpt (1973) (cit. on p. 9).

[KPE08] D. Kosloff, R. Pestana, H. T. Ezer. *Numerical Solution of the Constant Density Acoustic Wave Equation By Implicit Spatial Derivative Operators*. Nov. 2008 (cit. on p. 14).

[KPT10] D. Kosloff, R. C. Pestana, H. Tal-Ezer. “Acoustic and elastic numerical wave simulations by recursive spatial derivative operators”. In: *GEOPHYSICS* 75.6 (Oct. 2010), T167–T174. ISSN: 0016-8033. DOI: 10.1190/1.3485217 (cit. on p. 14).

[KT99] D. Komatitsch, J. Tromp. “Introduction to the spectral element method for three-dimensional seismic wave propagation”. In: *Geophys J Int* 139.3 (Dec. 1999), pp. 806–822. ISSN: 0956-540X. URL: https://doi.org/10.1046/j.1365-246x.1999.00967.x (cit. on p. 9).

[KV98] D. Komatitsch, J.-P. Vilotte. “The Spectral Element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures”. In: *Bulletin of the Seismological Society of America* 88 (Apr. 1998), pp. 368–392 (cit. on p. 9).

[KWTA76] K. R. Kelly, R. W. Ward, S. Treitel, R. M. Alford. “Synthetic seismograms; a finite-difference approach”. In: *Geophysics* 41.1 (Feb. 1976), pp. 2–27. ISSN: 0016-8033. DOI: 10.1190/1.1440605 (cit. on p. 8).

[LDB72] J. Lysmer, L. A. Drake, A. Bruce. “A Finite Element Method for Seismology”. In: *Methods in Computational Physics: Advances in Research and Applications*. Vol. 11. Elsevier, 1972, pp. 181–216. URL: https://www.sciencedirect.com/science/article/pii/B978012460811550009X (cit. on p. 9).

[LS09] Y. Liu, M. K. Sen. “A practical implicit finite-difference method: examples from seismic modelling”. In: *J Geophys Eng* 6.3 (Sept. 2009), pp. 231–249. ISSN: 1742-2132. DOI: 10.1088/1742-2132/6/3/003 (cit. on pp. 8, 14, 17).

[LSJ+09] Y. Liu, M. K. Sen, K. Jackson, et al. “Advanced finite-difference methods for seismic modeling”. In: *Geohorizons* 14.2 (2009), pp. 5–16 (cit. on p. 8).

[LW64] P. D. Lax, B. Wendroff. “Difference schemes for hyperbolic equations with high order of accuracy”. In: *Communications on pure and applied mathematics* 17.3 (1964), pp. 381–398 (cit. on p. 8).

[MAL04] S. Ma, R. J. Archuleta, P. Liu. “Hybrid modeling of elastic P-SV wave motion: A combined finite-element and staggered-grid finite-difference approach”. In: *Bulletin of the Seismological Society of America* 94.4 (2004), pp. 1557–1563 (cit. on p. 10).

[Mar84] K. J. Marfurt. “Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations”. In: *GEOPHYSICS* 49.5 (May 1984), pp. 533–549. ISSN: 0016-8033. DOI: 10.1190/1.1441689 (cit. on p. 9).

[Muf85] I. R. Mufti. “Seismic modeling in the implicit mode”. In: *Geophysical Prospecting* 33.5 (Aug. 1985), pp. 619–656. DOI: 10.1111/j.1365-2478.1985.tb00770.x (cit. on pp. 8, 10, 11, 13).

[MWM06] G. S. Martin, R. Wiley, K. J. Marfurt. “Marmousi2: An elastic upgrade for Marmousi”. In: *The Leading Edge* 25.2 (Feb. 2006), pp. 156–166. ISSN: 1070-485X. DOI: 10.1190/1.2172306 (cit. on p. 55).

[Nys26] E. J. Nyström. *Über die numerische Integration von Differentialgleichungen*. Deutsch. Acta Societatis Scientiarum Fennicae ; Tom. 50, N:o 13. Erscheinungsjahr 1926 vom Gesamttitelblatt. Helsingfors: Druckerei der Finn. Literaturges., 1926, 55 Seiten (cit. on pp. 18, 23).

[PR74] A. Prothero, A. Robinson. “On the Stability and Accuracy of One-Step Methods for Solving Stiff Systems of Ordinary Differential Equations”. In: *Mathematics of Computation* 28.125 (1974), pp. 145–162. ISSN: 00255718, 10886842. DOI: 10.2307/2005822. URL: http://www.jstor.org/stable/2005822 (cit. on p. 17).

[QZ92] M. Z. Qin, M. Q. Zhang. “Symplectic Runge-Kutta algorithmz for Hamilton systems”. In: *Journal of Computational Mathematics, Supplementary Issue, pp. 205–215* (1992) (cit. on p. 58).

[RBS94] J. O. A. Robertsson, J. O. Blanch, W. W. Symes. “Viscoelastic finite-difference modeling”. In: *GEOPHYSICS* 59.9 (Sept. 1994), pp. 1444–1456. ISSN: 0016-8033. DOI: 10.1190/1.1443701 (cit. on p. 8).

[RL19] Z. Ren, Z. Li. “High-order temporal and implicit spatial staggered-grid finite-difference operators for modelling seismic wave propagation”. In: *Geophys J Int* 217.2 (May 2019), pp. 844–865. ISSN: 0956-540X. DOI: 10.1093/gji/ggz059 (cit. on p. 14).

[RR97] D. Ristow, T. Rühl. “3-D implicit finite-difference migration by multiway splitting”. In: *GEOPHYSICS* 62.2 (Mar. 1997), pp. 554–567. ISSN: 0016-8033. DOI: 10.1190/1.1444165. URL: https://doi.org/10.1190/1.1444165 (cit. on p. 17).

[RW03] B. Rivière, M. Wheeler. “Discontinuous finite element methods for acoustic and elastic wave problems”. In: *Contemporary Mathematics* 329 (Jan. 2003). ISSN: 9780821832615 (cit. on p. 9).

[SFB90] P. Sharp, J. Fine, K. Burrage. “Two-stage and Three-stage Diagonally Implicit Runge-Kutta Nyström Methods of Orders Three and Four”. In: *IMA Journal of Numerical Analysis* 10.4 (1990), pp. 489–504. DOI: 10.1093/imanum/10.4.489 (cit. on pp. 18, 58).

[Sha07] G. Shan. “Optimized implicit finite?difference migration for TTI media”. In: SEG Technical Program Expanded Abstracts 0. Society of Exploration Geophysicists, Jan. 2007, pp. 2290–2294. DOI: 10.1190/1.2792941. URL: https://doi.org/10.1190/1.2792941 (cit. on p. 17).

[Shu88] C.-W. Shu. “Total-variation-diminishing time discretizations”. In: *SIAM Journal on Scientific and Statistical Computing* 9.6 (1988), pp. 1073–1084 (cit. on p. 15).

[SO88] C.-W. Shu, S. Osher. “Efficient implementation of essentially non-oscillatory shock-capturing schemes”. In: *Journal of computational physics* 77.2 (1988), pp. 439–471 (cit. on p. 15).

[SP94] G. Seriani, E. Priolo. “Spectral element method for acoustic wave simulation in heterogeneous media”. In: *Finite Elements in Analysis and Design* 16.3 (1994), pp. 337–348. ISSN: 0168-874X. URL: https://www.sciencedirect.com/science/article/pii/0168874X94900760 (cit. on p. 9).

[Tar84] A. Tarantola. “Inversion of seismic reflection data in the acoustic approximation”. In: *GEOPHYSICS* 49.8 (Aug. 1984), pp. 1259–1266. ISSN: 0016-8033. DOI: 10.1190/1.1441754 (cit. on p. 7).

[TG00] N. Takeuchi, R. J. Geller. "Optimally accurate second order time-domain finite difference scheme for computing synthetic seismograms in 2-D and 3-D media". In: *Physics of the Earth and Planetary Interiors* 119.1 (2000), pp. 99–131. ISSN: 0031-9201. URL: https://www.sciencedirect.com/science/article/pii/S0031920199001557 (cit. on p. 8).

[VO09] J. Virieux, S. Operto. "An overview of full-waveform inversion in exploration geophysics". In: *Geophysics* 74.6 (2009), WCC1–WCC26 (cit. on p. 7).

[WL18] E. Wang, Y. Liu. "An implicit spatial and high-order temporal finite difference scheme for 2D acoustic modelling". In: *null* 49.2 (Apr. 2018), pp. 187–201. ISSN: 0812-3985. DOI: 10.1071/eg16094 (cit. on p. 14).

[WYL12] N. Wang, D. Yang, F. Liu. "Weak dispersion wave-field simulations: A predictor-corrector algorithm for solving acoustic and elastic wave equations". In: *Journal of Seismic Exploration* 21 (May 2012), pp. 125–152 (cit. on pp. 15, 16).

[WZ14] N. Wang, Y. Zhou. "A weak dispersion 3d wave field simulation method: A predictor-corrector method of the Implicit Runge-Kutta Scheme". In: *Journal of Seismic Exploration* 23 (Nov. 2014), pp. 431–462 (cit. on pp. 15, 16).

[YLWP04] D. Yang, M. Lu, R. Wu, J. Peng. "An Optimal Nearly Analytic Discrete Method for 2D Acoustic and Elastic Wave Equations". In: *Bulletin of the Seismological Society of America* 94.5 (Oct. 2004), pp. 1982–1992. ISSN: 0037-1106. DOI: 10.1785/012003155 (cit. on p. 9).

[YPLT06] D. Yang, J. Peng, M. Lu, T. Terlaky. "Optimal Nearly Analytic Discrete Approximation to the Scalar Wave Equation". In: *Bulletin of the Seismological Society of America* 96.3 (June 2006), pp. 1114–1130. ISSN: 0037-1106. DOI: 10.1785/0120050080 (cit. on p. 9).

[YTZL03] D. Yang, J. Teng, Z. Zhang, E. Liu. "A Nearly Analytic Discrete Method for Acoustic and Elastic Wave Equations in Anisotropic Media". In: *Bulletin of the Seismological Society of America* 93.2 (Apr. 2003), pp. 882–890. ISSN: 0037-1106. DOI: 10.1785/0120020125 (cit. on p. 9).

[YWCS09] D. Yang, N. Wang, S. Chen, G. Song. "An Explicit Method Based on the Implicit Runge-Kutta Algorithm for Solving Wave Equations". In: *Bulletin of the Seismological Society of America* 99.6 (Nov. 2009), pp. 3340–3354. DOI: 10.1785/0120080346 (cit. on pp. 15, 16).

[YWD10] D. Yang, L. Wang, X. Deng. "An explicit split-step algorithm of the implicit Adams method for solving 2-D acoustic and elastic wave equations". In: *Geophys J Int* 180.1 (Jan. 2010), pp. 291–310. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.2009.04407.x (cit. on pp. 15, 16, 84, 85).

[YWL12] D. Yang, N. Wang, E. Liu. "A strong stability-preserving predictor-corrector method for the simulation of elastic wave propagation in anisotropic media". In: *Communications in Computational Physics* 12.4 (2012), pp. 1006–1032 (cit. on pp. 14, 17, 51, 84, 85).

[YYL17] L. Yang, H. Yan, H. Liu. “An optimal implicit staggered-grid finite-difference scheme based on the modified Taylor-series expansion with minimax approximation method for elastic modeling”. In: *Journal of Applied Geophysics* 138 (2017), pp. 161–171. ISSN: 0926-9851. URL: https://www.sciencedirect.com/science/article/pii/S0926985116303998 (cit. on p. 14).

[ZC06] H. Zhou, X. Chen. “A new approach to simulate scattering of SH waves by an irregular topography”. In: *Geophys J Int* 164.2 (Feb. 2006), pp. 449–459. ISSN: 0956-540X. DOI: 10.1111/j.1365-246X.2005.02670.x (cit. on p. 10).

[ZC08] H. Zhou, X. Chen. “The localized boundary integral equation–discrete wavenumber method for simulating P-SV wave scattering by an irregular topography”. In: *Bulletin of the Seismological Society of America* 98.1 (2008), pp. 265–279 (cit. on p. 10).

[ZCC03] H.-M. Zhang, X.-F. Chen, S. Chang. “An Efficient Numerical Method for Computing Synthetic Seismograms for a Layered Half-space with Sources and Receivers at Close or Same Depths”. In: *Seismic Motion, Lithospheric Structures, Earthquake and Volcanic Sources: The Keiiti Aki Volume*. Ed. by Y. Ben-Zion. Basel: Birkhäuser Basel, 2003, pp. 467–486. DOI: 10.1007/978-3-0348-8010-7_3 (cit. on p. 10).

[Zha92] M.-Q. Zhang. “Diagonally implicit symplectic Runge-Kutta schemes for Hamiltonian systems”. In: *Proc. of Int. Conf. on Scientific Computation, Hangzhou,China, T. Chan and Z.-C. Shi, Eds., World Scientific, Singapore (1992) 259-262.* (1992) (cit. on p. 58).

[ZZS07] H. (Zhang, Y. Zhang, J. Sun. “Implicit splitting finite?difference scheme for multidimensional wave simulation”. In: SEG Technical Program Expanded Abstracts 0. Society of Exploration Geophysicists, Jan. 2007, pp. 2011–2015. DOI: 10.1190/1.2792885 (cit. on pp. 12, 13).

All links were last followed on April 25, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Waiblingen, 26.04.21, S. Mönchel

place, date, signature