A Fast Matrix-Free Algorithm for Spectral Approximations to High-Dimensional Partial Differential Equations

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Bernd Brumm

aus Mühlacker

Tübingen
2015

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Prüfung:	13.01.2016
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Christian Lubich
2. Berichterstatter:	Prof. Dr. Marlis Hochbruck

Abstract

This thesis is concerned with the computational intractabilities that arise from spectral discretizations of high-dimensional partial differential equations. Using the example of the time-dependent multi-particle Schrödinger equation, we consider a spectral Galerin approximation in space with a tensor product basis of Hermite functions. When propagating the resulting system of ordinary differential equations in time, one typically needs to evaluate matrix-vector products involving a matrix representation of the Hamiltonian operator in each time step. Since the size of this matrix equals the number of equations, which (for an unreduced basis) depends exponentially on the dimension and thus quickly becomes enormously large, this can make computations infeasible—both due to a lack of memory and due to unbearably long computation times.

We present a fast algorithm for an efficient computation of these matrix-vector products that scales only linearly with the size of the Galerkin basis—without assembling the matrix itself. Besides being a matrix-free approach, the fast algorithm is compatible with the idea of reducing the index set that underlies the basis. The computational speed-up is achieved using orthogonality of the Hermite functions in combination with their generic three-term recurrence. Briefly, these properties allow to compute the action of the matrix representations of the coordinatewise position operators on vectors in linear time. The basic idea is then to insert these coordinate matrices into a polynomial approximate of the potential. This has originally been proposed by E. Faou, V. Gradinaru, and Ch. Lubich in [23]. We modify their approach and turn it into a rigorous algorithm. For an unreduced set of basis functions, we show this tentative proceeding to be equivalent to a suitable entrywise approximation of the potential matrix by Gauß–Hermite quadrature. Reducing the index set for the basis functions yields an additional error.

We derive error estimates for all approximation steps involved. In particular, using a binary tree approach, bounds for the errors due to quadrature and index set reduction are deduced. Both errors decay spectrally if the potential is significantly smoother than the exact solution wherever the latter does not essentially vanish. Extensive numerical experiments corroborate these findings. Besides, we show performance tests comparing the fast algorithm to a matrix-free approach from the chemical literature.

Apart from the above basic form of the fast algorithm, we present applications of the general methodology to the nonlinear Schrödinger equation and, most prominently, to initial-boundary value problems. As an example, we study the acoustic wave equation with non-constant coefficients and Engquist–Majda boundary conditions, and construct efficient procedures for the different kinds of matrix-vector products together with an error analysis and numerical tests.

Deutsche Zusammenfassung

Das Lösen hochdimensionaler partieller Differentialgleichungen mittels Spektralmethoden stellt in Anbetracht der damit verbundenen enormen Speicherplatz- wie auch Zeitkomplexität eine besondere Herausforderung dar. Wir betrachten das Beispiel der zeitabhängigen Vielteilchen-Schrödingergleichung, die mit einem Galerkinansatz im Raum diskretisiert wird. Die zugrundeliegende Basis besteht aus Tensorprodukten von Hermitefunktionen. Bei der Integration des zugehörigen Systems gewöhnlicher Differentialgleichungen sind in jedem Zeitschritt typerischerweise Matrix-Vektor-Produkte mit der Darstellungsmatrix des Hamiltonoperators der Gleichung bezüglich der gewählten Galerkinbasis zu berechnen. Die Größe dieser Matrix hängt im Falle einer nicht ausgedünnten Basis exponentiell von der Dimension des Problems ab. Das explizite Aufstellen der Matrix überschreitet damit leicht den zur Verfügung stehenden Speicherplatz und erfordert unerträglich lange Rechenzeiten.

Diese Arbeit stellt einen schnellen Algorithmus zur Berechnung solcher Matrix-Vektor-Produkte vor, dessen Komplexität nur linear von der Größe der Galerkinbasis abhängt und der ohne explizites Aufstellen der Matrix auskommt. Darüber hinaus erlaubt er nahezu beliebiges Ausdünnen der Basis – sofern eine entsprechende Ausdünnung selbst eine gute Approximation an die gesuchte Lösung des Problems liefert. Stellt man die Ortsoperatoren bezüglich der einzelnen Koordinaten in der gewählten Basis dar, so lassen sich mithilfe der wechselseitigen Orthogonalität der Hermitefunktionen sowie der sie definierenden Drei-Term-Rekursion Produkte dieser Koordinatenmatrizen mit Vektoren in linearer Zeit berechnen. Die bereits von E. Faou, V. Gradinaru und Ch. Lubich in [23] vorgestellte Kernidee des schnellen Algorithmus besteht nun darin, die Koordinatenmatrizen formal in eine polynomielle Approximation des Potentials einzusetzen. Wir modifizieren diesen Vorschlag und präsentieren einen rigorosen Algorithmus. Für eine nicht ausgedünnte Galerkinbasis erweist sich diese Idee als äquivalent zur Approximation der Integrale in jedem Eintrag der Matrixdarstellung des Potentials mittels einer spezifisch gewählten Gauß-Hermite-Quadratur, wie in der vorliegenden Arbeit gezeigt wird. Ausdünnen der Basis generiert einen zusätzlichen Fehler.

Ein wichtiger Bestandteil dieser Arbeit ist die Fehleranalyse. Insbesondere lassen sich die durch Quadratur und ggf. Ausdünnen der Basis verursachten Fehler jeweils durch geschicktes Umschreiben der Hermite-Rekursion als Binärbaum kontrollieren. Unter der Annahme eines im Vergleich zur exakten Lösung hinreichend glatten Potentials fallen diese Fehler spektral ab. Dies wird in numerischen Experimenten bestätigt. Als Vergleichsmaß für die tatsächliche Einsparung an Rechenzeit durch den schnellen Algorithmus dient uns ein matrixfreier Ansatz, der in der chemischen Literatur entwickelt wurde.

Darüber hinaus übertragen wir den obigen Ansatz auf eine analoge Behandlung u.a. der nichtlinearen Schrödingergleichung und von Anfangsrandwertproblemen. Als Beispiel für

letztere Klasse betrachten wir im zweiten Teil der Arbeit die Wellengleichung mit variablen Koeffizienten und Engquist-Majda-Randbedingungen und konstruieren analoge effiziente Verfahren für die zugehörigen Matrix-Vektor-Produkte. Wir führen ebenfalls eine Fehleranalyse durch und präsentieren numerische Experimente.

Danksagung

Diese Arbeit wäre ohne die Ideen Christian Lubichs sowie ohne die durch ihn erfahrene Betreuung nicht denkbar. Für die mir angebotene Stelle als Mitarbeiter in seiner Gruppe, für seine mathematischen (und manchmal auch literarischen) Anstöße sowie für die durch ihn ermöglichten Konferenzteilnahmen und Reisen bin ich mehr als dankbar.

Mein Dank gilt meiner zweiten Gutachterin und Prüferin Marlis Hochbruck, nicht zuletzt für die freundliche Einladung nach Karlsruhe.

Sehr verbunden bin ich dem GRK 1838, dem ich als Doktorand assoziiert war und aus dessen Mitteln ein guter Teil meiner Reisen bestritten wurde. Namentlich erwähnt sei vor allem Stefan Teufel, der sich auch als Prüfer zur Verfügung gestellt hat, sowie die stets hilfreiche Stefanie Engstler.

Andreas Prohl möchte ich für seine Bereitschaft, als Prüfer zu fungieren, herzlich danken.

Bezahlt wurde ich aus Mitteln des DFG-Schwerpunktprogrammes 1324, wofür ich ebenfalls zu Dank verpflichtet bin.

Die in Kingston an der Queen's University als Gast in Tucker Carringtons Gruppe verbrachten Monate werden mir eine besondere Erinnerung bleiben. Tucker und Gustavo Ávila haben mir dort viel von ihrer Zeit gewidmet, so dass ich den Aufenthalt in Kanada als Horizonterweiterung erfahren durfte – fachlich und darüber hinaus.

Meinen Tübinger Kollegen, von denen mir viele zu Freunden geworden sind, danke ich für die gemeinsame Zeit und für den fachlichen (und natürlich auch für den außerfachlichen!) Austausch, insbesondere Thomas Dunst, Ludwig Gauckler, Markus Klein, Balázs Kovács, Dhia Mansour, Chris Power, Jonathan Seyrich, Hanna Walach und nicht zuletzt Daniel Weiß, von dem ich viel gelernt habe. Ausdrücklich danken möchte ich Balázs für sein wertvolles und sehr gründliches Lektorat.

Ich danke meinem zeitweiligen "room mate" Emil Kieri aus Uppsala für die produktive Zusammenarbeit, wie mir auch Nurcan Gücüyenen aus Izmir, Wencheng Li aus Xian und Toshiaki Itoh aus Kyoto als Zimmergenossen in freundschaftlicher Erinnerung bleiben werden.

Die mittägliche Kaffeerunde[1] am runden Tisch wäre nicht vollständig ohne unsere Diplomanden und Masterstudenten, insbesondere Anna, Raphi und Thy. Und die Arbeitsgruppe wäre nicht funktionsfähig ohne die Hilfe unserer bei Notfällen stets verlässlichen Computer-

[1] Dank anbei auch den Firmen Philips und Jacobs Douwe Egberts für die Entwicklung einer gewissen Portionskaffeemaschine mit Kaffeepads, welche die in unserer Arbeitsgruppe zahlreich durchgeführten Belastungstests allesamt mit Bravour bestanden hat!

administratoren. Ich werde den Umgang mit den vielen Studenten, denen ich in der Lehre begegnet bin, vermissen.

Meinen Eltern sowie meinem Bruder Jochen danke ich für die langjährige Unterstützung in allen Lebenslagen. Und meinem Lebenspartner Mirko, den ich sehr liebe.

Contents

Introduction

"Had we but world enough, and time"
– Andrew Marvell, *To His Coy Mistress*

In Andrew Marvell's famous mid-17th century poem that has the above words at its beginning, a poor soul laments a coy woman's slow and hesitant response to his courtship; see the commented edition [80]. By the constraint of a normal lifespan, so he mocks her while she keeps playing hard to get, she's putting the two of them at risk of dying of old age before finally giving in. Within the temporal bounds of human existence, solving partial differential equations (PDEs) can be no less an exercise in patience than winning a reluctant heart. For both endeavors, time (or the lack thereof, that is) can be the crucial issue. But since Marvell's "coy mistress" is a hopeless case, the present thesis is rather concerned with the computational intractabilities that arise from spectral discretizations in space of high-dimensional PDEs than with love's grief and impatience.

We start from a very brief characterization of spectral methods. For a given number of degrees of freedom, spectral methods exhibit a higher accuracy than finite element or finite difference methods. The reason is that they employ global basis functions of a high degree, whereas the latter methods are built from local low-degree approximations to the unknown solution. If the solution is infinitely differentiable, we get convergence rates faster than any polynomial in the number of basis functions; see [57, 83]. Finite elements or finite differences can be fitted to irregularly-shaped domains, while spectral methods are most useful when the underlying geometry is fairly regular. Unbounded domains, in particular, typically fall within the scope of spectral methods. On the downside, whereas finite elements or finite differences yield sparse matrices, the matrices due to spectral discretizations are in general dense. There is an abundant literature on spectral methods. To name just a few, see the handbook article [8] and the textbooks [10, 15, 26, 38, 47, 76, 88].

As an example well-suited both for a spectral ansatz in general and for an illustration of the computational challenges such an ansatz can bring about, consider the linear time-dependent Schrödinger equation over the whole $\mathbb{R}^d$,

$$i\psi_t(x,t) = H(x,t)\psi(x,t), \qquad x = (x_1, \ldots, x_d) \in \mathbb{R}^d,\ t \geq 0,$$

where the dimension d might become large. Using a spectral Galerkin approach in space, we search for an approximation

$$\psi(x,t) \approx \psi_{\mathcal{K}}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x)$$

that is a linear combination of $L^2(\mathbb{R}^d)$-orthonormal basis functions $\varphi_{\mathbf{k}}$ with multi-indices $\mathbf{k} = (k_1, \ldots, k_d)$ taken from a d-dimensional index set $\mathcal{K} \subset \mathbb{N}^d$, and we determine the

unknown coefficients $\mathbf{c} = (c_\mathbf{k})_{\mathbf{k}\in\mathcal{K}}$ by the requirement that the residual be in the orthogonal complement of the corresponding linear approximation space. Equivalently, this yields a system of $|\mathcal{K}|$ ordinary differential equations (ODEs),

$$i\dot{\mathbf{c}}(t) = \mathbf{H}(t)\mathbf{c}(t), \qquad \mathbf{H}_{\mathbf{jk}}(t) = \int_{\mathbb{R}^d} \bar{\varphi}_\mathbf{j}(x) H(x,t) \varphi_\mathbf{k}(x)\,\mathrm{d}x,$$

with a $(|\mathcal{K}| \times |\mathcal{K}|)$-matrix representation $\mathbf{H}$ of the Hamiltonian operator. Discretizing the latter system in time using, e.g., a polynomial integrator or approximations to the matrix exponential by a splitting method or a Magnus integrator all require the computation of matrix-vector products involving $\mathbf{H}$ in each time step. If the Galerkin basis consists of tensor products of univariate $L^2(\mathbb{R})$-orthonormal functions,

$$\varphi_\mathbf{k}(x) = \varphi_{k_1}(x_1) \cdot \ldots \cdot \varphi_{k_d}(x_d), \qquad \mathbf{k} \in \mathcal{K},$$

where each k_α ranges from 0 to some threshold K, the number of equations amounts to $(K+1)^d$. We write $\mathcal{K} = \mathcal{K}(d, K)$ and call this the full index cube. For a moderate choice of dimension, say $d = 4$, with a typical threshold of $K = 30$, storing all entries of the corresponding matrix $\mathbf{H}$ separately on a machine in double precision arithmetics (64 bits) requires an impressive amount of 6.2 TB of memory (symmetries and zeros not taken into account, though). Besides memory, computation time is equally severe an issue. For instance, on a desktop computer with an Intel Core 2 Duo E8400 3.00 GHz processor with 4 GB RAM using an implementation in `C`, the mere assembly takes more than 20 days. Cheaper than the assembly, but still unbearably expensive, is the computation of the product $\mathbf{Hv}$ for some vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$, which scales quadratically with the size of the Galerkin basis. The exponential dependency on the dimension thus proves to be enormously harmful, and we realize that, in our context, the poet's cry both for enough "world" and "time" is really a cry for more memory and faster computations. For this computational intractability, centuries after the coy mistress had passed away presumably untouched, the term *curse of dimensionality* has been coined; see [7].

This thesis is a contribution to cope with the infamous curse of dimensionality in the context of spectral discretizations of high-dimensional PDEs. It decomposes into two parts: In Part I, we consider the Schrödinger equation as maybe the most prominent example of such a problem. In Part II, we apply the methodology developed for the Schrödinger equation on an unbounded domain to the wave equation on a bounded domain. This introduction covers only the first part.

There are three established strategies to deal with the computational burden the above problem poses: First, one *reduces* the Galerkin basis and skips those indices from $\mathcal{K}$ that contribute the least to an acceptable approximation $\psi \approx \psi_\mathcal{K}$. If the index set can only be sufficiently reduced, this can speed up computations drastically. Second, instead of assembling $\mathbf{H}$ and then doing the matrix-vector product $\mathbf{Hv}$ explicitly, people try to compute the resulting vector directly, i.e., in a *matrix-free* way. Leaving aside the tremendous assembly costs for a moment, ideally, this reduces the quadratically scaling costs to a number of operations that scales only linearly with the size of the Galerkin basis. Whereas index set reduction techniques thus make the matrix smaller, but stick with quadratically scaling costs, matrix-free approaches mitigate the computational complexity by avoiding an assembly of $\mathbf{H}$ at all. Third, one replaces the matrix-vector product with the application of

some *fast transform* between coefficient and physical space for which linearly scaling procedures are available or can be devised. Since matrix-free approaches, as we use the word, are not transforms between different representation spaces, they are closely related to fast transforms, but conceptually different.

In the subsequent paragraphs, we shall briefly comment on all three techniques. As will become clear, these techniques are not mutually exclusive, and we shall indeed combine the first two strategies in our own approach.

Reduced index set techniques have recently gained a greater popularity, but the very idea can be traced back to the early 1960s. So-called *sparse grids* of nodes for high-dimensional quadrature are commonly attributed to S. Smolyak (see [81]), who introduced a tensorized construction based on a hierarchy of one-dimensional (1D) quadrature formulas to approximate integrals of multivariate functions over d-dimensional cubes and thus overcome the curse of dimensionality to some extent. An overview of subsequent applications of Smolyak's idea in various fields or for various problems is given in [14, 33] and, more recently, in [37]. Applying the technique with a spectral collocation approach for the solution of a PDE, one employs a *hyperbolic reduction* for the indices for the spectral basis and a sparse grid of collocation points, which are in a bijective correspondence. Drawing such a reduced index set in two dimensions yields a cross-like distribution. Sparse grid methods are therefore also known by the name of hyperbolic cross approximations; see [77, 84]. We shall comment on spectral collocation methods and their relation to spectral Galerkin methods in more detail in the introduction to Part II of this thesis. Since neither does our fast algorithm make use of any kind of quadrature nor do we consider a collocation ansatz, optimal choices of quadrature or collocation points taken from a reduced grid are not of interest for us, and we focus solely on a reduction of the index set for the Galerkin basis. But then, besides hyperbolic reductions, we consider a broad class of reduced index sets. *Additive reductions*, in particular, will play a role.

In computational chemistry, people have long been aware of the need for matrix-free approaches; see the methods and references given in [20], where matrices of the above kind are justifiably characterized as "monster matrices". Another valuable source of references is the review article [59]. Since we are addressing a mathematical audience, a comprehensive reception of the corresponding chemical literature is out of scope. *Sequential summations* is a matrix-free approach that has come up in the mid-1980s. It has since been put forward most prominently by T. Carrington and his coworkers, who, to the best of our knowledge, have contributed the most to its development. Roughly, if H is a sum of separable operators, a clever re-arrangement of the indices considerably economizes computations when evaluating the sums

$$(\mathbf{Hv})_{\mathbf{j}} = \sum_{\mathbf{k}\in\mathcal{K}} H_{\mathbf{jk}} v_{\mathbf{k}}, \qquad \mathbf{j} \in \mathcal{K}.$$

A starting point is the work [11], where a full index cube is considered; see also [19, 27, 62] for similar ideas. The approach has been generalized to an additively reduced basis in [90], both in a quadrature-free version and with a full tensor grid of quadrature nodes for an approximation of the integrals in the Galerkin matrix. In [2], an adaptation for modified Smolyak quadrature is presented, which has since been put to use in [3, 4, 5] and adapted in [6] to Smolyak interpolation in the context of a collocation approach. These latter references cover situations where our matrix-free approach is not generally applicable, though, while

the proceedings due to [11, 90] bear close resemblance to our own ideas. We shall extensively comment on sequential summations and their relation to the present work.

For the full index cube, fast algorithms for the discrete transforms between grid values and expansion coefficients such as the Fast Fourier Transform (for a periodic setting) or the Fast Cosine Transform (in case of a Chebyshev basis for problems with bounded domain) are well-known techniques; see, e.g., Chapter 12 of [72]. We shall comment on a simple instance of how fast transforms can be applied in combination with spectral methods in more detail in the introduction to Part II of this thesis. In recent years, going back to K. Hallatschek's work [44], fast transforms for sparse grids (i.e., hyperbolic reductions) and their application to solving PDEs has become a topic. As for trigonometric polynomials and periodic problems, see the works [34, 35, 52, 53, 54]. As for algebraic polynomials and bounded, non-periodic as well as unbounded problems, an application of the Fast Cosine Transform for sparse grid interpolation on $[-1,1]^d$ is presented in [56]; the work [77] studies hyperbolic cross approximations based on Jacobi polynomials and considers applications to high-dimensional elliptic equations; sparse grid Chebyshev–Galerkin methods for elliptic problems on bounded and unbounded domains (or Chebyshev–Legendre–Galerkin and Chebyshev–Hermite–Galerkin methods, respectively) are proposed in [78, 79]; finally, in [16], the authors consider orthogonal polynomial expansions on sparse grids and develop a rather involved procedure for the computation of the expansion coefficients.

So, how does this thesis fit into the broader picture and what do we intend to do? We develop a procedure for an efficient computation of matrix-vector products of the above form, $\mathbf{Hv}$, for any choice of dimension d that we shall henceforth call the *fast algorithm*. Our approach is supposed to meet the following benchmark criteria:

- Index set reduction proves to be enormously helpful for lowering the computational costs. However, not all kinds of reductions are always viable due to their specific approximation properties with respect to specific problems. The above references for fast transforms on reduced grids are all related to hyperbolic reductions. The fast algorithm, in contrast, works with (almost) any kind of subset of the full index cube and is therefore compatible with the idea of skipping basis functions and making the matrix smaller in a more general sense.

- It scales essentially linearly with the size of the underlying Galerkin basis. Therefore, it is necessarily matrix-free.

- The fast algorithm exploits the structure of the Galerkin representation matrix $\mathbf{H}$ in using the facts that the basis functions are mutually orthogonal and consist of tensor products of univariate functions that are each constructed recursively. Besides these orthogonality and recursion properties, the underlying Galerkin basis can be any set of function that is suitable for the very problem under consideration. In particular, it is applicable with sets of basis functions that do not easily allow for a fast transform, or that do not allow for such a transform at all.

This is the task we aim to perform in this thesis. In addition, as the fast algorithm is only an approximate means to compute the above matrix-vector product, we wish the corresponding error to decay spectrally and thus preserve the appproximation quality of the spectral ansatz itself.

In its very basic form, the fast algorithm has originally been devised for a Hamiltonian operator with a multiplicative, possibly time-dependent potential V, viz.,

$$H = -\frac{1}{2}\Delta + V, \qquad (V\psi)(x,t) = V(x,t)\psi(x,t).$$

For the sake of brevity, we drop the time-dependency in V for the moment. The Laplacian can be trivially dealt with using an appropriate choice of basis functions. In later chapters of this thesis, we transfer the approach from potentials to spatial derivatives that do no longer allow for as easy a treatment as the Laplacian, and that may even be accompanied by non-constant coefficient functions. Additionally, we consider problems on bounded domains and present an adaptation of the basic idea to boundary terms—without having to employ a Galerkin basis suitably taylored for the very problem under consideration. A further topic is nonlinearities. We shall comment on the details in due time. Anyway, the fast algorithm shall be seen rather as a general tool or methodology for problems that share some specific, yet fundamental properties with the above discretization of the linear Schrödinger equation. It is in no way restricted to the Schrödinger equation alone.

How have we solved our task and what are those fundamental properties that any problems needs to exhibit in order to allow for an application of the fast algorithm? Our method rests on two pillars: First, a suitable approximation of the potential by a multivariate polynomial; second, a direct operation procedure with representation matrices of the position operators with respect to every single coordinate.
As a first step, we approximate the potential by a multivariate polynomial. Any set of polynomial basis functions is viable, as long as the polynomial representation yields a sum of products of univariate polynomials. A suitable way to do this is Chebyshev interpolation over a given cube $[-S,S]^d$, $S \in \mathbb{R}$. Slightly deviating from the actual proceeding, we set

$$V(x) \approx \sum_{\mathbf{r}\in\mathcal{R}} \hat{v}_{\mathbf{r}} \prod_{\alpha=1}^{d} T_{r_\alpha}(x_\alpha/S), \qquad x \in [-S,S]^d,$$

where T_r is the univariate Chebyshev polynomial of order r, and $\mathcal{R} = \mathcal{R}(d,R) \subset \mathbb{N}^d$ is another set of multi-indices with threshold R. The fast algorithm itself allows for other than a Chebyshev basis, though.
Next, we define coordinate matrices, i.e., representations of position operators,

$$\mathbf{X}^{(\alpha)}_{\mathbf{jk}} = \int_{\mathbb{R}^d} \bar{\varphi}_{\mathbf{j}}(x) x_\alpha \varphi_{\mathbf{k}}(x)\,\mathrm{d}x, \qquad \alpha = 1,\ldots,d.$$

Using the orthogonality property and the recurrence relation for $\varphi_{\mathbf{k}}$, the product $\mathbf{X}^{(\alpha)}$ can be done directly in linear time only, without assembling $\mathbf{X}^{(\alpha)}$.
The key idea is then simply to insert $\mathbf{X}^{(\alpha)}$ in place of the coordinate x_α into the polynomial representation of V, which we represent by $V(\mathbf{X})$. Using the one-dimensional recurrence for the Chebyshev polynomial in combination with the direct operation approach for the coordinate matrices, this can be turned into a rigorous algorithm that yields an efficient way to compute the product $V(\mathbf{X})\mathbf{v}$. As we shall explain in much more detail, the overall time complexity is proportional to

$$|\mathcal{R}||\mathcal{K}|.$$

The requirements that the solution ψ have support within the cube $[-S, S]^d$ and that the potential be substantially smoother than the solution translate into the relations

$$K \gg R, \qquad |\mathcal{K}| \gg |\mathcal{R}|$$

between the two index sets involved. In this sense, since the contribution from $|\mathcal{R}|$ to the computational costs is negligible, the fast algorithm scales essentially linearly with $|\mathcal{K}|$.

Inserting the coordinate matrices into the polynomially approximated potential is a rather tentative idea. We therefore dedicate a significant proportion of this thesis to a thoroughgoing error analysis in order to account for how good an approximation $V(\mathbf{X})\mathbf{v}$ actually is when compared to a suitable entrywise quadrature approximation of the matrix representation of V. As it turns out, there is a close connection between the fast algorithm and Gaussian quadrature: For the example of the linear Schrödinger equation over $\mathbb{R}^d$, we choose tensor products of univariate Hermite functions as basis functions. Hermite functions are a natural and, thus, widely-used spectral basis for the Schrödinger equation on unbounded domains; see, e.g., [23, 61] for the linear and [30] for the nonlinear case. In particular, they are the eigenfunctions of the harmonic oscillator, which trivializes the derivative part. Now, if the integrals

$$\mathbf{V}_{\mathbf{jk}} = \int_{\mathbb{R}^d} \bar{\varphi}_{\mathbf{j}}(x) V(x) \varphi_{\mathbf{k}}(x) \,\mathrm{d}x$$

are approximated by a full-product Gauß–Hermite quadrature with exactly $K+1$ nodes in each direction, the approximate representation matrix $\mathbf{V}^{\mathrm{quad}}$ turns out to be equivalent to the matrix $V(\mathbf{X})$—at least if the index set $\mathcal{K}$ is the full index cube. This error due to quadrature can be estimated using an appropriate projection technique that is related to the exactness properties of Gaussian quadrature. However, if the index set $\mathcal{K}$ is reduced, invoking the exactness properties of Gaussian quadrature becomes considerably harder, and many arguments that are valid for the full index cube cease to do the trick. In this case, there is an additional error contribution from index set reduction. In case of a reduced index set, both the error due to quadrature and due to index set reduction can be analyzed by transforming the three-term recurrence relation for the underlying Hermite–Galerkin basis into suitable binary trees and doing some extensive combinatorial reasonings on these binary trees. If the potential can be sufficiently well approximated by a multivariate polynomial, i.e., if $K \gg R$, also the errors are well-behaved. E.g., for a vector $\mathbf{v} = (v_{\mathbf{k}})_{\mathbf{k}\in\mathcal{K}}$ that decays according to

$$v_{\mathbf{k}} \leq C \prod_{\alpha=1}^{d} \max\{1, k_\alpha\}^{-s}$$

with some $s \in \mathbb{N}$, we find the local error to decay according to

$$\big(V(\mathbf{X}) - \mathbf{V}^{\mathrm{quad}}\big)\mathbf{v} \leq C(d, \mathcal{R}, V^{\mathrm{pol}}, v, S) K^{-s}$$

where the constant depends on the dimension d, the size of the index set $\mathcal{R}$, the regularity of V^{pol} and $v = \sum_{\mathbf{k}\in\mathcal{K}} v_{\mathbf{k}} \varphi_{\mathbf{k}}$, and the size of the cube S, but is independent of K. The fast algorithm thus also decays spectrally—as does the overall Galerkin ansatz in the first place. As is commonly done, the global error can be decomposed into a standard L^2-projection

error and a defect contribution, which in turn employs the above local error result. Analogous results for the aforementioned generalizations to problems other than the Schrödinger equation are presented in Part II of this thesis.

The ideas that this thesis is based on as well as some of the techniques employed are due to the following sources. In [34, 35], V. Gradinaru studies a spectral approach with a hyperbolically reduced tensor product Fourier basis and collocation on a sparse grid for the linear Schrödinger equation with periodic boundary conditions. As is pointed out in by Ch. Lubich in [61], Chapter III.1.4, unlike on a full grid, the resulting coefficient ODE does not exhibit a Hermitian matrix, which possibly gives rise to numerical troubles and limits the range of applicable time-stepping methods. As a remedy, a Fourier Galerkin method in combination with an approximation of the potential by a trigonometric polynomial,

$$V(x) \approx \sum_{\mathbf{r}\in\mathcal{R}} \exp(i\mathbf{r}\cdot x),$$

is proposed. This is essentially the same idea as in the present thesis, but in the much simpler setting of a periodic problem. The setting is indeed much simpler since the triple-products

$$\int_{[-\pi,\pi]^d} \exp(-i\mathbf{j}\cdot x)\exp(i\mathbf{r}\cdot x)\exp(i\mathbf{k}\cdot x)\,\mathrm{d}x$$

do not vanish only if $j_\alpha = r_\alpha + k_\alpha$, and the corresponding matrix-vector product for the potential can trivially be evaluated exactly. We adopt this basic idea for the situation of an unbounded problem with a Hermite instead of a Fourier basis and, say, a tensor product $T_\mathbf{r}$ of Chebyshev polynomials in place of $\exp(i\mathbf{r}\cdot x)$. There does not seem to be an easily exploitable product identity for such integrals, and classical results such as [24] for the product of two Hermite polynomials do not seem to open a computationally viable loophole. In [23], in collaboration with E. Faou, the same authors have already transferred the very idea from trigonometric to algebraic polynomials, in the context of a splitting procedure for the linear Schrödinger equation in the semi-classical regime. They introduce the above coordinate matrices and propose a formal insertion into the polynomial. They present, however, no rigorous algorithmic description, and neither do they relate the idea to a suitably chosen Gaussian quadrature nor do they give an error analysis. This, together with a broader perspective on where this general idea can be employed and all subsequent generalizations and applications, is done in the present thesis.

As briefly explained above, the insight that $V(\mathbf{X})$ is equivalent to a full product of $(K{+}1)$-nodes Gaussian quadrature of the same type as the basis functions is the crucial step towards an error estimate. In the context of discrete variable representations (DVR), arguments of the kind as they lead to this very equivalence are common; see [59].

Not only does this make an error analysis feasible, such DVR-style reasonings also supply the ideas for the treatment of non-constant coefficients as presented in Part II of this thesis. Doing derivatives in linear time, in contrast, is a well-known topic; see, e.g., the textbook [15]. The ideas that our approach for boundary terms is based on shall be traced back separately in the extensive introduction to Part II.

Let us review some prominent features of our approach. First, it is appealing insofar as it combines the advantages from different strategies to coping with the curse of dimensionality:

It is matrix-free, allows for index set reductions of almost any kind, and it can be seen as a substitute for a missing fast transform.

Second, in all these respects, it retains a great flexibility with respect to specific choices of index sets, basis functions, and time integration schemes involving Galerkin matrix-vector products. In particular, the methodology is not limited to Hermite functions, but is applicable with all kinds of classical orthogonal polynomials since their construction is generic. In Part II of this thesis, we present the choice of a Legendre basis. All of the above general references for spectral methods contain extensive material on classical orthogonal polynomials, on which there is also an abundant literature. The rich variety of these polynomials has tempted some authors to speak about them in colorful wordings borrowed from animality. In the appendix to the textbook [47], e.g., the reader is guided through "a zoo of polynomials", while [10] presents a whole "bestiary" of basis functions. For further, comprehensive overviews over possible choices of basis functions and their main properties, see the more prosaic handbook [1] and its successor [67], both available online. The general theory of orthogonal polynomials can be found, e.g., in [31, 82].

Next, the fast algorithm brings together techniques from different fields, both for the algorithmic development and for the error analysis. Quadrature is never actually employed, and we consider the implementation to be relatively easy. The methodology is generic and thus serviceable also for spectral Galerkin approximations to other (linear and nonlinear) problems using a suitable Galerkin basis. Most prominent, though, is the fact that it scales essentially linearly—regardless of the dimension and regardless of the way the basis is reduced. Combining all these features in a single algorithm is clearly a novelty.

On the downside, the applicability of the fast algorithm is restricted to the case of potentials (or non-constant coefficients in derivative operators) that are considerably smoother than the solution we aim to approximate. In contrast to the aforementioned matrix-free approach from the chemical literature, this limits its range of application and to some extent corrupts its standing as a general tool for spectrally discretized PDEs. Finally, we still have a quadratic cost for transformations between physical and coefficient space. We need such transforms, e.g., to prepare the initial data and to visualize the solution. These operations are, however, rare compared to the evaluation of the differential operator, which needs to be done in every time step.

We conclude this introduction with a concise outline of the thesis. In Part I (Chapters 1–6), we present the fast algorithm in its basic form as devised for a multiplicative potential in the context of a linear Schrödinger equation. Whereas some immediate generalizations (nonlinearities, in particular) are discussed at the end of the first part, a generalization of the methodology to bounded domains is the topic of Part II (Chapters 7–11).

In Chapter 1, we present a setting where the fast algorithm in its basic form can ideally be employed (and that it has originally been devised for): Starting from a Galerkin ansatz for the discretization of a linear Schrödinger equation in space (Section 1.1), we motivate the choice of a tensor product basis of Hermite functions and discuss their main properties as they shall be put into use for the derivation of the fast algorithm (Section 1.2). These basis functions are taken from a multidimensional index set that is a subset of the aforementioned full index cube. The two kinds of index reductions considered in this thesis, viz., hyperbolic and additive reductions, are presented in Section 1.3. An important issue for the applicability of the fast algorithm is smoothness assumptions both about the wave function and about the potential. We comment on this issue together with how the poten-

tial can be approximated by a polynomial in Section 1.4. Finally, in Section 1.5, we give a brief sketch of Magnus integrators as our choice of time-stepping method together with a Lanczos-based approximation to the matrix exponential, and point out when exactly the matrix-vector products we aim to compute come into play.

In Chapter 2, we present the fast algorithm itself. After a concise record of all prerequisites for a succesful application (Section 2.1), we present the linearly scaling direct operation approach with the coordinate matrices $\mathbf{X}^{(\alpha)}$ in Section 2.2 as the core of the algorithm. A full algorithmic description, i.e., the interplay of inserting these coordinate matrices into the polynomially approximated potential and the application of the 1D Chebyshev recurrence for the polynomial representation basis, is given in Section 2.3, both in a recursive and in a non-recursive form. We shall discuss time and space complexity of both versions of the fast algorithm in Section 2.4, and comment on the implementation in Section 2.5. To conclude the presentation, we derive the exact relation between inserting $\mathbf{X}^{(\alpha)}$ into the polynomial and Gauß–Hermite quadrature (Section 2.6).

Chapter 3 is dedicated to an extensive time comparison between the fast algorithm on the one hand, and both an explicit assembly of the basis representation of the potential matrix (Section 3.1) and a matrix-free approach from the chemical literature (Section 3.2) on the other hand. The aim is to account for how much an economization in computational costs the fast algorithm actually is. Besides a theoretical complexity analysis of all three approaches to the matrix-vector product, some performance tests are shown in Section 3.3.

Chapter 4 contains the error analysis for the approximation of the matrix-vector product by the fast algorithm. Starting from an outline that includes the main error results (Section 4.1), we briefly derive the error due to polynomial interpolation of the potential (Section 4.2), before the full spatial discretization error in case of the full index cube is analyzed in Section 4.3. As mentioned earlier, this error decomposes into a contribution from Galerkin projection and another one from quadrature. Both error contributions are readily analyzed. Using a reduced Galerkin basis, in contrast, is considerably more difficult and shall thus be treated separately. We give a suitable error decomposition for the case of a reduced index set in Section 4.4. Besides an error due to quadrature, we additionally have to deal with an error contribution due to index set reduction. Bounding these errors necessitates an appropriate decay assumption on the vector as we discuss it in Secion 4.5. Both errors can then be analyzed using the idea of converting the underlying three-term recurrence for the univariate Hermite basis functions into suitably constructed binary trees (Sections 4.6 and 4.7). To conclude the error analysis, we give some remarks on the actual error behavior as we expect them in practical test (Section 4.8).

In Chapter 5, we show extensive numerical tests that corroborate the theoretical findings. These tests cover three different kinds of errors: First, local errors due to quadrature and index set reduction when applying the fast algorithm instead of explicitly assembling the basis representation of the potential and multiplying it with a vector (Section 5.1), both for hyperbolic and additive reductions. Next, in Section 5.2. we discuss how using the fast algorithm in each step of the Hermitian Lanczos process yields a local perturbation to the approximation of the matrix exponential. Finally, concluding the way from local to global errors, we present a time propagation study for an instance of the original Schrödinger equation (Section 5.3).

The last chapter of Part I, is dedicated to three immediate further applications of the basic fast algorithm that are all still related to the Hermite–Galerkin framework for a PDE on an unbounded domain. Having carved out retrospectively the essentials and non-

essentials from the above derivation of the fast algorithm (Section 6.1), we first comment on derivatives other than the Laplacian that do no longer allow to invoke the Hermite eigenfunction relation, such that the derivative part no longer becomes trivial (Section 6.2). This way, we meet the objection the fast algorithm might not be applicable unless derivatives can be trivialized. Second, to show that restricting the wave function to a fixed cube known in advance is dispensable, we briefly discuss the case of a moving wavepacket basis instead of basis functions localized around zero (Section 6.3). Finally, in Section 6.4, we show how the methodology carries over to the nonlinear Schrödinger equation.

Chapter 7 is the introduction to Part II of this thesis, where we present a generalization of the fast algorithm to bounded domains. In contrast to the immediate generalizations from the preceeding chapter, we leave the Hermite–Galerkin framework behind. Due to the presence of boundary conditions and derivatives with non-constant coefficients, the above techniques need to be developed further. We shall discuss the specific aims and challenges together with references to the existing literature in this second introduction.

In Chapter 8, as a well-suited, educational example of an initial-boundary value problem for an application of the generalized fast algorithm, we present a Galerkin approximation to the wave equation on the unit hypercube with non-constant coefficients together with Engquist–Majda boundary conditions (Sections 8.1 and 8.2). In particular, we comment on how to impose the boundary conditions weakly. The chosen Galerkin basis is a set of tensor products of Legendre polynomials indexed over the full index cube, introduced in Section 8.3.

This Galerkin ansatz plus weak imposition of boundary conditions necessitates matrix-vector products with three kinds of matrices in each time step: representations of non-constant coefficients, derivatives, and boundary term matrices. Chapter 9 contains the efficient procedures for the corresponding matrix-vector products. Again, there is a specific relation of the generalized fast algorithm to Gaussian quadrature. Starting from a discussion of Gauß–Legendre quadrature (Section 9.1), we present the effective procedures corresponding to the three kinds of matrices in Sections 9.2–9.4. The chapter concludes with a brief note on the overall complexity of the method (Section 9.5).

In Chapter 10, we present an error analysis for the overall approximation to the solution of our model problem. As we shall again make use of polynomial interpolation, after a concise outline given in Section 10.1 where we also briefly summarize the main results, we give a separate estimate for the polynomial interpolation error when approximating the coefficient functions as occurring in the wave equation (Section 10.2). Our way of weakly imposing the boundary condition allows for the derivation of a stability estimate of the spatially discrete approximation that closely parallels the continuous case, as will be shown in Section 10.3. Finally, a bound for the overall error due to spatial discretization is derived in Section 10.4.

In the last chapter, we present some numerical experiments when propagating in time the acoustic wave equation using the above discretization in space together with the generalized fast algorithm.

Contributions and sources

We give a concise overview over the references the material presented in this thesis is drawn from.

As explained in the above introduction and again in Section 2.3, the basic idea for the fast algorithm is due to E. Faou, V. Gradinaru, and Ch. Lubich; see [23]. Their idea is equivalent to the first version of the fast algorithm as given in this thesis, but neither do they specify the chosen polynomial basis for the approximation to the potential nor do they provide a rigorous algorithmic description. The relation to Gaussian quadrature, the error analysis, and all generalizations are due to the present thesis alone.

The sources we have drawn our material from are the following: The main source, which Part I is based upon, is a published article by the same author; see [12]. This covers the first chapter; Sections 2.1–2.3, 2.6; Sections 4.1, 4.4–4.8; Sections 5.1 and 5.2. The way we propose to assemble the matrix given in Section 3.1 has improved, though; the assembly can now be done faster as in [12], but is still decisively outperformed. The second version of the fast algorithm as given in Section 2.3.4 is a modification of Algorithm 1 from [13]; it had not been developed yet when [12] was published. The numerical experiments shown in Section 5.3 are based on the corresponding experiments done in [12], but with a different initialization and partly different choices of parameters. Throughout Part I, all numerical experiments have been redone and extended.

The next source is an unpublished, yet submitted manuscript by the same author in collaboration with E. Kieri[2] from Uppsala; see [13]. Most of the research reported in this manuscript was carried out while E.K. was visiting Universität Tübingen. This covers Part II of this thesis, but for a slightly modified introduction and additional numerical experiments for the reduced index set case. Both authors have contributed equally to the manuscript. Contributions that are exclusively due to E.K. are the following: First, the weak imposition of boundary conditions by a penalty approach (see Chapter 7 and Section 8.2 and the references therein) together with the stability estimates facilitated by this ansatz (see the proofs of Lemmas 8 and 11). These are well-known techniques, which E.K. has acquainted me with, for the benefit of our common research. Second, using a projection matrix that is related to the exactness of Gaussian quadrature (see the proof of Theorem 8; reused in the proof of Lemma 4). Third, many technicalities given in the proofs of Lemma 10 and Theorem 8 are due to E.K.

Finally, the idea of sequential summations as presented in Section 3.2 is due to the following sources. Although matrix-free computations have been a topic in computational

[2] Division of Scientific Computing, Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden. emil.kieri@it.uu.se

chemistry for decades, it is T. Carrington and his collaborators who have most prominently advocated for the sequential summations approach as a natural way to do products in a matrix-free way, and, to the best of our knowledge, they have contributed the most to the development of this idea. The material given in Section 3.2 has been compiled from [11, 90], adapted for a mathematical audience, and boiled down from the chemical context to its pure algorithmic core. Additionally, personal communication with G. Ávila, who has advanced the technique further than we present it, during my visit at Queen's University has considerably helped giving the presentation its final shape. The idea has never been laid out in this form before.

In all chapters, if some material has been taken from the above sources, we have rather freely adapted it: The material has been rearranged, more details have been provided, and the notation has been standardised. Figures taken from one of the above sources are indicated as such.

To a considerable extent (ca. 30%), this thesis presents material that has never been published or submitted before. In more detail, the following sections or chapters are completely new: the comments on complexity and implementation of the fast algorithm given in Sections 2.4 and 2.5; the time comparison study given in Chapter 3; Sections 4.2 and 4.3 on the errors due to interpolation and quadrature (full index cube), respectively, and the discussion of the decay assumption given in Section 4.5; Section 5.3, and the experiments given in Sections 5.1 and 5.2 have been extended as compared to [12]; all generalizations presented in Chapter 6.

I.

Basic fast algorithm

1 Spectral approximation of the linear Schrödinger equation

In Part I of this thesis, we present the fast algorithm as applied to the linear time-dependent Schrödinger equation with a multiplicative potential. This is what the fast algorithm has originally been devised for; see [12, 23]. Although the underlying methodology is rather general and allows for a wider range of applications as well as modifications, we postpone further generalizations to Part II, focussing for now on the essential ingredients for a straightforward implementation of the methodology in a sufficiently simple setting. This first chapter is dedicated entirely to setting up a scenario where the fast algorithm can be put into use.

In Section 1.1, we introduce a Galerkin discretization of the Schrödinger equation without specifying the underlying Galerkin basis of $L^2(\mathbb{R}^d)$-orthonormal functions yet, and we rewrite the Galerkin condition as a system of ODEs for the coefficients of the unknown approximation. In one spatial dimension, we employ Hermite functions as basis functions, to be introduced in Section 1.2. In higher dimensions, an appropriate choice is a basis of tensor products of univariate Hermite functions. Some choices of sets of multi-indices underlying the tensor product basis are presented in Section 1.3. In Section 1.4, we discuss the approximation of the potential by a multivariate polynomial, which already constitutes the first step in developing the fast algorithm. Additionally, we bring up the issues of smoothness assumptions both for the exact solution and for the potential as they are required for the fast algorithm to be applicable. The fast algorithm is an efficient means to do matrix-vector products as they typically arise when propagating in time the ODE resulting from spatial discretization. In Section 1.5, we present a Magnus integrator in combination with a Lanczos-based approximation to the matrix exponential as a discretization in time, and we show where exactly the fast algorithm comes into play.

1.1 Galerkin approach

We consider the dimensionless linear Schrödinger equation over the whole $\mathbb{R}^d$,

$$\begin{aligned} i\psi_t(x,t) &= H(x,t)\psi(x,t) \\ &= -\frac{1}{2}\Delta\psi(x,t) + V(x,t)\psi(x,t), \end{aligned} \qquad x = (x_1,\ldots,x_d) \in \mathbb{R}^d, \tag{1.1}$$

for $t \geq 0$, with a Hamiltonian operator H that consists of a Laplacian and a real-valued multiplicative, possibly time-dependent potential.

Let us briefly comment on the existence and uniqueness of a solution to (1.1). Throughout this thesis, we assume that $V(\cdot,t)$ is such that $H(\cdot,t)$ yields a well-defined self-adjoint operator on $D(H(\cdot,t)) = D(\Delta) = H^2(\mathbb{R}^d)$ for all $t \geq 0$. This hold, e.g., if V is bounded, which is a special case of the Kato–Rellich theorem; see [55], Section V.4.1. Then, for an initial value $\psi_0 = \psi(x,0) \in H^2(\mathbb{R}^d)$, the equation (1.1) has a unique solution $\psi(x,t) = \Phi(t)\psi_0(x)$ with a unitary propagator $\Phi(t)$. See, e.g., the textbook [39], Chapter 2, based on [73], for a proof in case V is time-independent. In case of a time-dependent Hamiltonian, the proof can be found in [74], Section X.12, for $d = 3$ and $V(\cdot,t) = V_\infty(\cdot,t) + V_2(\cdot,t)$, with $V_\infty(\cdot,t) \in L^\infty(\mathbb{R}^3)$, $V_2(\cdot,t) \in L^2(\mathbb{R}^3)$, and V being continuously differentiable in t, making use of another criterion for self-adjointness of H that follows from the Kato-Rellich theorem together with the Sobolev inequality on $\mathbb{R}^3$ (see [55], Section V.5.3). In Section 1.4, besides questions of existence and uniqueness of the solution, we shall comment on further assumptions on $V(\cdot,t)$ as they are related to the fast algorithm itself.

Using a Galerkin ansatz, we search for an approximation $\psi_\mathcal{K}$ to ψ,

$$\psi(x,t) \approx \psi_\mathcal{K}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}(d,K)} c_\mathbf{k}(t)\varphi_\mathbf{k}(x), \tag{1.2}$$

from the linear, finite-dimensional approximation space

$$\operatorname{span}\{\varphi_\mathbf{k}\,;\,\mathbf{k}\in\mathcal{K}\} \subseteq L^2(\mathbb{R}^d),$$

where the functions $\{\varphi_\mathbf{k}\}_{\mathbf{k}\in\mathbb{N}^d}$ form an $L^2(\mathbb{R}^d)$-orthonormal basis. We comment on the choice of basis functions in Section 1.2. The set $\mathcal{K}(d,K)$ consists of multi-indices $\mathbf{k}$,

$$\mathcal{K}(d,K) \subseteq \mathcal{K}_{\text{full}}(d,K) = \left\{\mathbf{k} = (k_1,\ldots,k_d) \in \mathbb{N}^d\,;\, 0 \leq k_\alpha \leq K\right\},$$

which attain at most $K+1$ different values in each component k_α for some given threshold K. Throughout this work, multi-indices (i.e., d-tuples of natural numbers) are typed in bold, lower-case letters. For the sake of brevity, we shall often omit the arguments d and K in $\mathcal{K}$. We comment on the choice of $\mathcal{K}$ in Section 1.3.

The unknown coefficients $c_\mathbf{k}$ are determined such that, when inserting $\psi_\mathcal{K}$ into the equation (1.1), the residual is orthogonal to the approximation space, i.e.,

$$\big(\varphi_\mathbf{j}, i(\psi_\mathcal{K}(\cdot,t))_t - (H\psi_\mathcal{K})(\cdot,t)\big) = 0 \qquad \forall\,\mathbf{j}\in\mathcal{K}. \tag{1.3}$$

The curved brackets denote the standard $L^2(\mathbb{R}^d)$-inner product with complex conjugation in its first argument,

$$(\chi,\eta) = \int_{\mathbb{R}^d} \overline{\chi(x)}\eta(x)\,\mathrm{d}x, \qquad \chi,\eta\in L^2(\mathbb{R}^d). \tag{1.4}$$

The $L^2(\mathbb{R}^d)$-norm is then denoted by $\|\cdot\|$. We also introduce the $L^2(\mathbb{R}^d)$-orthogonal projection $\mathcal{P}_\mathcal{K} : L^2(\mathbb{R}^d) \to \operatorname{span}\{\varphi_\mathbf{k}\,;\,\mathbf{k}\in\mathcal{K}\}$,

$$(\varphi, \chi - \mathcal{P}_\mathcal{K}\chi) = 0 \qquad \forall\,\varphi\in\operatorname{span}\{\varphi_\mathbf{k}\,;\,\mathbf{k}\in\mathcal{K}\},\ \chi\in L^2(\mathbb{R}^d). \tag{1.5}$$

Assembling the coefficients as a vector

$$\mathbf{c}(t) = (c_\mathbf{k}(t))_{\mathbf{k}\in\mathcal{K}}$$

and inserting the ansatz (1.2) into (1.3) yields a linear system of $|\mathcal{K}|$ ODEs

$$i\dot{\mathbf{c}}(t) = \mathbf{H}(t)\mathbf{c}(t), \qquad t \geq 0,$$

with a matrix representation

$$\mathbf{H}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, H(\cdot,t)\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K}, \tag{1.6}$$

of the Hamiltonian operator H in the Galerkin basis $\{\varphi_{\mathbf{k}}\}_{\mathbf{k}\in\mathcal{K}}$. The number of equations is $|\mathcal{K}| \leq (K+1)^d$.

A first reasonable move is to get rid of all spatial derivatives hiding in $\mathbf{H}(t)$. This can be achieved by choosing an appropriate Galerkin basis and splitting the Hamiltonian operator into a corresponding eigenoperator D and a multiplicative potential $W(\cdot,t)$, viz.,

$$H(\cdot,t) = D + W(\cdot,t), \qquad D\varphi_{\mathbf{k}} = \lambda_{\mathbf{k}}\varphi_{\mathbf{k}}. \tag{1.7}$$

We present such a choice of $\{\varphi\}_{\mathbf{k}\in\mathbb{N}^d}$ in the next section. The gain is that the action of a representation of D on a vector reduces to a simple multiplication with a diagonal eigenvalue matrix, which scales only linearly with the size of the Galerkin basis. The system then reads

$$i\dot{\mathbf{c}}(t) = \mathbf{D}\mathbf{c}(t) + \mathbf{W}(t)\mathbf{c}(t), \qquad t \geq 0, \tag{1.8}$$

with matrices

$$\mathbf{D} = \mathrm{diag}_{\mathbf{k}\in\mathcal{K}}(\lambda_{\mathbf{k}}), \qquad \mathbf{W}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W(\cdot,t)\varphi_{\mathbf{k}}), \quad \mathbf{j},\mathbf{k} \in \mathcal{K}. \tag{1.9}$$

Throughout this work, bold, upper-case letters denote matrices (most often of size $|\mathcal{K}|\times|\mathcal{K}|$), and vectors (typically of size $|\mathcal{K}|$) are typed in bold, lower-case letters. The letter $\mathbf{c}$ is reserved for a coefficient vector of a spatially discrete equation, while $\mathbf{v}$ denotes a generic vector.

1.2 Hermite basis

As a Galerkin basis, we choose tensor products of univariate Hermite functions. In this section, we briefly review their construction alongside with basic properties as they are needed in later chapters. See, e.g., the standard reference [1], Section 22, or the quantum dynamics-related books [61], Chapter III.1.1, and [85], Section 7.7, for more details.

1.2.1 Hermite functions in 1D

In 1D, the Hermite functions can be constructed using the *three-term recurrence relation*

$$\begin{aligned} \varphi_{-1} \equiv 0, \quad \varphi_0(x) &= \pi^{-1/4}e^{-x^2/2}, \\ x\varphi_k(x) &= \sqrt{\frac{k+1}{2}}\varphi_{k+1}(x) + \sqrt{\frac{k}{2}}\varphi_{k-1}(x), \qquad k \geq 0, \end{aligned} \qquad x \in \mathbb{R}, \tag{1.10}$$

which yields a complete $L^2(\mathbb{R})$-orthonormal set $\{\varphi_k\}_{k\in\mathbb{N}} \subset \mathcal{S}(\mathbb{R})$ of Schwartz functions, in particular, $(\varphi_j, \varphi_k) = \delta_{jk}$. The recurrence relation (1.10) plays a crucial role both for

the algorithm development and for the error analysis given in later chapters. An explicit expression for the k-th Hermite function is

$$\varphi_k(x) = \pi^{-1/4} \left(2^k k!\right)^{-1/2} H_k(x) e^{-x^2/2},$$

where H_k denotes the classical Hermite polynomial of degree k. Thus, φ_k is the product of a normalization factor times a polynomial times a Gaussian. The Hermite functions φ_k differ from their polynomial counterparts H_k by the facts that they are orthogonal with respect to an unweighted scalar product, that they are bounded functions, viz., $\|\varphi_\mathbf{j}\| = 1$, and that they decay faster than any inverse power of x as $x \to \pm\infty$. Figure 1.1 shows some plots for different choices of k, where it becomes visible that φ_k is even if k is even, and otherwise odd. The larger the index k, the more oscillatory the corresponding function becomes and the wider it spreads. The largest extremal is bounded by $\sqrt{2(k+1)}$. The Hermite functions are readily seen to be the eigenfunctions of the harmonic oscillator, i.e.,

$$-\frac{1}{2}\left(\frac{\partial^2}{\partial x^2} - x^2\right)\varphi_k = \left(k + \frac{1}{2}\right)\varphi_k. \tag{1.11}$$

Introducing the operator

$$(A\chi)(x) = \frac{1}{\sqrt{2}}\left(x\chi(x) + \frac{\mathrm{d}}{\mathrm{d}x}\chi(x)\right), \tag{1.12}$$

we can state the following approximation result for a truncated Hermite expansion; see [76], Section 7.3: For all $0 \le s \le K+1$,

$$\|\chi - \mathcal{P}_K\chi\| \le C(s)K^{-s/2}\|A^s\chi\|$$

for all $\chi \in L^2(\mathbb{R})$ with $A^s\chi \in L^2(\mathbb{R})$, where $C(s)$ is a positive constant that depends on s only, and $\mathcal{P}_K$ is the $L^2(\mathbb{R})$-orthogonal projection onto the linear space spanned by the first $(K+1)$ Hermite functions. See also [61], Theorem III.1.2 for an analogous result for the case of χ being a Schwartz function.

1.2.2 Tensor product basis

In higher dimensions, we consider tensor products of Hermite functions, i.e.,

$$\varphi_\mathbf{k}(x) = \varphi_{k_1}(x_1) \cdot \ldots \cdot \varphi_{k_d}(x_d), \qquad x = (x_1, \ldots, x_d) \in \mathbb{R}^d,$$

where $\mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d$ is a multi-index and φ_{k_α} are univariate Hermite functions as above, $1 \le \alpha \le d$. Again, $\{\varphi_\mathbf{k}\}_{\mathbf{k}\in\mathbb{N}^d} \subset \mathcal{S}(\mathbb{R}^d)$ is a complete $L^2(\mathbb{R}^d)$-orthonormal set of Schwartz functions. Due to the eigenfunction property (1.11), we find

$$-\frac{1}{2}\left(\Delta - \sum_{\alpha=1}^d x_\alpha^2\right)\varphi_\mathbf{k} = \sum_{\alpha=1}^d \left(k_\alpha + \frac{1}{2}\right)\varphi_\mathbf{k}. \tag{1.13}$$

We comment on the approximation properties of a truncated tensor product basis with indices taken from a multidimensional index set $\mathcal{K} \subset \mathbb{N}^d$ in the next section. Turning back to the decomposition (1.7), we can thus define

$$D = -\frac{1}{2}\left(\Delta - \sum_{\alpha=1}^d x_\alpha^2\right), \qquad W(x,t) = V(x,t) - \frac{1}{2}\sum_{\alpha=1}^d x_\alpha^2 \tag{1.14}$$

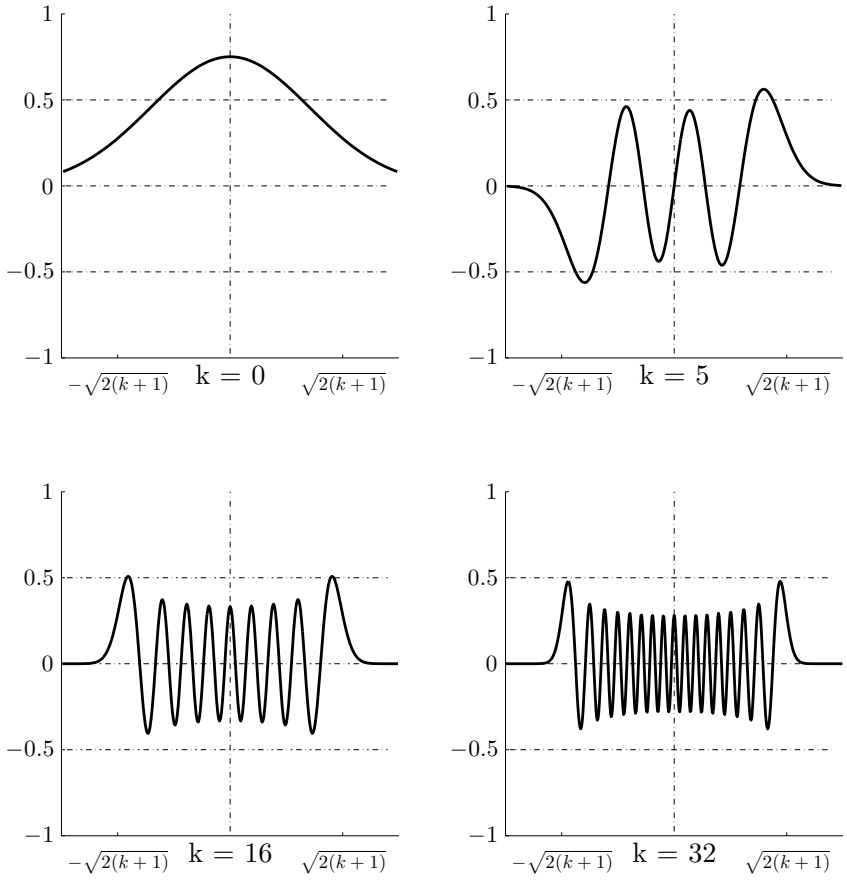


Figure 1.1: [12] Univariate Hermite functions for some choices of k.

with V given as in (1.1), where (slightly overloading the notation) x_α denotes both the position operator with respect to the αth coordinate and the spatial coordinate component itself. This yields a system of the form (1.8) with a diagonal eigenvalue matrix $\mathbf{D}$ with diagonal entries $\lambda_{\mathbf{k}} = \sum_{\alpha=1}^{d} \left(k_\alpha + \frac{1}{2}\right)$ and no spatial derivatives left.

1.3 Multidimensional index sets

We comment on possible choices of the multi-index set $\mathcal{K}(d, K) \subseteq \mathbb{N}^d$. Both from a conceptual point of view and with respect to an implementation of the subsequent method, the *full index cube*

$$\mathcal{K}_{\text{full}}(d, K) = \left\{ \mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d \,;\, 0 \leq k_\alpha \leq K \right\}$$

is clearly the simplest choice. However, the system (1.8) is then of size $|\mathcal{K}_{\text{full}}| = (K+1)^d$, thus, the number of unknowns grows exponentially with the dimension d. Disregarding for a moment the computational costs for an assembly of the right-hand side matrix $\mathbf{W}$ by an appropriate entrywise quadrature, doing the corresponding matrix-vector products explicitly scales quadratically with the size of the basis. In case of $d \geq 4$, this yields infeasible computational costs even for moderate choices of K, and interesting computational

problems become intractable due to the computational complexity of the full index cube ansatz. For this obstacle, the catch phrase *curse of dimensionality* has been coined; see [7]. The assembly of $\mathbf{W}$ proves to be even more harmful than the quadratically scaling matrix-vector products. In Section 3.1, we discuss how to assemble the matrix $\mathbf{W}$—which we strongly disencourage to do, anyway. In Section 3.3, we show some experimentally obtained computation times.

A manifest strategy to cope with the computational challenge of the basis growing too large too quickly is a *reduction* of the chosen index set such that $\mathcal{K} \subsetneq \mathcal{K}_{\text{full}}$—without changing the overall approximation properties of the expansion (1.2) too much for the worse, hopefully. The idea is commonly traced back to [81]. Bungartz and Griebel [14] provide a survey of the literature on sparse grids; see also the overview in the above introduction for further references.

The fast algorithm allows for any index set $\mathcal{K}$ such that,

$$\forall\, \alpha = 1, \ldots, d: \qquad \mathbf{k} \in \mathcal{K} \text{ with } k_\alpha > 0 \quad \Rightarrow \quad \mathbf{k} - \mathbf{e}_\alpha \in \mathcal{K}, \tag{1.15}$$

where $\mathbf{e}_\alpha$ is the αth unit vector, i.e., we require the index set to be closed under componentwise decrements. The restriction (1.15) eases the presentation of what we shall call a direct operation procedure; see Section 2.2. We comment on this issue in due time.

1.3.1 Hyperbolically reduced index sets

Starting from a full tensor product expansion, we delete basis functions from the set $\mathcal{K}_{\text{full}}$ in case they do not add to the quality of the approximation significantly. In the present work, we consider *hyberbolically reduced* index sets,

$$\mathcal{K}_{\text{hyp}} = \left\{ \mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d \,;\, 0 \le k_\alpha, \prod_{\alpha=1}^{d} (1 + k_\alpha) \le K+1 \right\}. \tag{1.16}$$

The number of basis function then shrinks to

$$|\mathcal{K}_{\text{hyp}}(d, K)| \le (K+1)\log(K+1)^{d-1} = \mathcal{O}(K \log(K)^{d-1});$$

see, e.g., [61], Lemma III.1.4, for a proof. Thus, the number of unknowns depends only linearly on the threshold K, and the exponential dependency on d is mitigated by the presence of the logarithm. An illustration of hyperbolic reductions in 2D and in 3D is given in Figure 1.2.

As for the approximation properties, a hyberbolically reduced index set can still do us a decent service: We define

$$A^{\mathbf{s}} = A_1^{s_1} \cdot \ldots \cdot A_d^{s_d}, \qquad \mathbf{s} \in \mathbb{N}^d,$$

where $A_\alpha = \frac{1}{\sqrt{2}} \left(x_\alpha + \frac{\partial}{\partial x_\alpha} \right)$; cf. (1.12). Slightly, overloading the notation, we let x_α denote also the position operator with respect to the αth coordinate. Additionally, we introduce the notation

$$|\mathbf{s}|_\infty = \max_{1 \le \alpha \le d} s_\alpha, \qquad \mathbf{s} \in \mathbb{N}^d, \tag{1.17}$$

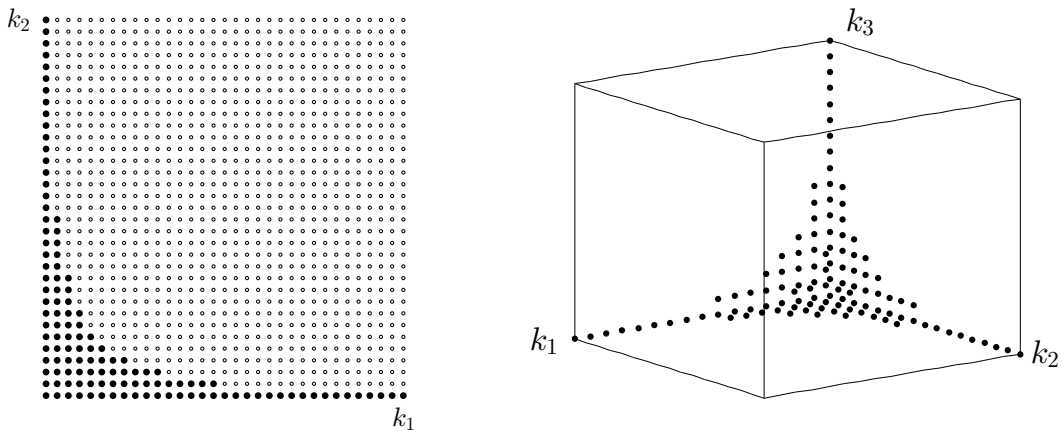


Figure 1.2: [12] Hyperbolically reduced index set. Left: $d=2$, $K=32$. Right: $d=3$, $K=16$. The number of retained indices is $\mathcal{O}(K\log(K)^{d-1})$. In case $d=2$, skipped indices are indicated by small empty circles.

for the maximum norm of an integer d-tuple. In [77], Theorem 3.1, the following estimate is shown with respect to a Korobov-type seminorm: For every fixed integer s and for all $\chi \in L^2(\mathbb{R}^d)$ such that $A^{\mathbf{s}}\chi \in L^2(\Omega)$ for all $\mathbf{s} \in \mathbb{N}^d$ with $0 \le |\mathbf{s}|_\infty \le s$,

$$\|\chi - \mathcal{P}_{\mathcal{K}}\chi\| \le C(d,s)K^{-s/2}\left(\sum_{|\mathbf{s}|_\infty=s}\|A^{\mathbf{s}}\chi\|\right)^{1/2}. \tag{1.18}$$

The constant $C(d,s)$ depends on d and s only. We introduce the notation

$$|\chi|_s = \left(\sum_{|\mathbf{s}|_\infty=s}\|A^{\mathbf{s}}\chi\|\right)^{1/2} \tag{1.19}$$

for the Korobov seminorm. An analogous result for the case of χ being a Schwartz function is shown in [61], Theorem III.1.5: For every fixed integer s and for all $\chi \in \mathcal{S}(\mathbb{R}^d)$,

$$\|\chi - \mathcal{P}_{\mathcal{K}}\chi\| \le C(d,s)K^{-s/2}\max_{0\le|\mathbf{s}|_\infty\le s}\|A^{\mathbf{s}}\chi\|,$$

where the maximum ranges over all $\mathbf{s}$ such that $0 \le |\mathbf{s}|_\infty \le s$. Again, the constant $C(d,s)$ depends on d and s only. For the right-hand side norm, we introduce the notation

$$|\chi|_{s;\infty} = \max_{0\le|\mathbf{s}|_\infty\le s}\|A^{\mathbf{s}}\chi\|. \tag{1.20}$$

Thus, even with a hyperbolically reduced index set, the approximation properties of a truncated Hermite basis are asymptotically as good as with a full index cube—given sufficient regularity of the approximated function.

1.3.2 Additive reduction

A less radical reduction is the *additive reduction*,

$$\mathcal{K}_{\text{add}} = \left\{\mathbf{k} = (k_1,\ldots,k_d) \in \mathbb{N}^d\,;\, |\mathbf{k}| \le K\right\}, \tag{1.21}$$

where

$$|\mathbf{k}| = \sum_{\alpha=1}^{d} k_\alpha, \qquad \mathbf{k} \in \mathbb{N}^d, \tag{1.22}$$

denotes the 1-norm of the integer d-tuple $\mathbf{k}$. The size of $\mathcal{K}_{\text{add}}$ equals the number of d-variate monomials of total degree $\leq K$, which is well-known to be

$$|\mathcal{K}_{\text{add}}(d, K)| = \binom{K+d}{d} \approx \frac{1}{d!} K^d.$$

Figure 1.3 provides an illustration.

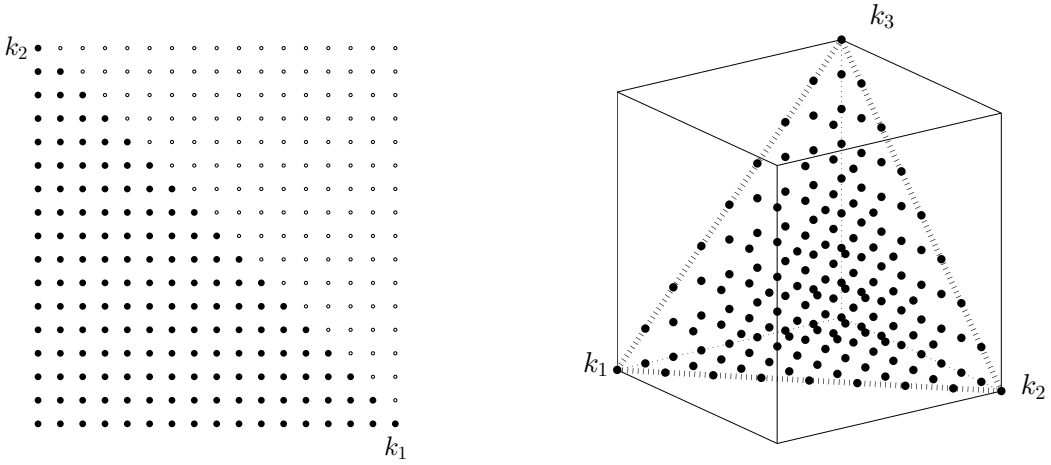


Figure 1.3: Additively reduced index set. Left: $d=2$, $K=16$. Right: $d=3$, $K=8$. In 2D, half of the indices is retained. In 3D, we retain only one sixth of them.

1.3.3 Linear order

The last paragraph in this section concerns the ordering of the occurring index sets. We deal with multidimensional objects consisting of data that is referenced by multi-indices. Throughout this thesis, we will always talk of "matrices" (of size $(|\mathcal{K}|\times|\mathcal{K}|)$ and "vectors" (of size $(|\mathcal{K}|)$, although these objects need not be seen as matricised or vectorised in principle. For this to make sense, the index set $\mathcal{K}$ (and all other occurring index sets, for that matter) requires a linear ordering. A convenient choice is the lexicographical order. Thus, $\mathbf{0} = (0,\ldots,0) \in \mathbb{N}^d$ is the first element, $\mathbf{e}_d = (0,\ldots,0,1) \in \mathbb{N}^d$ is the second element, and so forth up to the last element $(K,\ldots,K) \in \mathbb{N}^d$ of the full index cube. When talking occasionally about "next" or "previous" indices in a set, the underlying ordering is to be understood as the lexicographical order.

1.4 Smoothness assumptions and approximation of the potential

In this section, we discuss a basic assumption on the general problem setting (1.1) for the fast algorithm to be applicable, namely, an assumption on the smoothness of the exact solution ψ of the Schrödinger equation as related to the smoothness of the potential V.

1.4.1 Regularity of wave function and potential

We require the potential $W(\cdot,t)$ to be significantly smoother than the solution $\psi(\cdot,t)$ in a region $\Omega \subseteq \mathbb{R}^d$ where the solution does not essentially vanish for all times $t \geq 0$. We cast this as a relation of the Galerkin basis threshold K and of the degree R that is necessary for a decent polynomial approximation of the potential. For the sake of a simpler presentation, let us for now consider the case of $\psi(\cdot,t)$ being essentially supported within a cube

$$\Omega = [-S,S]^d \subseteq \mathbb{R}^d, \qquad S \text{ given},$$

with a maximal spatial extension of $\psi(\cdot,t)$ given in advance, which we describe by a support parameter $S \in \mathbb{R}$. Typically, for most times t, $\psi(\cdot,t)$ might only cover a much smaller subregion of Ω, as in the case of a moving wavepacket. Additionally, to vary the resolution of the Hermite approximation to ψ, we may define adequately compressed or stretched basis functions

$$\tilde{\varphi}_{\mathbf{k}}(x) = \prod_{\alpha=1}^{d} \varphi_{k_\alpha}(\tilde{S}x_\alpha), \qquad \mathbf{k} \in \mathcal{K},$$

for some positive stretching parameter $\tilde{S} \in \mathbb{R}$. As the univariate function φ_K is negligibly small outside the interval

$$[-\sqrt{2(K+1)}-1, \sqrt{2(K+1)}+1]$$

(see Figure 1.1), we require $\tilde{S}$ and K to be chosen such that

$$\tilde{S}S \geq \sqrt{2(K+1)}+1. \tag{1.23}$$

Increasing $\tilde{S}$ and K simultaneously yields a higher resolution within Ω. For ease of presentation, we restrict our attention to the case $\tilde{S}=1$.

1.4.2 Chebyshev interpolation

We approximate the potential $W(\cdot,t)$ as defined in (1.14) by a multivariate polynomial. We use, e.g., Chebyshev interpolation over the cube Ω to obtain an approximation

$$\begin{aligned} W(x,t) &\approx W^{\mathrm{pol}}(x,t) \\ &= \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) T_{\mathbf{r}}(x/S) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) \prod_{\alpha=1}^{d} T_{r_\alpha}(x_\alpha/S), \qquad x \in \Omega, \end{aligned} \tag{1.24}$$

with a multi-index set $\mathcal{R}(d,R) \subset \mathbb{N}^d$ that is defined by a maximal univariate polynomial degree parameter R. The approximation basis consists of tensor products of univariate Chebyshev polynomials, the latter being given by the recurrence relation

$$\begin{aligned} T_0(x) = 1, \quad T_1(x) = x, \\ T_{r+1}(x) = 2xT_r(x) - T_{r-1}(x), \quad r \geq 1, \end{aligned} \qquad x \in [-1,1]. \tag{1.25}$$

When $\mathcal{R} = \mathcal{R}_{\mathrm{full}}$ is the full index cube, the coefficients are given by

$$\hat{w}_{\mathbf{r}}(t) = \gamma_{r_1} \ldots \gamma_{r_d} \sum_{\mathbf{s}\in\mathcal{R}} W(Sz_{\mathbf{s}},t) T_{\mathbf{r}}(z_{\mathbf{s}}), \quad \gamma_{r_\alpha} = \begin{cases} 1/(R+1), & r_\alpha = 0, \\ 2/(R+1), & 1 \leq r_\alpha \leq R, \end{cases}$$

where $z_{\mathbf{s}} = (z_{s_1}, \ldots, z_{s_d})$ with z_{s_α} being the zeros of T_{R+1}. Within the cube Ω, this should give a reasonable approximation to the potential. Outside Ω, the Galerkin basis essentially vanishes anyway by assumption (1.23).

In case $W(\cdot, t) \in H^s_\omega(\Omega)$ with $s > d/2$, where $H^s_\omega(\Omega)$ is the weighted Sobolev space with the Chebyshev weight

$$\omega(x) = \prod_{\alpha=1}^{d} (1 - (x_\alpha/S)^2)^{-1/2}, \qquad x \in \Omega,$$

the coefficients $\hat{w}_{\mathbf{r}}$ decay exponentially, and the interpolation error in the weighted L^2-norm is bounded by

$$\left\| W(\cdot, t) - W^{\mathrm{pol}}(\cdot, t) \right\|_{L^2_\omega(\Omega)} \leq C R^{-s} \left| W(\cdot, t) \right|_{H^{s;R}_\omega(\Omega)},$$

the latter norm being a weighted Sobolev seminorm; see [15] for a detailed theory of approximation by orthogonal polynomials, in particular, Chapter 5.8, on Chebyshev approximation on product domains.

The index set $\mathcal{R}$ might also be reduced. In this case, there exist fast polynomial transforms on sparse grids to compute the interpolation coefficients efficiently; see, e.g., [16] for an $\mathcal{O}(R \ln(R)^{d+1})$ algorithm with still exponentially decaying coefficients $\hat{w}_{\mathbf{r}}$ (for W being sufficiently regular) with a hyperbolically reduced set $\mathcal{R}$.

1.4.3 Relation of index sets

Turning back to our basic smoothness assumption from the beginning of Section 1.4.1 and using the above terminology, we thus require that

$$|\mathcal{K}(d, K)| \gg |\mathcal{R}(d, R)|, \tag{1.26}$$

i.e., that the Chebyshev expansion W^{pol} consist of significantly fewer terms than the Hermite expansion $\psi_{\mathcal{K}}$, and

$$K \gg R. \tag{1.27}$$

As will become clear in the following chapters, we make these rather vaguely stated assumptions for two different reasons:

- First, to ease the computational burden. The fast algorithm is of complexity

  $$\mathcal{O}(|\mathcal{R}||\mathcal{K}|)$$

  which is essentially linear with the size of the basis only if (1.26) holds. If $|\mathcal{R}|$ comes close to $|\mathcal{K}|$, the computational costs deteriorate, and we end up with a possibly only slightly subquadratic procedure. In principle, though, the case $|\mathcal{R}| \geq |\mathcal{K}|$ is feasible for the fast algorithm, as long as $K \gg R$ still holds.

- Second, to guarantee convergence of the algorithm. For the analysis as given in Chapter 4, we require that powers of R be controlled by negative powers of K, which is guaranteed in case the relation (1.27) holds.

1.4.4 Moving wavepackets

We close the present section with a remark on the assumption that ψ is confined to a fixed cube during propagation in time. This assumption is restrictive, but in fact dispensable: If $\psi(\cdot, t)$ has support outside Ω, a reasonable Galerkin approximation might require a larger choice of K to reach out further in space, or the polynomial approximation of the potential W beyond the cube might become useless without increasing R. This is an issue of (1.2) with H replaced by $D + W^{\mathrm{pol}}$ yielding a good approximation to (1.1) in the first place. As long as the potential is sufficiently smooth in a region where the solution does not essentially vanish, the fast algorithm is applicable. This is true for the numerical examples considered in Chapter 5. If ψ resembles a moving wavepacket retaining a rather strong localization over time, a set of moving basis functions that adapt to this localization of ψ is preferable as a Galerkin basis. In fact, a modified version of the fast algorithm has been applied successfully with a moving wavepacket basis; see [23]. However, an adaptation to this setting both of the fast algorithm and notably of the following error analysis as given in Chapter 4 heavily complicates the presentation due to the presence of time-dependent evolution parameters in the basis. In Section 6.3, we shall briefly explain how the fast algorithm changes with a moving wavepacket basis. For the case of a full index cube, an adaptation of the analysis is sketched in Section 6.3.3. For a reduced, however, the analysis of the moving wavepacket case is still a partly unresolved issue, as we shall explain. In the following chapters, we restrict our attention to the case of confined ψ and basis functions localized around zero.

1.5 Discretization in time

In this section, we illustrate when to apply the fast algorithm during propagation in time after a Galerkin semidiscretization in space as given above.

1.5.1 Magnus integrators

The spatially discrete system under consideration reads

$$i\dot{\mathbf{c}}(t) = \mathbf{D}\mathbf{c}(t) + \mathbf{W}^{\mathrm{pol}}(t)\mathbf{c}(t), \qquad t \geq 0, \tag{1.28}$$

with $\mathbf{D}$ being the above diagonal eigenvalue matrix and

$$\mathbf{W}^{\mathrm{pol}}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}(\cdot, t)\varphi_{\mathbf{k}}), \quad \mathbf{j}, \mathbf{k} \in \mathcal{K}. \tag{1.29}$$

It is of the general form

$$i\dot{\mathbf{c}}(t) = \mathbf{A}(t)\mathbf{c}(t) \tag{1.30}$$

with a time-dependent Hermitian matrix $\mathbf{A}(t)$ and initial value $\mathbf{c}(0) = \mathbf{c}_0$. E.g., polynomial integrators of the form $\mathbf{c}^{(n+1)} = P(-ih\mathbf{A}(t))\mathbf{c}^{(n)}$ that comprise the class of explicit Runge–Kutta methods, as well as approximations to the matrix exponential by a Magnus integrator or by a splitting method each require multiplications of $\mathbf{A}$ with a vector in each time step. In the following, we consider instances of Magnus integrators; see the review [9], in particular, Sections 5 and 6, for numerical integration methods based on Magnus expansions. As for

splitting methods, see the review [64]. A concise review of numerical integrators for the time-dependent Schrödinger equation can be found in [60].

Magnus integrators are one-step methods consisting of an exponential stepping procedure of the form

$$\mathbf{c}^{(n+1)} = \exp(\mathbf{\Omega}^{(n)})\mathbf{c}^{(n)} \tag{1.31}$$

with a suitably chosen matrix $\mathbf{\Omega}^{(n)}$, where $\mathbf{c}^{(n)} \approx \mathbf{c}(t_n)$ is supposed to be an approximation at time $t_n = hn$ with time step size h. The exponential mid-point rule

$$\mathbf{\Omega}^{(n)} = -ih\mathbf{A}(t^n + h/2). \tag{1.32}$$

is a simple choice of $\mathbf{\Omega}^{(n)}$. A more complicated choice is the method

$$\mathbf{\Omega}^{(n)} = -\frac{ih}{2}(\mathbf{A}_1^{(n)} + \mathbf{A}_2^{(n)}) - \frac{i\sqrt{3}h^2}{12}[\mathbf{A}_2^{(n)}, \mathbf{A}_1^{(n)}] \tag{1.33}$$

with

$$\mathbf{A}_j^{(n)} = \mathbf{A}(t^n + c_j h),\ j = 1, 2,$$

based on two-stage Gauß–Legendre quadrature with nodes $c_{1,2} = \frac{1}{2} \mp \frac{\sqrt{3}}{6}$, where $[\cdot, \cdot]$ denotes the commutator of matrices. The schemes (1.32) and (1.33) both have a unitary propagator. For the time-dependent Schrödinger equation, M. Hochbruck and Ch. Lubich show that they are of optimal temporal orders 2 and 4, respectively, under the assumptions of a potential with bounded time derivatives of low order and certain commutator bounds; see [50].

1.5.2 Approximation of matrix exponential

The action $\exp(\mathbf{\Omega}^{(n)})\mathbf{c}^{(n)}$ of the matrix exponential of $\mathbf{\Omega}^{(n)}$ on a vector $\mathbf{c}^{(n)}$ as required in each time step (1.31) can be approximated using a Galerkin approach on a Krylov subspace where the Krylov basis is generated by the Hermitian Lanczos process. Using the Lanczos method to approximate matrix exponentials of the form $\exp(-ih\tilde{\mathbf{\Omega}}^{(n)})\mathbf{v}$ has first been proposed in [70], and Krylov subspace approximation to the matrix exponential operator has since been included into the famous review article [65]. An error analysis is due to [49], including further references to previous error studies. We briefly explain the essentials of the procedure. For more details, see [61], Chapter III.2.2.

We start writing

$$\mathbf{\Omega}^{(n)} = -ih\tilde{\mathbf{\Omega}}^{(n)}, \qquad \tilde{\mathbf{\Omega}}^{(n)} = \begin{cases} \mathbf{A}(t^n + h/2), & (1.32), \\ \frac{1}{2}(\mathbf{A}_1^{(n)} + \mathbf{A}_2^{(n)}) + \frac{\sqrt{3}h}{12}[\mathbf{A}_2^{(n)}, \mathbf{A}_1^{(n)}], & (1.33), \end{cases}$$

such that $\tilde{\mathbf{\Omega}}^{(n)}$ is a Hermitian matrix. The m-step Hermitian Lanczos process then generates recursively the Krylov basis

$$\mathbf{V}_m^{(n)} = (\mathbf{v}_1^{(n)} | \dots | \mathbf{v}_m^{(n)}) \in \mathbb{C}^{|\mathcal{K}| \times m}$$

of

$$\operatorname{span}\left(\mathbf{v}, \tilde{\mathbf{\Omega}}^{(n)}\mathbf{v}, \dots, \left(\tilde{\mathbf{\Omega}}^{(n)}\right)^{m-1}\mathbf{v}\right)$$

and a tridiagonal, real symmetric coefficient matrix

$$\mathbf{T}_m^{(n)} \in \mathbb{R}^{m\times m}$$

such that

$$\mathbf{T}_m^{(n)} = (\mathbf{V}_m^{(n)})^* \tilde{\mathbf{\Omega}}^{(n)} \mathbf{V}_m^{(n)},$$

starting from $\mathbf{v}_1^{(n)} = \mathbf{c}^{(n)}$. This requires m multiplications of $\tilde{\mathbf{\Omega}}^{(n)}$ with a Lanczos vector, where $m \ll |\mathcal{K}|$. A Galerkin approximation of the initial value problem (1.30) on this Krylov subspace then allows for

$$\exp(\mathbf{\Omega}^{(n)})\mathbf{c}^{(n)} \approx \mathbf{V}_m^{(n)} \exp(-ih\mathbf{T}_m^{(n)})e_1, \qquad e_1 = (1,0,\ldots,0)^T \in \mathbb{R}^m,$$

using the matrices $\mathbf{V}_m^{(n)}$ and $\mathbf{T}_m^{(n)}$. This reduces the problem to a diagonalization of the small matrix $\mathbf{T}_m^{(n)}$, and matrix-vector products with $\mathbf{V}_m^{(n)}$ scale linearly with $|\mathcal{K}|$ anyway.

Algorithm 1: Hermitian Lanczos process for the computation of $\mathbf{T}_m$ and $\mathbf{V}_m = (\mathbf{v}_1|\ldots|\mathbf{v}_m)$ such that

$$\mathbf{T}_m = \mathbf{V}_m^* \mathbf{A} \mathbf{V}_m$$

starting from a given unit vector $\mathbf{v} = \mathbf{v}_1$, where $\mathbf{A}$ is a Hermitian matrix and m is the number of Lanczos steps.

```
   function [T_m, V_m] = lanczos(A, v, m)
1  v_1 := v,        v_0 := 0,        β_1 := 0
2  for j to m do
3  |   u := A v_j − β_j v_{j−1},        α_j := v_j^* u
4  |   u := u − α_j v_j,        β_{j+1} = ‖u‖
5  |   v_{j+1} := u/β_{j+1}
6  |   T_jj := α_j
7  |   if j < m then
8  |   |   T_{j+1,j} := β_{j+1},        T_{j,j+1} := β_{j+1}
```

Algorithm 1 shows the Hermitian Lanczos process. Consider line **3**. In each step of the Lanczos process, the action of $\tilde{\mathbf{\Omega}}^{(n)}$ on a Lanczos basis vector $\mathbf{v}_k^{(n)}$, $1 \le k \le m$, consists of applications of the diagonal matrix $\mathbf{D}$ on vectors, which is trivially done exactly in $\mathcal{O}(|\mathcal{K}|)$, and of applications of the matrix $\mathbf{W}^{\mathrm{pol}}(t)$ on vectors, which can be approximated using the fast algorithm as given in Chapter 2 in $\mathcal{O}(|\mathcal{R}||\mathcal{K}|)$, i.e., in essentially linear time under the assumption (1.26) on the sizes of $\mathcal{R}$ and $\mathcal{K}$. The overall Lanczos process can thus be efficiently realized in $\mathcal{O}(m|\mathcal{R}||\mathcal{K}|)$ operations.

The time-dependent polynomial approximation coefficients $\hat{w}_{\mathbf{r}}(t)$ of $W^{\mathrm{pol}}(t)$ are evaluated at times t as prescribed by the chosen Magnus integrator. If the potential W does not depend on t, the coefficients $\hat{w}_{\mathbf{r}}$ can be computed in advance once and for all. If W is time-dependent, these coefficients have to be recomputed in each time step according to the choice of $\mathbf{\Omega}^{(n)}$. Algorithm 2 gives an algorithmic description of a numerical integration of (1.28) for the choice of the exponential mid-point rule (1.32).

Algorithm 2: Computation of an approximation

$$\mathbf{c}^{(N)} = (c_{\mathbf{k}}^{(N)})_{\mathbf{k}\in\mathcal{K}} \approx \mathbf{c}(t^N), \qquad t^N = Nh,$$

to the solution of (1.28) with initial value $\mathbf{c}^{(0)}$ using the exponential mid-point rule (1.32) with time step size h and m Lanczos steps in each time step. The underlying index set $\mathcal{K}(d,K)$ for the Galerkin basis is either the full cube or any appropriately reduced index set.

function $\mathbf{c}^{(N)} = \mathsf{time\text{-}stepping}\,(\mathbf{c}^{(0)}, h, m, N)$
for $n = 0$ **to** $N-1$ **do**

I. *Polynomial approximation of the potential*:
If W depends on time: compute coefficients of approximation

$$W(x, t^n + h/2) \approx W^{\mathrm{pol}}(x, t^n + h/2) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t^n + h/2) T_{\mathbf{r}}(x/S)$$

for some suitable index set $\mathcal{R}(d,R) \subset \mathbb{N}^d$ such that

$$|\mathcal{K}| \gg |\mathcal{R}|, \qquad K \gg R.$$

If W does not depend on time, compute $(\hat{w}_{\mathbf{r}})_{\mathbf{r}\in\mathcal{R}}$ in advance.

II. *Hermitian Lanczos process* (Algorithm 1):

$$[\mathbf{T}_m^{(n)}, \mathbf{V}_m^{(n)}] = \mathsf{lanczos}(\mathbf{D} + \mathbf{W}^{\mathrm{pol}}(t^n + h/2), \mathbf{c}^{(n)}, m)$$

In each Lanczos step:

- matrix-vector products with diagonal $\mathbf{D}$ are trivial
- use *fast algorithm* (see Chapter 2) for $\mathbf{W}^{\mathrm{pol}}(t^n + h/2)$

III. *Approximation of matrix exponential*:

$$\mathbf{c}^{(n+1)} = \mathbf{V}_m^{(n)} \exp(-ih\mathbf{T}_m^{(n)}) e_1$$

- use (small) diagonalization of $\mathbf{T}_m^{(n)}$
- matrix-vector product with $\mathbf{V}_m^{(n)}$ costs $\mathcal{O}(|\mathcal{K}|m^2)$

2 The fast algorithm

Having spent the first chapter of this thesis on setting up a detailed scenario for a possible application of the fast algorithm, we shall now finally turn to the algorithm itself.

The fast algorithm as we shall outline it in the present chapter allows for generalizations in quite a few respects, be it the choice of Galerkin basis functions, be it the way the polynomial approximation of the potential is done, or be it possible reductions of the multi-index sets both for the Galerkin approximation of the wave function and for the approximation of the potential. What needs to be given, in essence, is only an appropriate recurrence relation for the Galerkin basis, the fact that the approximate potential is a multivariate polynomial with significantly fewer terms than the approximate wave function expansion in the exact sense given in Section 1.4, and an index set for the basis that is closed under componentwise decrements. For ease of presentation, we shall stick for now with the chosen Hermite function approximation of the wave function and with Chebyshev interpolation for the potential. The index sets we consider in the present chapter, however, are full index cubes or arbitrary subsets, and need to obey the aforementioned restrictions only. Some immediate generalizations and further applications (nonlinearities, in particular) are postponed to Chapter 6. In Part II of this thesis, we present a more far-reaching adaptation of the fast algorithm to problems with position-dependent coefficient functions in the spatial derivative terms, with boundary values. The present chapter, though, gives the fast algorithm in its very basic form, i.e., for a multiplicative potential.

It is outlined as follows: In Section 2.1, we introduce the general setting for an application of the fast algorithm. Section 2.2 contains the construction of coordinate matrices, which do not need to be assembled, but allow for a *direct operation* approach. In Section 2.3, we use these matrices to set up the *matrix-free fast algorithm* to approximate the matrix-vector products under conisderation itself, and we provide an algorithmic description. Its complexity is analyzed in Section 2.4. In Section 2.5, we comment on the implementation of the fast algorithm. As it turns out, the approximation is closely linked to a suitably chosen quadrature rule, which is discussed in Section 2.6. This relation between the algorithm and a specific application of numerical quadrature will play a crucial role in the analysis of our method given in Chapter 4.

2.1 General setting

The fast algorithm is a means to compute efficiently the action of a $(|\mathcal{K}| \times |\mathcal{K}|)$-matrix $\mathbf{Q}$,

$$\mathbf{Q}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, q\varphi_{\mathbf{k}}) = \int_{\mathbb{R}^d} \varphi_{\mathbf{j}}(x) q(x) \varphi_{\mathbf{k}}(x) \, \mathrm{d}x, \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \tag{2.1}$$

on a vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$. The sets

$$\mathcal{K}(d,K) \subseteq \mathcal{K}_{\text{full}}(d,K) = \left\{\mathbf{k} = (k_1,\ldots,k_d) \in \mathbb{N}^d\,;\, 0 \le k_\alpha \le K\right\}$$

and $\mathcal{R}(d,R) \subseteq \mathcal{R}_{\text{full}}(d,R)$ are sets of multi-indices $\mathbf{k}$ and $\mathbf{r}$ with components k_α and r_α ranging from 0 to some given thresholds K and R, respectively. We often omit the arguments d, K, and R in $\mathcal{K}(d,K)$ and $\mathcal{R}(d,R)$. Both sets can be arbitrary subsets of the respective full index cubes, but we require that the set $\mathcal{K}$ be closed under componentwise decrements, see (1.15), and that the relations

$$|\mathcal{K}| \gg |\mathcal{R}|, \qquad K \gg R$$

hold. The functions $\varphi_{\mathbf{k}}$ are tensor products

$$\varphi_{\mathbf{k}}(x) = \varphi_{k_1}(x_1) \cdot \ldots \cdot \varphi_{k_d}(x_d), \qquad x = (x_1,\ldots,x_d) \in \mathbb{R}^d,$$

of univariate Hermite functions $\varphi_{k_\alpha}(x_\alpha)$ as introduced in Section 1.2. The univariate Hermite functions can be constructed by the three-term recurrence relation (1.10) and form an $L^2(\mathbb{R})$-orthonormal set. We introduce the domain

$$\Omega = [-S,S]^d \subseteq \mathbb{R}^d, \qquad S \text{ given in advance},$$

with a parameter $S \in \mathbb{R}$ that does not depend on K, for which we assume

$$\sqrt{2(K+1)} + 1 \le S, \tag{2.2}$$

i.e., all functions $\varphi_{\mathbf{k}}$ essentially vanish outside Ω. The function q is a multivariate polynomial in Chebyshev representation on the cube Ω, viz.,

$$q(x) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} T_{\mathbf{r}}(x/S), \qquad x \in \Omega,$$

where, in analogy to $\varphi_{\mathbf{k}}$, the functions $T_{\mathbf{r}}$ are tensor products of univariate Chebyshev polynomials,

$$T_{\mathbf{r}}(x/S) = T_{r_1}(x_1/S) \cdot \ldots \cdot T_{r_d}(x_d/S), \qquad x \in \Omega.$$

The reasoning behind the introduction of the cube Ω in combination with the assumptions (2.2) and $|\mathcal{K}| \gg |\mathcal{R}|$ has been given in Section 1.4: The matrix $\mathbf{Q}$ as defined in (2.1) stems from a spatial Galerkin discretization with a basis $\{\varphi_{\mathbf{k}}\}_{\mathbf{k}\in\mathcal{K}}$ of an equation that involves a multiplicative potential—which q is a placeholder for, obviously. We require the potential to be considerably smoother than the solution in a region where the solution does not essentially vanish. Now, when doing Chebyshev interpolation on a compact domain, this translates into the Chebyshev expansion consisting of much fewer terms than there are basis functions, which explains the requirement $|\mathcal{K}| \gg |\mathcal{R}|$; and we have to assume that the exact solution is supported within the cube for all times. The latter assumption means that the basis functions do not need to have support outside Ω, which explains (2.2). This joint smoothness assumption on the solution and the potential is for complexity reasons, as will soon become clear. Alternatively, we might want to work with a moving Galerkin basis to track the solution instead of in a basis localized around zero.

2.2 Direct operation with coordinate matrices

This section contains an efficient procedure to compute the action of what we call coordinate matrices on a vector in linear time only. The procedure is based on the orthogonality of the chosen Galerkin basis and its three-term recurrence, and it plays a crucial role in setting up the fast algorithm as done in the subsequent section.

2.2.1 One-dimensional approach

We begin with some simple considerations in 1D that will be generalized afterwards to arbitrary dimensions. The action of the one-dimensional $((K+1)\times(K+1))$-coordinate matrix X, $X_{jk} = (\varphi_j, x\varphi_k)$, on a vector $\mathbf{v} \in \mathbb{C}^{K+1}$ can be computed in $\mathcal{O}(K)$ operations only. Invoking the recurrence relation (1.10) together with the orthogonality of the Hermite functions, this is seen as follows: We write out a component of the resulting vector in a suitable way, viz.,

$$\begin{aligned}(X\mathbf{v})_j &= \sum_{k=0}^{K} (\varphi_j, x\varphi_k) v_k \\ &= \sum_{k=0}^{K} \sqrt{\frac{k+1}{2}} \underbrace{(\varphi_j, \varphi_{k+1})}_{=\delta_{j,k+1}} v_k + \sum_{k=0}^{K} \sqrt{\frac{k}{2}} \underbrace{(\varphi_j, \varphi_{k-1})}_{=\delta_{j,k-1}} v_k \\ &= \sqrt{\frac{j}{2}} v_{j-1} + \sqrt{\frac{j+1}{2}} v_{j+1}\end{aligned} \tag{2.3}$$

for all $j = 0, \ldots, K$, where we use $v_{-1} = v_{K+1} = 0$. The recurrence relation replaces the position operator x applied to a basis function by a finite number of shifted and appropriately weighted basis functions, and the orthogonality then makes the outer sum break down. A single component of the resulting vector is thus computed in $\mathcal{O}(1)$ from weighted components of the original vector, and the whole resulting vector, being of size $\mathcal{O}(K)$, can therefore be computed in $\mathcal{O}(K)$ componentwise.

2.2.2 Generalization to higher dimensions

This procedure generalizes to higher dimensions. We define coordinate representations of the position operators with respect to every coordinate,

$$\mathbf{X}^{(\alpha)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, x_\alpha \varphi_{\mathbf{k}}), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \qquad \alpha = 1, \ldots, d. \tag{2.4}$$

By the tensor product structure of the Galerkin basis, for a fixed coordinate α, we can repeat the argument given in 1D to obtain

$$\big(\mathbf{X}^{(\alpha)}\mathbf{v}\big)_{\mathbf{j}} = \sqrt{\frac{j_\alpha}{2}} v_{\mathbf{j}-\mathbf{e}_\alpha} + \sqrt{\frac{j_\alpha+1}{2}} v_{\mathbf{j}+\mathbf{e}_\alpha} \tag{2.5}$$

for all $\mathbf{j} \in \mathcal{K}$, where $\mathbf{e}_\alpha$ is the αth unit vector and $v_{\mathbf{j}-\mathbf{e}_\alpha} = 0$ if $j_\alpha = 0$ and $v_{\mathbf{j}+\mathbf{e}_\alpha} = 0$ if $\mathbf{j} + \mathbf{e}_\alpha$ happens to fall outside $\mathcal{K}$. Doing this for all components, we obtain $\mathbf{X}^{(\alpha)}\mathbf{v}$ via (2.5) in $\mathcal{O}(|\mathcal{K}|)$ operations. An algorithmic description is given in Algorithm 3. The matrices $\mathbf{X}^{(\alpha)}$ are never actually assembled. If the index set $\mathcal{K}$ is not closed under componentwise

decrements (cf. (1.15)), the first term in (2.5) also vanishes in case $\mathbf{j}-\mathbf{e}_\alpha \notin \mathcal{K}$, even though $j_\alpha \geq 1$, and Algorithm 3 needs to be adjusted accordingly.

Algorithm 3: Computation of

$$\mathbf{w} = \mathbf{X}^{(\alpha)}\mathbf{v}$$

in $\mathcal{O}(|\mathcal{K}|)$ using the direct operation given in (2.5).

```
function w = directoperation(α, v)
for j ∈ K do
    w_j := { (1/√2) v_{j+e_α},                          j_α = 0,
           { √(j_α/2) v_{j−e_α},                        j + e_α ∉ K,
           { √(j_α/2) v_{j−e_α} + √((j_α+1)/2) v_{j+e_α},   else.
```

2.3 Algorithmic description

Having defined the above coordinate matrices, the idea underlying the fast algorithm can now be motivated in a simple way. We introduce the notations

$$\mathbf{j}^{(\neg\alpha)} = (j_1, \ldots, j_{\alpha-1}, j_{\alpha+1}, \ldots, j_d) \in \mathbb{N}^{d-1}, \qquad \mathbf{j} = (j_1, \ldots, j_d) \in \mathbb{N}^d, \tag{2.6}$$

for the deletion of the α-th component from a given multi-index $\mathbf{j}$, and

$$\mathbf{j} \overset{\alpha}{\leftarrow} k = (j_0, \ldots, j_{\alpha-1}, k, j_{\alpha+1}, \ldots, j_d) \in \mathbb{N}^d, \qquad \mathbf{j} = (j_1, \ldots, j_d) \in \mathbb{N}^d, \tag{2.7}$$

for the replacement of the αth component in a given multi-index $\mathbf{j}$ by a given integer k.

2.3.1 Insertion of coordinate matrices into the potential

Consider the following two examples of q. In case $q(x) = x_\alpha$, for some coordinate α, the Galerkin matrix trivially reduces to the αth coordinate matrix, i.e., $\mathbf{Q} = \mathbf{X}^{(\alpha)}$. In case $q(x) = x_\alpha x_\beta$, for some distinct coordinates α and β, we find $\mathbf{Q} = \mathbf{X}^{(\alpha)}\mathbf{X}^{(\beta)} = \mathbf{X}^{(\beta)}\mathbf{X}^{(\alpha)}$, which is seen from

$$\begin{aligned}
\big(\mathbf{X}^{(\alpha)}\mathbf{X}^{(\beta)}\big)_{\mathbf{jk}} &= \sum_{\mathbf{m}\in\mathcal{K}} (\varphi_{\mathbf{j}}, x_\alpha\varphi_{\mathbf{m}})(\varphi_{\mathbf{m}}, x_\beta\varphi_{\mathbf{k}}) \\
&= \sum_{\mathbf{m}\in\mathcal{K}} (\varphi_{\mathbf{j}^{(\neg\alpha)}}, \varphi_{\mathbf{m}^{(\neg\alpha)}})(\varphi_{\mathbf{m}^{(\neg\beta)}}, \varphi_{\mathbf{k}^{(\neg\beta)}})(\varphi_{j_\alpha}, x_\alpha\varphi_{m_\alpha})(\varphi_{m_\beta}, x_\beta\varphi_{k_\beta}) \\
&= \sum_{\mathbf{m}\in\mathcal{K}} \delta_{\mathbf{m},\mathbf{j}\overset{\alpha}{\leftarrow}k_\alpha}\delta_{\mathbf{m},\mathbf{k}\overset{\beta}{\leftarrow}j_\beta}(\varphi_{j_\alpha}, x_\alpha\varphi_{m_\alpha})(\varphi_{m_\beta}, x_\beta\varphi_{k_\beta}) \\
&= (\varphi_{\mathbf{j}}, x_\alpha x_\beta\varphi_{\mathbf{k}}) = \mathbf{Q}_{\mathbf{jk}}
\end{aligned}$$

and the fact that the coordinate matrices are symmetric matrices. The argument no longer works in case there occur squares or higher powers of some $\mathbf{X}^{(\alpha)}$. Still, carrying on with

the above procedure for an arbitrary multivariate polynomial q, we tentatively compute $\mathbf{Qv}$ by formally inserting the coordinate matrices $\mathbf{X}^{(\alpha)}$ in place of the position coordinates x_α into the polynomial q, for all $1 \leq \alpha \leq d$, and approximate the matrix-vector product $\mathbf{Qv}$ by the product of the matrix due to formal insertion of the coordinate matrices into the polynomial q times $\mathbf{v}$, as proposed in [23]. We denote this formal insertion by

$$\mathbf{Qv} \approx q(\mathbf{X})\mathbf{v} = \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} T_{\mathbf{r}}\big(\tfrac{1}{S}\mathbf{X}\big)\mathbf{v} = \textstyle\sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} \prod_{\alpha=1}^{d} T_{r_\alpha}\big(\tfrac{1}{S}\mathbf{X}^{(\alpha)}\big)\mathbf{v}. \tag{2.8}$$

2.3.2 Using the Chebyshev recurrence: the 1D case

How can (2.8) be computed efficiently using the coordinate matrices? To see this, consider again the 1D case first. In 1D, using the Chebyshev recurrence relation (1.25), viz.,

$$\begin{aligned} T_0\big(\tfrac{1}{S}X\big)\mathbf{v} = \mathbf{v}, \quad T_1\big(\tfrac{1}{S}X\big)\mathbf{v} = \tfrac{1}{S}X\mathbf{v}, \\ T_{r+1}\big(\tfrac{1}{S}X\big)\mathbf{v} = 2\tfrac{1}{S}XT_r\big(\tfrac{1}{S}X\big)\mathbf{v} - T_{r-1}\big(\tfrac{1}{S}X\big)\mathbf{v}, \quad r \geq 1, \end{aligned} \tag{2.9}$$

the right-hand side of (2.8) is obtained in $\mathcal{O}(RK)$ operations: Given vectors $T_r\big(\frac{1}{S}X\big)\mathbf{v}$ and $T_{r-1}\big(\frac{1}{S}X\big)\mathbf{v}$, we employ (2.9) in combination with the direct operation technique (2.3) to compute $\mathbf{w} = T_{r+1}\big(\frac{1}{S}X\big)\mathbf{v}$ in $\mathcal{O}(K)$, then multiply $\mathbf{w}$ with $\hat{q}_r$, and proceed with the next term.

This 1D approach generalizes to arbitrary dimensions in two different ways: There is a non-recursive version of the fast algorithm that is optimal with respect to space complexity (i.e., it requires less storage), but suffers from a weaker time complexity. The second, recursive version is not space-, but time-optimal; thus, it exhibits the converse complexity behavior. The first version is due to [12], while the second version constitutes a modification thereof and has been proposed in [13] (for a Legendre basis, though). In both versions, the idea is to compute the right-hand side of (2.8) termwise.

2.3.3 First version

As for the first, non-recursive version, consider the algorithmic description given in Algorithm 4. We factorize

$$\begin{aligned} \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} \prod_{\alpha=1}^{d} T_{r_\alpha}\big(\tfrac{1}{S}\mathbf{X}^{(\alpha)}\big)\mathbf{v} = \\ \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} \left(T_{r_1}\big(\tfrac{1}{S}\mathbf{X}^{(1)}\big) \cdot \big(T_{r_2}\big(\tfrac{1}{S}\mathbf{X}^{(2)}\big) \cdot \big(\ldots \cdot \big(T_{r_d}\big(\tfrac{1}{S}\mathbf{X}^{(d)}\big)\mathbf{v}\big)\big)\big)\right) \end{aligned} \tag{2.10}$$

and compute the action of T_{r_d} evaluated at $\frac{1}{S}\mathbf{X}^{(d)}$ on $\mathbf{v}$ first, which is done using the recursive 1D procedure as given above in combination with (2.5) (lines **6** to **16**), then operate in the same way on the resulting vector with $T_{r_{d-1}}$ evaluated at $\frac{1}{S}\mathbf{X}^{(d-1)}$, and so forth (loop in line **5**). We then move on to the next term in the Chebyshev expansion according to the linear ordering of $\mathcal{R}$ (loop in line **3**). Apparently, there is no analog of the 1D Chebyshev recurrence in higher dimensions (or a competitive analog of Clenshaw's algorithm, for that matter) that allows to compute $T_{\mathbf{r}}\big(\frac{1}{S}\mathbf{X}\big)$ from an $\mathcal{R}$-independent number of insertions of the coordinate matrices into tensor products of other Chebyshev polynomials each bearing

an index that is lower with respect to a specific ordering of $\mathcal{R}$. For this reason, we factorize the tensor products as in (2.10).

Algorithm 4: First, non-recursive version of the fast algorithm for

$$\mathbf{w} = q(\mathbf{X})\mathbf{v}$$

for given $\mathbf{v}$, where $q(\mathbf{X})$ is given as in (2.8). Figure taken from [12] and modified.

```
1  function w = fastalgorithm_V1(v)
2  w := 0
3  for r ∈ R do
4      v_+ := v
5      for α = 1 to d do
6          if r_α > 0 then
7              v_− := v_+
8              w := directoperation(α, v_+)          use Algorithm 3: O(|K|)
9              v_+ := (1/S) w
10             for r = 2 to r_α do
11                 temp := v_+
12                 w := directoperation(α, v_+)      use Algorithm 3: O(|K|)
13                 v_+ := 2 (1/S) w − v_−            Chebyshev recurrence (2.9)
14                 v_− := temp
15         else
16             v_+ := v
17     w := w + q̂_r v_+
```

2.3.4 Second version

In contrast to the first version, the second version is recursive. Consider the pseudocode formulation given in Algorithm 5. The first term in the right-hand side expansion of (2.8) is the one corresponding to $\mathbf{r} = \mathbf{0}$, which is also the initial term in this termwise procedure. Starting from $r = 0$ and $\alpha = 1$, we carry out a single step of the 1D Chebyshev recurrence (2.9) in combination with (2.5) (lines **3** to **11**) and then carry on recursively with $\mathbf{r} \overset{\alpha}{\Leftarrow} r$ instead of $\mathbf{r}$ and with $\alpha+1$ instead of α (lines **13**, **15**). Having reached $\alpha = d$, we multiply with $\hat{q}_{\mathbf{r}}$, where $\mathbf{r}$ indicates the currently considered term in (2.8), and sum up the result (line **17**). Figure 2.1 shows the order of function calls of $\mathsf{fastalgorithm}_{\mathrm{V2}}(\ldots, \alpha, \mathbf{r})$ according to the choices of $\mathbf{r}$ taken by the algorithm together with how the recursions are nested. It becomes visible that Algorithm 5 adds up matrix-vector products termwise in lexicographical order.

Algorithm 5: Second, recursive version of the fast algorithm for

$$\mathbf{w} = q(\mathbf{X})\mathbf{v}$$

for a given vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ starting from $\mathbf{w} = \mathbf{0}$, $\alpha = 1$, and $\mathbf{r} = (0, \ldots, 0) \in \mathbb{N}^d$, where $q(\mathbf{X})$ is given as in (2.8). Figure taken from [13] and modified.

```
1  function w = fastalgorithmV2(v, w, α, r)
2  for r = 0 to R do
3      if r = 0 then
4          v_- := v
5      else if r = 1 then
6          v_+ := (1/S) directoperation(α, v)              use (2.5): O(|K|)
7      else
8          temp := v_+
9          temp2 := directoperation(α, v_+)                use Algorithm 3: O(|K|)
10         v_+ := 2(1/S) temp2 − v_-                       Chebyshev recurrence (2.9)
11         v_- := temp
12     w̃ := { v_-,  r = 0,
            { v_+,  else,
13     r := r ←α r
14     if α < d then
15         w := fastalgorithmV2 (w̃, w, α+1, r)             recursion: next coordinate
16     else
17         w := w + q̂_r w̃                                  last coordinate: sum up
```

2.3.5 Reduced index sets for polynomial approximation

If the index set $\mathcal{R}(d, R)$ is also reduced, Algorithm 5 needs to be modified slightly, whereas Algorithm 4 remains unaltered. We introduce the following notation to comment on this modification: For a given index $\mathbf{r} \in \mathcal{R}(d-1, R)$ taken from a $(d-1)$-dimensional analog of $\mathcal{R}(d, R)$, we define

$$v^{\max}_{\mathcal{R}(d,R)}(\mathbf{r}) = \arg\max_{r\in\mathbb{N}} \big\{ \mathbf{r} \overset{?}{\leftarrow} r \in \mathcal{R}(d, R) \big\}, \qquad \mathbf{r} \in \mathcal{R}(d-1, R), \tag{2.11}$$

to be the maximum over all integer values r such that, when inserted into $\mathbf{r}$ at an arbitrary position, the resulting vector is in $\mathcal{R}(d, R)$. Trivially,

$$v^{\max}_{\mathcal{R}_{\text{full}}}(\mathbf{r}) = R, \qquad \mathbf{r} \in \mathcal{K}_{\text{full}}(d-1, R).$$

For a hyperbolically reduced index set (1.16), an analytic expression is

$$v^{\max}_{\mathcal{K}_{\text{hyp}}}(\mathbf{r}) = \left\lfloor (R+1) / \prod_{\alpha=1}^{d-1} (r_\alpha + 1) \right\rfloor, \qquad \mathbf{r} \in \mathcal{R}_{\text{hyp}}(d-1, R),$$

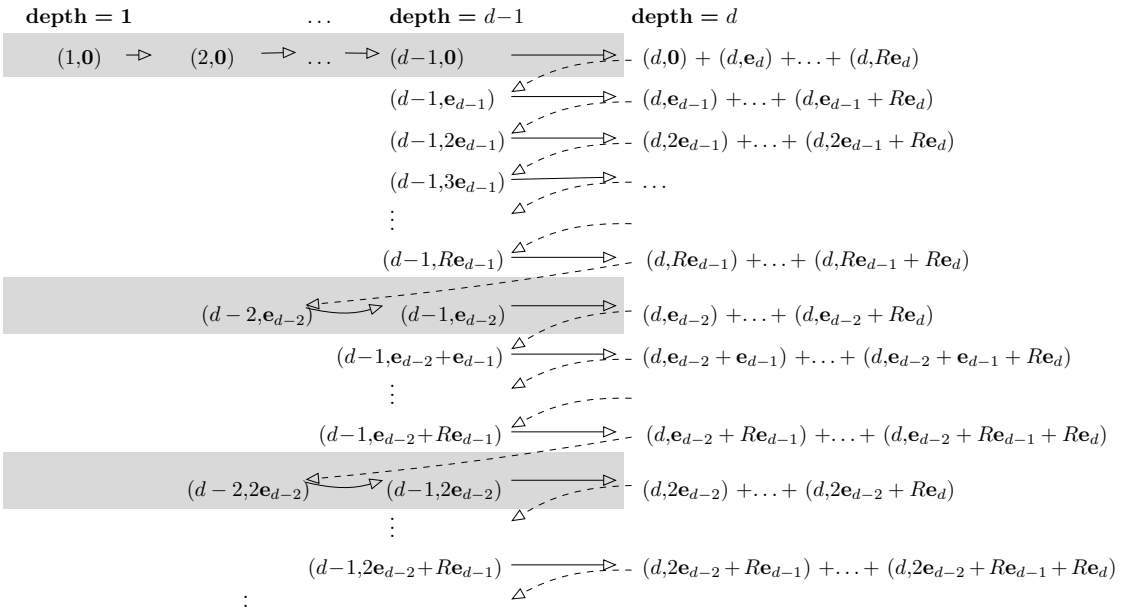


Figure 2.1: Function calls of $(\alpha, \mathbf{r}) = \mathsf{fastalgorithm}_{\mathrm{V2}}(\ldots, \alpha, \mathbf{r})$ according to the order of chosen $\mathbf{r}$ starting from $(1,\mathbf{0})$. Solid arrows represent recursive function calls of $\mathsf{fastalgorithm}_{\mathrm{V2}}$, while dashed arrows stand for the termination of a recursive call. Plus signs represent addition of terms computed within one function call; see line **17** of Algorithm 5. Clearly, the underlying order of $\mathbf{r}$ is lexicographical. The gray background color indicates that the recursive depth is always equal to d.

where $\lfloor\cdot\rfloor$ is the floor function. For an additively reduced index set (1.21), we find

$$v^{\max}_{\mathcal{R}_{\mathrm{add}}}(\mathbf{r}) = R - \sum_{\alpha=1}^{d-1} r_\alpha, \qquad \mathbf{r} \in \mathcal{R}_{\mathrm{add}}(d-1, R).$$

The required modification consists in replacing R in line **2** of Algorithm 5, which is only true in case $\mathcal{R} = \mathcal{R}_{\mathrm{full}}$, by

$$v^{\max}_{\mathcal{R}}(\mathbf{r}^{(\neg\alpha)}),$$

where $v^{\max}_{\mathcal{R}}$ is defined according to the chosen reduction of $\mathcal{R}$.

2.4 Complexity

Time and space complexity of both versions can readily be extracted from the above pseudocode formulations.

2.4.1 Space complexity

The first version of the fast algorithm requires the storage of a fixed, d- and $\mathcal{R}$-independent number of vectors of size $|\mathcal{K}|$ each. Algorithm 5 is less favorable with respect to space complexity. Since it has maximum recursion depth d, we need to keep a number of vectors in storage that is of the size $\mathcal{O}(d)$, but does not depend on $\mathcal{R}$ either. This can be seen from Figure 2.1.

2.4.2 Time complexity

Accounting for the numbers of operations, Algorithm 5 proves to be superior over the first version: Both algorithms proceed termwise. For every term bearing an index $\mathbf{r}$, Algorithm 4 invokes the 1D Chebyshev recurrence (2.9) up to order r_α together with an application of the direct operation procedure (2.5) in every recursion step, for all choices of α, which yields computational costs of $\mathcal{O}(|\mathbf{r}||\mathcal{K}|)$. Summing up, the overall computational costs for the matrix-vector product amount to

$$\mathcal{O}\big(\sum_{\mathbf{r}\in\mathcal{R}} |\mathbf{r}||\mathcal{K}|\big).$$

In Algorithm 5, the recursion is used in a more efficient way. For every choice of $\mathbf{r}$, the 1D Chebyshev recursion (2.9) is invoked exactly once. This yields overall computational costs of only

$$\mathcal{O}(|\mathcal{R}||\mathcal{K}|).$$

We can roughly estimate the contribution of the polynomial degree to the time complexity of Algorithm 4 by

$$\sum_{\mathbf{r}\in\mathcal{R}} |\mathbf{r}| \leq \sum_{\mathbf{r}\in\mathcal{R}} dR \leq dR|\mathcal{R}|.$$

Since we assume $|\mathcal{R}| \ll |\mathcal{K}|$ and $R \ll K$, both versions scale essentially linearly with $|\mathcal{K}|$.

2.5 Comments on implementation

As stated in Section 1.3, for the high-dimensional objects we deal with, we adhere to the notion of "vectors" and "matrices" with entries being referenced by multi-indices, where the underlying index sets are linearly ordered. The order of choice is the lexicographical order. When implementing the above methods, it is not in general advisable to represent what we see as vectors or matrices indexed over multi-index sets as multidimensional arrays. Instead, we treat vectors as 1D arrays and matrices as 2D arrays, whatever the underlying dimension d of the problems happens to be. Every row or every column in such a vector or a matrix, respectively, is uniquely referenced by a multi-index $\mathbf{k}$ according to the linear order of the set $\mathcal{K}$. We are thus confronted with the issue of determining the linear address of $\mathbf{k}$, i.e., the corresponding row or column number. The present section is a brief discussion of how we opt to do the mapping between multi-indices and their linear address in a given multi-index set $\mathcal{K}$.

2.5.1 Linear addresses

We denote the linear address of a given multi-index $\mathbf{k}$ from a lexicographically ordered multi-index set $\mathcal{K} \subseteq \mathbb{N}^d$ by $\mathrm{lin}(\mathbf{k})$. Additionally, we need to obtain the linear addresses

$$\mathrm{lin}(\mathbf{k} \pm \mathbf{e}_\alpha), \qquad \alpha = 1, \ldots, d, \tag{2.12}$$

when using the direct operation procedure for $\mathbf{X}^{(\alpha)}$ as given in Section 2.2; see lines **8**, **12** in Algorithm 4 and lines **6**, **9** in Algorithm 5.

For a full index cube, addressing a given multi-index set in such a structure is rather trivial, viz.,

$$\mathrm{lin}(\mathbf{k}) = \sum_{\alpha=1}^{d} k_\alpha (K+1)^{d-\alpha} + 1 \qquad \forall\, \mathbf{k} \in \mathcal{K}_{\mathrm{full}}(d,K).$$

The multi-indices (2.12) are then not difficult to obtain either: For every $r \in \mathbb{N}$ and for every coordinate α such that $\mathbf{k} \pm r\mathbf{e}_\alpha \in \mathcal{K}_{\mathrm{full}}$, respectively, we find

$$\begin{aligned}\mathrm{lin}(\mathbf{k} \pm r\mathbf{e}_\alpha) &= \sum_{\beta=1}^{d} k_\beta (K+1)^{d-\beta} + 1 - k_\alpha (K+1)^{d-\alpha} + (k_\alpha \pm r)(K+1)^{d-\alpha} \\ &= \mathrm{lin}(\mathbf{k}) + 1 \pm r(K+1)^{d-\alpha}.\end{aligned}$$

For a reduced index set, such a mapping might not be readily available, or we might prefer to avoid its computation at run time. For an additively reduced index set, linear addresses can be computed recursively. Briefly, the linear address of an index $\mathbf{k}$ in a d-dimensional additively reduced index set is the sum of all sizes of additively reduced index sets in dimension $d-1$ with thresholds $K-k$ for k ranging from 0 to k_1-1, plus the linear address of $\mathbf{k}^{(\neg 1)}$ in $\mathcal{K}(d-1, K-k_1)$. In 2D, there is an explicit formula. Formally,

$$\begin{aligned}\mathrm{lin}_{\mathcal{K}_{\mathrm{add}}(d,K)}(\mathbf{k}) &= \sum_{k=0}^{k_1-1} \binom{K-k+d-1}{d-1} + \mathrm{lin}_{\mathcal{K}_{\mathrm{add}}(d-1,K-k_1)}(\mathbf{k}^{(\neg 1)}) \\ &= \frac{K+d}{d}\binom{K+d-1}{d-1} - \frac{K+d-k_1}{d}\binom{K-k_1+d-1}{d-1} \\ &\quad + \mathrm{lin}_{\mathcal{K}_{\mathrm{add}}(d-1,K-k_1)}(\mathbf{k}^{(\neg 1)}) \\ \mathrm{lin}_{\mathcal{K}_{\mathrm{add}}(2,\tilde{K})}((k_1,k_2)) &= k_1(\tilde{K}+1) + k_2 - \sum_{k=0}^{k_1-1} k + 1.\end{aligned} \tag{2.13}$$

If one precomputes the values

$$\binom{K-\beta+d-1}{d-1}, \qquad 0 \le \beta \le K,$$

$\mathrm{lin}_{\mathcal{K}_{\mathrm{add}}(d,K)}(\mathbf{k})$ can be computed in $\mathcal{O}(d-2+k_{d-1})$ operations. For a hyperbolic reduction, we are not aware of an analogous formula. Instead, we propose a precomputing approach. This resolves the difficulty of evaluating a computationally expensive or even unavailable analytic function $\mathrm{lin}(\cdot)$. The following proceeding is not restricted to a specific kind of reduction.

2.5.2 Index manuals

We consider the task of computing the linear addresses of all increments or decrements by 1 in a single components for any $\mathbf{k} \in \mathcal{K}$. We set up a so-called *index manual*, where all indices are stored according to their linear order together with $\mathrm{lin}(\mathbf{k} \pm \mathbf{e}_\alpha)$, $\alpha = 1, \ldots, d$. The index manual is a 2D, $(|\mathcal{K}| \times (3d))$-array of integers.

Its lin($\mathbf{k}$)th row reads

$\mathbf{k}$	$\lin(\mathbf{k}-\mathbf{e}_1$	$\lin(\mathbf{k}+\mathbf{e}_1)$	...	$\lin(\mathbf{k}-\mathbf{e}_d)$	$\lin(\mathbf{k}+\mathbf{e}_d)$

where the first d columns contain $\mathbf{k}$, and the next $2d$ columns contain the linear addresses of the increments or decrements in a single component starting with $\alpha = 1$. If an increment or decrement does not exist in $\mathcal{K}$, we simply write 0. An example of an index manual for a hyperbolically reduced index set is shown in Figure 2.2, where we choose $d=3$ and $K=5$.

1	0	0	0	0	15	0	7	0	2
2	0	0	1	0	16	0	8	1	3
3	0	0	2	0	17	0	9	2	4
4	0	0	3	0	0	0	0	3	5
5	0	0	4	0	0	0	0	4	6
6	0	0	5	0	0	0	0	5	0
7	0	1	0	0	18	1	10	0	8
8	0	1	1	0	0	2	11	7	9
9	0	1	2	0	0	3	0	8	0
10	0	2	0	0	19	7	12	0	11
11	0	2	1	0	0	8	0	10	0
12	0	3	0	0	0	10	13	0	0
13	0	4	0	0	0	12	14	0	0
14	0	5	0	0	0	13	0	0	0
15	1	0	0	1	20	0	18	0	16
16	1	0	1	2	21	0	0	15	17
17	1	0	2	3	0	0	0	16	0
18	1	1	0	7	22	15	19	0	0
19	1	2	0	10	0	18	0	0	0
20	2	0	0	15	23	0	22	0	21
21	2	0	1	16	0	0	0	20	0
22	2	1	0	18	0	20	0	0	0
23	3	0	0	20	24	0	0	0	0
24	4	0	0	23	25	0	0	0	0
25	5	0	0	24	0	0	0	0	0

Figure 2.2: Index manual for a hyperbolically reduced index set with $d = 3$ and $K = 5$ in lexicographical order. The very first column (bold numbers) gives the row number. The first d columns (typewritten) contain the indices themselves, the remaining $2d$ columns show the linear addresses of $\mathbf{k} \pm \mathbf{e}_\alpha$. Their order is $\lin(\mathbf{k}-\mathbf{e}_1)$, $\lin(\mathbf{k}+\mathbf{e}_1)$, $\lin(\mathbf{k}-\mathbf{e}_2)$, $\lin(\mathbf{k}+\mathbf{e}_2)$, ..., $\lin(\mathbf{k}+\mathbf{e}_d)$. As an example, consider line **18**. Decrementing the first and second components by 1 leads to the indices in lines **7** and **15**, respectively. Incrementing them leads to lines **22** and **19**. Decrementing the last component would yield a negative component, incrementing it violates the hyperbolic reduction; the last two columns thus have 0 as their entries.

An algorithmic description for an efficient computation of the index manual both for a hyperbolic and an additive reduction is given in Algorithm 6. The basis idea is to browse the full index cube starting with $\mathbf{0}$, i.e., the very first index according to the lexicographical order (line **2**). If an index $\mathbf{k}$ under consideration belongs to the reduced set $\mathcal{K}$, we store it (line **7**). Then, for each coordinate α, we decrement the αth component of $\mathbf{k}$ by 1 and browse the already computed multi-indices for the decrement, i.e., descend from the current row until there's a match (lines **10** to **20**). Trivially, this also yields the linear address for the increment of the decrement (line **18**). If an index $\mathbf{k}$ does not belong to $\mathcal{K}$, seen from the right, we set all its components to the right of its first zero component to K, because incrementing one of these components will result in an index that does not belong to $\mathcal{K}$ either (lines **23** and **24**). This way, we avoid checking unnecessarily whether an index belongs to

$\mathcal{K}$ or not by skipping the appropriate indices in the full cube. Finally, we increment the current index (line **25**). In both the hyperbolic and the additive reduction, the index $K\mathbf{e}_1$ is the last index according to lexicographical order, giving us a stopping criterion (line **9**).

Algorithm 6: Computation of the index manual for chosen d and K and an arbitrarily reduced index set $\mathcal{K}(d,K) \subsetneq \mathcal{K}_{\text{full}}(d,K)$.

```
1  function man = indexmanual (d, K)
2  k := 0                                              start with first index
3  lin:= 1
4  continue:= true
5  while continue do
6      if k ∈ K then
7          man (lin,1 : d) := k                        store indices ∈ K
8          if k_1 = K then
9              continue:= false                        k = Ke_1 is the last index: stop
10         for α = 1 to d do
11             if k_α > 0 then
12                 k⁻ := k − e_α                       consider all decrements ...
13                 continue⁻:= true                    ... and search for their linear addresses
14                 lin⁻:= lin
15                 while continue⁻ do
16                     if man (lin⁻,1:d)=k⁻ then
17                         man (lin, d + 2α − 1) := lin⁻   store addresses
18                         man (lin⁻, d + 2α) := lin
19                         continue⁻:= false
20                     lin⁻:= lin⁻- 1
21         lin:= lin + 1
22     else
23         β := argmax_α{k_α ≠ 0}
24         k_{β+1} := ... := k_d := K
25     k := increment (k)                              increment k according to linear order
```

2.5.3 Complexity

We briefly comment on the complexity of Algorithm 6. Let $N(\mathbf{k},\alpha)$ be the number of iterations used in the innermost while loop (line **15**). We find $N(\mathbf{k},d) = 1$ and

$$N(\mathbf{k},\alpha) \leq \begin{cases} \mathcal{K}\left(d-\alpha, \left\lfloor \frac{K+1}{\prod_{\beta=1}^{\alpha} k_\beta + 1} - 1 \right\rfloor\right), & \mathcal{K} = \mathcal{K}_{\text{hyp}}, \\ \mathcal{K}\left(d-\alpha, K - \sum_{\beta=1}^{\alpha} k_\beta\right), & \mathcal{K} = \mathcal{K}_{\text{add}}, \end{cases}$$

for $\alpha < d$, and the overall number of iterations amounts to

$$\sum_{\mathbf{k}\in\mathcal{K}} \sum_{\alpha=1}^{d} N(\mathbf{k}, \alpha).$$

This yields sub-quadratic complexity behavior, but we are unable to provide an explicit bound. In Figure 2.3, we show the overall number of iterations for some choices of d and K and both kinds of index set reduction. The actual complexity appears to be similar to

$$d \cdot |\mathcal{K}|^{3/2}$$

for the choices of d and K we consider in this thesis. Actual computation times are shown in Table 2.1. All computations are carried out on a desktop computer with an Intel Core 2 Duo E8400 3.00 GHz processor with 4 GB RAM using an implementation in `C`. As the figures reveal, even for a choice as large as $K = 50$, precomputing the index manual is feasible. For an additive reduction, though, the combination of high dimensions and large bases can become tedious, and the usage of the explicit formula (2.13) might be preferable in some cases, although it worsens the run time. Since it consists of integers only, storing the index manual is not an issue either.

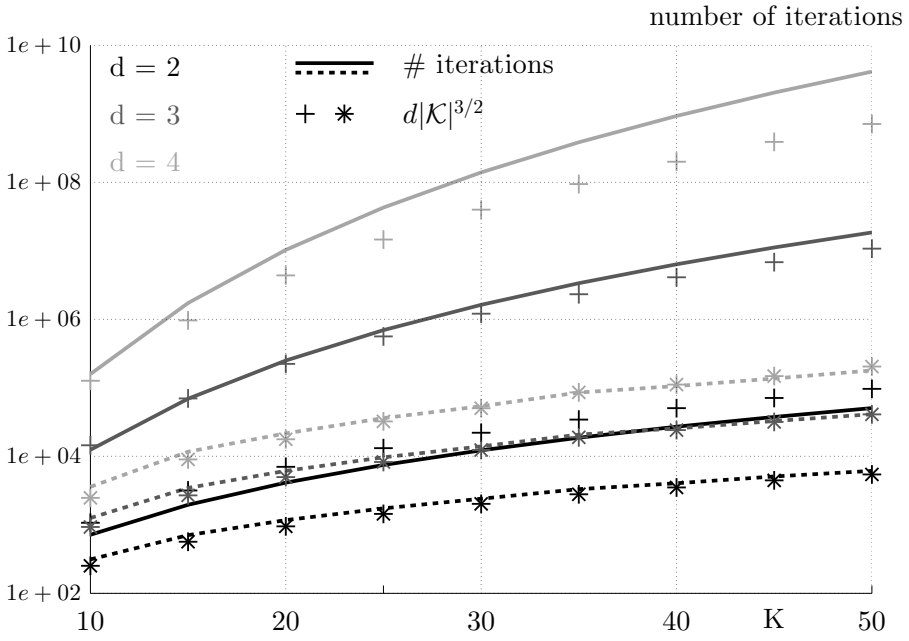


Figure 2.3: Actual number of iterations in Algorithm 6 compared to the size of the index set for different choices of d and K both for a hyperbolic (dashed lines, asterisks) and an additive reduction (solid lines, plus signs). Black, dark and light gray color stands for $d=2$, $d=3$, and $d=4$, respectively. The plus signs and asterisks represent $d|\mathcal{K}|^{3/2}$. Semi-logarithmic plot.

The linear addresses of the indices $\mathbf{r} = \mathbf{r} \overset{\alpha}{\leftarrow} r$ as they occur in line **13** of Algorithm 5 are not used in the computational process. Thus, we do not need to compute them.

$d\rightarrow$	2	3	4	5
hyperbolic	1.91e-04	1.62e-03	8.11e-03	3.33e-02
additive	1.36e-03	5.37e-01	1.23e+02	2.36e+04

Table 2.1: Actual computation times for the index manual in sec for $K = 50$ and varying choices of d. In case $d \geq 3$, the choice of $K = 50$ constitutes the upper bound for most of our subsequent experiments; see Chapter 5, in particular.

2.6 Relation to Gauß–Hermite quadrature

In Section 2.3, inserting the coordinate matrices into the polynomial potential to approximate the matrix-vector product $\mathbf{Qv}$ has been presented as a rather tentative idea. The error of this approximation is actually closely related to a suitable application of Gaussian quadrature for the approximation of the entries in $\mathbf{Q}$. This is discussed in the present section. Note that the fast algorithm never actually employs any kind of quadrature.

2.6.1 Preliminaries

We start with some general considerations about Gauß–Hermite quadrature, which, in 1D, is the instance of Gaussian quadrature that uses e^{-x^2} as a weight function and approximates integrals of weighted functions over the whole real line,

$$\int_{\mathbb{R}} f(x)e^{-x^2}\,\mathrm{d}x \approx \sum_{m=0}^{M} w_m f(\xi_m);$$

see, e.g., [32], Chapter 3.2. As for the choice of the order parameter M, if the corresponding quadrature nodes ξ_m are the zeros of H_{M+1} with corresponding weights w_m, the resulting quadrature formula $(w_m, \xi_m)_{m=0}^{M}$ is exact for polynomials of degree $\leq 2M+1$. In higher dimensions, we define

$$\begin{aligned} \xi_{\mathbf{m}} &= (\xi_{m_1}, \ldots, \xi_{m_d}), \\ \omega_{\mathbf{m}} &= \prod_{\alpha=1}^{d} \omega_{m_\alpha} = \prod_{\alpha=1}^{d} w_{m_\alpha} e^{\xi_{m_\alpha}^2}, \qquad \mathbf{m} \in \mathcal{M}_{\mathrm{full}}(d, M), \end{aligned} \tag{2.14}$$

which constitutes a product of 1D Gauß–Hermite quadrature formulas with $M+1$ nodes in every direction. We speak of $(M+1)$-nodes *full-product quadrature*. This d-dimensional quadrature rule is exact for polynomials of degree up to $2M+1$ in each variable, denoted henceforth by a subscript or a superscript "quad". The entries of $\mathbf{Q}$ might then be approximated by

$$\mathbf{Q}_{\mathbf{jk}} \approx (\mathbf{Q}^{\mathrm{quad}})_{\mathbf{jk}} = \sum_{\mathbf{m}\in\mathcal{M}} \omega_{\mathbf{m}} \varphi_{\mathbf{j}}(\xi_{\mathbf{m}}) q(\xi_{\mathbf{m}}) \varphi_{\mathbf{k}}(\xi_{\mathbf{m}}) \tag{2.15}$$

In the context of Section 2.1, further restrictions notwithstanding, the index set $\mathcal{K}$ has been introduced as any subset of the full index cube $\mathcal{K}_{\mathrm{full}}$. For the subsequent considerations, however, we assume $\mathcal{K}(d, K) = \mathcal{K}_{\mathrm{full}}(d, K)$. Additionally, let us now choose $M = K$ in the definition of full-product Gauß–Hermite quadrature (2.14).

2.6.2 Equivalence of formal insertion and quadrature

Given these preliminaries, we examine the relation between formal insertion of the coordinate matrices into the polynomially approximated potential and Gaussian quadrature. First, we define the matrix

$$\mathbf{U}_{\mathbf{jk}} = \sqrt{\omega_{\mathbf{j}}}\varphi_{\mathbf{k}}(\xi_{\mathbf{j}}) = \prod_{\alpha=1}^{d} \sqrt{\omega_{j_\alpha}}\varphi_{k_\alpha}(\xi_{j_\alpha}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K}, \tag{2.16}$$

bearing square roots of Gauß–Hermite weights multiplied by Hermite functions evaluated at corresponding Gauß–Hermite nodes in its entries.

Lemma 1. (orthogonality of $\mathbf{U}$)
The matrix $\mathbf{U}$ *is orthogonal.*

Proof. Since $\mathcal{K}(d,K) = \mathcal{K}_{\text{full}}(d,K)$, we can invoke the exactness properties of full-product Gauß-Hermite quadrature with exactly $K+1$ nodes in each direction together with the orthogonality of the Hermite functions to obtain

$$\begin{aligned}(\mathbf{U}^T\mathbf{U})_{\mathbf{jk}} &= \sum_{\mathbf{m}\in\mathcal{K}} \mathbf{U}_{\mathbf{mj}}\mathbf{U}_{\mathbf{mk}} = \sum_{\mathbf{m}\in\mathcal{K}} \omega_{\mathbf{m}}\varphi_{\mathbf{j}}(\xi_{\mathbf{m}})\varphi_{\mathbf{k}}(\xi_{\mathbf{m}}) \\ &= (\varphi_{\mathbf{j}},\varphi_{\mathbf{k}})_{\text{quad}} = (\varphi_{\mathbf{j}},\varphi_{\mathbf{k}}) = \delta_{\mathbf{jk}}.\end{aligned}$$

Thus, the matrix $\mathbf{U}$ has full rank,

$$\mathbf{U}\mathbf{v} = \mathbf{0} \Rightarrow \mathbf{U}^T\mathbf{U}\mathbf{v} = \mathbf{0} \Rightarrow \mathbf{v} = \mathbf{0},$$

and so has $\mathbf{U}\mathbf{U}^T$. Therefore,

$$\mathbf{U}^T\mathbf{U} = \mathbf{I} \Rightarrow \mathbf{U}\mathbf{U}^T\mathbf{U}\mathbf{U}^T = \mathbf{U}\mathbf{U}^T \Rightarrow \mathbf{U}\mathbf{U}^T = \mathbf{I}.$$

□

In case of a $\mathcal{K}$ being the full index cube, the matrix $\mathbf{U}$ provides an orthogonal diagonalization of each coordinate matrix $\mathbf{X}^{(\alpha)}$,

$$\mathbf{X}^{(\alpha)} = \mathbf{X}^{(\alpha)}_{\text{quad}} = \mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(\xi_{m_\alpha})\mathbf{U}, \qquad \alpha = 1,\ldots,d, \tag{2.17}$$

which is readily seen from

$$\begin{aligned}\mathbf{X}^{(\alpha)}_{\mathbf{jk}} &= (\varphi_{\mathbf{j}}, x_\alpha\varphi_{\mathbf{k}}) = (\varphi_{\mathbf{j}}, x_\alpha\varphi_{\mathbf{k}})_{\text{quad}} \\ &= \sum_{\mathbf{m}\in\mathcal{K}} \omega_{\mathbf{m}}\varphi_{\mathbf{j}}(\xi_{\mathbf{m}})\xi_{m_\alpha}\varphi_{\mathbf{k}}(\xi_{\mathbf{m}}) = \left(\mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(\xi_{m_\alpha})\mathbf{U}\right)_{\mathbf{jk}}\end{aligned}$$

using the exactness properties of full-product Gauß–Hermite quadrature once more. This diagonalization allows to prove the following lemma, which is the backbone of the error analysis given in Chapter 4.

Lemma 2. (equivalence lemma)
For an arbitrary multivariate polynomial q defined on $\Omega = [-S, S]^d$, we have

$$q(\mathbf{X}) = \mathbf{Q}^{\mathrm{quad}}.$$

The left-hand side denotes formal insertion of the coordinate matrices $\mathbf{X}^{(\alpha)} = (\varphi_{\boldsymbol{j}}, x_\alpha \varphi_{\boldsymbol{k}})$, $\boldsymbol{j}, \boldsymbol{k} \in \mathcal{K}_{\mathrm{full}}(d, K)$, defined over the full index cube in place of x_α into q; and

$$(\mathbf{Q}^{\mathrm{quad}})_{\boldsymbol{jk}} = \sum_{\boldsymbol{m} \in \mathcal{K}} \omega_{\boldsymbol{m}} \varphi_{\boldsymbol{j}}(\xi_{\boldsymbol{m}}) q(\xi_{\boldsymbol{m}}) \varphi_{\boldsymbol{k}}(\xi_{\boldsymbol{m}}), \qquad \boldsymbol{j}, \boldsymbol{k} \in \mathcal{K}_{\mathrm{full}}(d, K),$$

results from entrywise approximation of the representation matrix of q in the Galerkin basis $\{\varphi_{\boldsymbol{k}}\}_{\boldsymbol{k} \in \mathcal{K}_{\mathrm{full}}}$, i.e.,

$$\mathbf{Q}_{\boldsymbol{jk}} = (\varphi_{\boldsymbol{j}}, q \varphi_{\boldsymbol{k}}), \qquad \boldsymbol{j}, \boldsymbol{k} \in \mathcal{K}_{\mathrm{full}}(d, K),$$

by full-product Gauß–Hermite quadrature with exactly $K+1$ nodes in each direction. The matrix $q(\mathbf{X})$ is symmetric.

Proof. Consider an arbitrary multi-index $\mathbf{r} \in \mathbb{N}^d$. Using the orthogonal diagonalization (2.17), we find

$$\begin{aligned}
\mathbf{X}^{\mathbf{r}} &= \big(\mathbf{X}^{(1)}\big)^{r_1} \cdot \ldots \cdot \big(\mathbf{X}^{(d)}\big)^{r_d} \\
&= \big(\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}}(\xi_{m_1}) \mathbf{U}\big)^{r_1} \cdot \ldots \cdot \big(\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}}(\xi_{m_d}) \mathbf{U}\big)^{r_d} \\
&= \big(\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}} \big(\xi_{m_1}^{r_1}\big) \mathbf{U}\big) \cdot \ldots \cdot \big(\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}} \big(\xi_{m_d}^{r_d}\big) \mathbf{U}\big) \\
&= \mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}} \big(\xi_{\mathbf{m}}^{\mathbf{r}}\big) \mathbf{U} \\
&= \sum_{\mathbf{m} \in \mathcal{K}} \omega_{\mathbf{m}} \varphi_{\mathbf{j}}(\xi_{\mathbf{m}}) \xi_{\mathbf{m}}^{\mathbf{r}} \varphi_{\mathbf{k}}(\xi_{\mathbf{m}}) \\
&= (\varphi_{\mathbf{j}}, x^{\mathbf{r}} \varphi_{\mathbf{k}})_{\mathrm{quad}},
\end{aligned} \tag{2.18}$$

where the order of the factors in $\mathbf{X}^{\mathbf{r}}$ is actually arbitrary. In particular, symmetry of $\mathbf{X}^{\mathbf{r}}$ is seen from

$$\begin{aligned}
&\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}} \big(\xi_{\mathbf{m}}^{\mathbf{r}}\big) \mathbf{U} = \\
&\qquad \big(\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}} \big(\xi_{m_d}^{r_d}\big) \mathbf{U}\big) \cdot \ldots \cdot \big(\mathbf{U}^T \operatorname{diag}_{\mathbf{m} \in \mathcal{K}} \big(\xi_{m_1}^{r_1}\big) \mathbf{U}\big) = \big(\mathbf{X}^{\mathbf{r}}\big)^T.
\end{aligned}$$

Since this is true for any monomial, the claim follows. □

This kind of argument is common in the context of DVR techniques; see [59].

2.6.3 Error due to index set reduction

The fact that both $\mathcal{M}_{\mathrm{full}}(d, M)$ and $\mathcal{K}(d, K)$ are full index cubes as well as the choice $M = K$ are crucial ingredients for the proof of Lemma 2 to be valid. If $M < K$, the resulting quadrature does not exactly integrate all occurring Hermite functions. Then, the orthogonality of $\mathbf{U}$ is lost, and so is the diagonalization (2.17) of $\mathbf{X}^{(\alpha)}$. Thus, there needs to be at least a bijection between the quadrature and the basis indices. However, this obviates any reduction of $\mathcal{K}$ for Lemma 2 still to hold true. Nevertheless, for a reduced index set $\mathcal{K} \subsetneq \mathcal{K}_{\mathrm{full}}$, we adhere to the idea of inserting the coordinate matrices into the polynomial. Besides an approximation that is equivalent to the above Gauß–Hermite quadrature, this induces an additional error due to index set reduction. As the analysis given in Chapter 4 shall reveal, the latter error still behaves almost equally well as the error due to quadrature.

3 Time comparison

The fast algorithm as presented in Section 2.3 constitutes an efficient way to compute an approximation to the matrix-vector product

$$\mathbf{W}^{\mathrm{pol}}(t)\mathbf{v}, \qquad \mathbf{W}^{\mathrm{pol}}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}(\cdot,t)\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K},$$

where $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ is a vector and $W^{\mathrm{pol}}(\cdot,t) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) T_{\mathbf{r}}$ is a polynomial approximation to the potential $W(\cdot,t)$ due to Chebyshev interpolation over an index set $\mathcal{R}$. As seen in Section 2.4, the two version of the fast algorithm require

$$\sum_{\mathbf{r}\in\mathcal{R}} |\mathbf{r}| \cdot |\mathcal{K}| \qquad \text{and} \qquad |\mathcal{R}| \cdot |\mathcal{K}|$$

operations, respectively, for a single matrix-vector product. In the present chapter, we account for how good this actually is.

When trying to compute $\mathbf{W}^{\mathrm{pol}}(t)\mathbf{v}$, a first idea might be to assemble a quadrature approximation (in every time step), and then multiply it with the vector. We comment on this naive approach in Section 3.1. This sharply contrasts to matrix-free approaches. In Section 3.2, we present a matrix-free approach based on an efficient organization of the sum when formally writing out the matrix-vector product as

$$\big(\mathbf{W}^{\mathrm{pol}}(t)\mathbf{v}\big)_{\mathbf{j}} = \sum_{\mathbf{k}\in\mathcal{K}} \mathbf{W}^{\mathrm{pol}}(t)_{\mathbf{jk}} v_{\mathbf{k}}.$$

In Section 3.2.4, we compare this idea of so-called sequential summations to our fast algorithm. The chapter concludes with an overview over experimentally obtained computation times for all three approaches given in Section 3.3. Throughout this chapter, we shall occasionally skip the dependency on t.

3.1 Assembling the matrix

We briefly outline how the matrix

$$\begin{aligned}\mathbf{W}^{\mathrm{quad}}_{\mathbf{jk}}(t) &= (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}(t)\varphi_{\mathbf{k}})_{\mathrm{quad}} \\ &= \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) \prod_{\alpha=1}^{d} \underbrace{(\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})_{\mathrm{quad}}}_{\star}, \qquad \mathbf{j},\mathbf{k} \in \mathcal{K},\end{aligned} \tag{3.1}$$

could be assembled using $(M+1)$-nodes full-product Gauß–Hermite quadrature for every entry in the most effective way—while we strongly disencourage anyone to actually do so.

First, we need to obtain the Gaussian quadrature nodes ξ_m and weights w_m, $0 \leq m \leq M$, which can be done using, e.g., the procedures given in [72], Chapter 4.6. We assume these nodes and weights to be given. Second, the 1D Hermite and Chebyshev recurrences (see (1.10) and (1.25), respectively) yield the values

$$\begin{aligned} &\varphi_k(\xi_m), \quad 0 \leq k \leq K, \\ &T_r(\xi_m), \quad 0 \leq r \leq R, \end{aligned} \qquad 0 \leq m \leq M, \tag{3.2}$$

in $\mathcal{O}((R+K) \cdot M)$. Third, this enables us to compute the terms

$$(\varphi_j, T_r \varphi_k)_{\text{quad}}, \qquad 0 \leq j,k \leq K,\, 0 \leq r \leq R, \tag{3.3}$$

in $\mathcal{O}(K^2 \cdot R \cdot M)$. Even in case of W^{pol} being time-dependent, the above computations can be done in advance and only once. If the coefficients $\alpha_{\mathbf{r}}$ depend on time, the subsequent steps need to be repeated in every time step.

Given the terms (3.3), which we mark with a $\star$ in (3.1), we compute a single entry $\mathbf{W}^{\text{quad}}_{\mathbf{jk}}$ in $\mathcal{O}(|\mathcal{R}| \cdot d)$. Hence, the assembly of $\mathbf{W}^{\text{quad}}$ requires $\mathcal{O}\left(|\mathcal{K}|^2 \cdot |\mathcal{R}| \cdot d\right)$ operations, computation of the terms marked with $\star$ not taken into account. By virtue of the equivalence for the fast algorithm given in Lemma 2, we choose $M = K$. This yields overall costs that scale as

$$\begin{cases} |\mathcal{K}|^2 \cdot |\mathcal{R}| \cdot d, & \text{in every time step,} \\ K^3 \cdot R + (R+K) \cdot K, & \text{for precomputations.} \end{cases}$$

Using a full index set $\mathcal{K}$, the time-dependent contribution is dominant in case $d \geq 2$. Choosing a hyberbolical reduction, we can expect the precomputed part to dominate over a matrix assembly in a single time step (for a sufficiently small set $|\mathcal{R}|$.

Employing Smolyak sparse grid quadrature (see [33, 81, 91]) does not reduce the cost for assembling the matrix significantly. An adaptation of Smolyak quadrature to the increasingly oscillatory behavior of the high-order Hermite functions is discussed in [61], Chapter III.1.2., where it is pointed out that a sufficiently accurate sparse grid quadrature (i.e., yielding asymptotically the same quadrature error as a full-product quadrature) requires at least $\mathcal{O}(|\mathcal{K}|^2 \cdot M)$ evaluations of the potential anyway.

3.2 Sequential summations

The last section has been concerned with an explicit assembly of the matrix representation $\mathbf{W}^{\text{pol}}$ of the polynomial potential W^{pol}, which is a prohibitive choice due to the poor computational performance of any actual assembly procedure. The idea of using instead a matrix-free approach for the direct compuation of the product $\mathbf{W}^{\text{pol}}\mathbf{v}$ has been around in the chemical literature for many years; see the references given in the introduction to Part I. In a nutshell, the main difference between the approach given in this thesis and what has been done by chemists (to the extent we are going to receive the chemical literature) lies in the organization of the computations. We crucially make use of recurrence relations as they underly the chosen Galerkin approximation basis in order to speed up computations, most prominently so in the context of the direct application procedure for the coordinate matrices $\mathbf{X}^{(\alpha)}$; see Section 2.2. The following approach from the chemical literature, in contrast, does not employ given recurrence relations. Instead, the basic idea is to appropriately rewrite

the matrix-vector products and then ingeniously juggle with the occurring multi-indices. In Section 3.2.4, we shall give a more detailed comparison between the two approaches.

The following scheme is due to [11] for a full and due to [90] for an additively reduced index set, respectively. We present the ideas due to [11, 90] in more generality than the actual authors do, i.e., for any kind of reduced index set, and we add a brief derivation of the computational complexity, which has not been provided in [90], but only in the simpler setting of [11].

3.2.1 Basic idea

We consider the product of a matrix representation $\mathbf{T} \in \mathbb{C}^{|\mathcal{K}|\times|\mathcal{K}|}$ of an operator T,

$$\mathbf{T}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, T\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K},$$

with a vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$, where $\mathcal{K}(d,K) \subseteq \mathcal{K}_{\text{full}}(d,K)$ is a multi-index set, $\{\varphi_{\mathbf{k}}\}_{\mathbf{k}\in\mathbb{N}^d}$ is a tensor product $L^2(\mathbb{R}^d)$-orthonormal set, and T is in sum-of-products form (SOP), viz.,

$$T = \sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}} T_{\mathbf{r}} = \sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}} \prod_{\alpha=1}^{d} T_{r_\alpha}^{(\alpha)} \tag{3.4}$$

with a multi-index set $\mathcal{R}(d,R) \subseteq \mathcal{R}_{\text{full}}$, where $T_r^{(\alpha)}$ acts on functions of x_α only. Using (3.4), we write

$$\mathbf{T}_{\mathbf{jk}} = \sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}} \mathbf{T}^{\mathbf{r}}_{\mathbf{jk}}, \qquad \mathbf{T}^{\mathbf{r}}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, T_{\mathbf{r}}\varphi_{\mathbf{k}}) = \prod_{\alpha=1}^{d} \underbrace{\left(\varphi_{j_\alpha}, T_{r_\alpha}^{(\alpha)} \varphi_{k_\alpha}\right)}_{\star}$$

We assume the terms denoted by $\star$ to be either analytically known or otherwise easily computable using a 1D quadrature formula. As seen above, the matrix-vector product

$$\mathbf{w} = \mathbf{T}\mathbf{v}$$

could then be naively computed componentwise according to

$$(\mathbf{T}\mathbf{v})_{\mathbf{j}} = \sum_{\mathbf{k}\in\mathcal{K}} (\varphi_{\mathbf{j}}, T\varphi_{\mathbf{k}})\, v_{\mathbf{k}} = \sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}} \sum_{\mathbf{k}\in\mathcal{K}} \underbrace{\prod_{\alpha=1}^{d} \left(\varphi_{j_\alpha}, T_{r_\alpha}^{(\alpha)} \varphi_{k_\alpha}\right)}_{\mathcal{O}(d)} v_{\mathbf{k}}$$

in $|\mathcal{K}|^2 \cdot |\mathcal{R}| \cdot d$ operations altogether. The approach due to [11] constitutes a much faster way to do this. Assume, for a moment, $\mathcal{K} = \mathcal{K}_{\text{full}}$ and $\mathcal{R} = \mathcal{R}_{\text{full}}$. The SOP form of T allows to write the matrix-vector product termwise according to

$$\begin{aligned}
\mathbf{w}_{\mathbf{j}}^{\mathbf{r}} = (\mathbf{T}^{\mathbf{r}}\mathbf{v})_{\mathbf{j}} &= \sum_{\mathbf{k}\in\mathcal{K}} \mathbf{T}^{\mathbf{r}}_{\mathbf{jk}} v_{\mathbf{k}} = \sum_{\mathbf{k}\in\mathcal{K}} \prod_{\alpha=1}^{d} (\varphi_{j_\alpha}, p_\alpha^{r_\alpha} \varphi_{k_\alpha})\, v_{\mathbf{k}} \\
&= \sum_{k_1=0}^{K} \left(\varphi_{j_1}, T_{r_1}^{(1)} \varphi_{k_1}\right) \cdot \ldots \cdot \sum_{k_d=0}^{K} \left(\varphi_{j_d}, T_{r_d}^{(d)} \varphi_{k_d}\right) v_{\mathbf{k}} \\
&= \underbrace{\sum_{k_1=0}^{K} (T_{r_1}^{(1)})_{j_1 k_1} \cdot \ldots \cdot \sum_{k_d=0}^{K} (T_{r_d}^{(d)})_{j_d k_d} v_{\mathbf{k}}}_{d \text{ sums}},
\end{aligned} \tag{3.5}$$

with precomputed 1D matrices (slightly overloading the notation)

$$(T_r^{(\alpha)})_{jk} = \left(\varphi_j, T_r^{(\alpha)}\varphi_k\right), \qquad 0 \le j,k \le K, \quad 0 \le r \le R,$$

and, eventually,

$$\mathbf{w} = \sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}}\mathbf{w}^{\mathbf{r}}.$$

The key idea is the evaluate the sums in equation (3.5) *sequentially*—instead of naively using d loops with $(K+1)^d$ operations for $\mathbf{w}_{\mathbf{j}}^{\mathbf{r}}$ and, thus, $|\mathcal{R}|\cdot(K+1)^{2d}$ operations for $\mathbf{w}$ altogether. For the moment, let us write indices in brackets instead of as subscripts. We define

$$\begin{aligned}
v^{(1)}(k_1,\ldots,k_{d-1},j_d) &= \sum_{k_d=0}^{K} (T_{r_d}^{(d)})_{j_d k_d} v_{\mathbf{k}},\\
v^{(2)}(k_1,\ldots,k_{d-2},j_{d-1},j_d) &= \sum_{k_{d-1}=0}^{K} (T_{r_{d-1}}^{(d-1)})_{j_{d-1}k_{d-1}} v^{(1)}(k_1,\ldots,k_{d-1},j_d),\\
&\vdots\\
v^{(\alpha)}(k_1,\ldots,k_{d-\alpha},j_{d-\alpha+1},\ldots,j_d) &=\\
&\sum_{k_{d-\alpha+1}=0}^{K} (T_{r_{d-\alpha+1}}^{(d-\alpha+1)})_{j_{d-\alpha+1}k_{d-\alpha+1}} v^{(\alpha-1)}(k_1,\ldots,k_{d-\alpha+1},j_{d-\alpha+2},\ldots,j_d),\\
&\vdots\\
\mathbf{w}_{\mathbf{j}}^{\mathbf{r}} = v^{(d)}(j_1,\ldots,j_d) &= \sum_{k_1=0}^{K} (T_{r_1}^{(1)})_{j_1 k_1} v^{(d-1)}(k_1,j_2,\ldots,j_d).
\end{aligned}$$

Each vector

$$v^{(\alpha)}(k_1,\ldots,k_{d-\alpha},j_{d-\alpha+1},\ldots,j_d)$$

depends on d indices taking $K{+}1$ different values each and, for fixed values of these indices, $v^{(\alpha)}(k_1,\ldots,k_{d-\alpha},j_{d-\alpha+1},\ldots,j_d)$ is computed by summation over one index taking at most $K{+}1$ different values, depending on the sparsity of $T_{r_{d-\alpha+1}}^{(d-\alpha+1)}$. If the matrix $T_{r_{d-\alpha+1}}^{(d-\alpha+1)}$ contains at most $B_{r_{d-\alpha+1}}^{(d-\alpha+1)}$ non-zero entries in each row, the right-hand side sum over $k_{d-\alpha+1}$ consists of $T_{r_{d-\alpha+1}}^{(d-\alpha+1)}$ terms only.

Therefore, the scalar quantities

$$v^{(\alpha)}(k_1,\ldots,k_{d-\alpha},j_{d-\alpha+1},\ldots,j_d), \qquad 0 \le k_1,\ldots,k_{d-\alpha},j_{d-\alpha+1},\ldots,j_d \le K,$$

can be obtained in $B_{r_{d-\alpha+1}}^{(d-\alpha+1)}\cdot(K{+}1)^d$ operations. This is done sequentially for $\alpha = 1,\ldots,d$. After d blocks of $d{+}1$ nested loops each, the output vector $v^{(d)} = \mathbf{w}^{\mathbf{r}}$ is obtained in

$$\sum_{\alpha=1}^{d} B_{r_{d-\alpha+1}}^{(d-\alpha+1)}\cdot(K+1)^d$$

operations. For each choice of $\mathbf{r}$, only three vectors of size $(K+1)^d$ have to be stored. This is repeated for all $\mathbf{r} \in \mathcal{R}$, which yields overall computational costs for $\mathbf{w}$ of

$$\sum_{\mathbf{r}\in\mathcal{R}} \sum_{\alpha=1}^{d} B_{r_{d-\alpha+1}}^{(d-\alpha+1)} \cdot (K+1)^d \leq |\mathcal{R}| \cdot d \cdot (K+1) \cdot (K+1)^d = d \cdot (K+1) \cdot |\mathcal{R}| \cdot |\mathcal{K}|$$

operations—instead of a naive approach using $|\mathcal{R}| \cdot (K+1)^{2d}$ operations.

3.2.2 Algorithmic description

In the following, we provide an algorithmic description, slightly generalizing summations as given above: For fixed α, we want to compute

$$v^{(\alpha)}(k_1, \ldots, k_{d-\alpha}, j_{d-\alpha+1}, \ldots, j_d) = \sum_{k_{d-\alpha+1}=0}^{K} (T_{r_{d-\alpha+1}}^{(d-\alpha+1)})_{j_{d-\alpha+1}k_{d-\alpha+1}} v^{(\alpha-1)}(k_1, \ldots, k_{d-\alpha+1}, j_{d-\alpha+2}, \ldots, j_d)$$

with $d+1$ indices with index ranges

$$0 \leq k_\beta \leq k_\beta^{\max}, \qquad 1 \leq \beta \leq d-\alpha+1$$

and

$$0 \leq j_\beta \leq j_\beta^{\max}, \quad d-\alpha+1 \leq \beta \leq d,$$

and given values

$$(T_{r_{d-\alpha+1}}^{(d-\alpha+1)})_{j_{d-\alpha+1}k_{d-\alpha+1}}, \qquad \begin{aligned} 0 &\leq k_{d-\alpha+1} \leq k_{d-\alpha+1}^{\max}, \\ 0 &\leq j_{d-\alpha+1} \leq j_{d-\alpha+1}^{\max}, \\ 0 &\leq r_{d-\alpha+1} \leq r_{d-\alpha+1}^{\max}. \end{aligned}$$

We require

$$\max_\beta k_\beta^{\max} \leq K, \qquad \max_\beta j_\beta^{\max} \leq K.$$

In case of $\mathcal{K} = \mathcal{K}_{\text{full}}$, we have $k_\beta^{\max} = j_\beta^{\max} = K$ for all β. In case $\mathcal{R} = \mathcal{R}_{\text{full}}$, we have $r_\beta^{\max} = R$ for all β. Transitioning

$$\text{from} \quad \mathbf{v}^{(\alpha-1)} \in \mathbb{C}^{|\mathcal{K}(d-\alpha+1,K)|\cdot|\mathcal{K}(\alpha-1,K)|} \quad \text{to} \quad \mathbf{v}^{(\alpha)} \in \mathbb{C}^{|\mathcal{K}(d-\alpha,K)|\cdot|\mathcal{K}(\alpha,K)|},$$

the dependency on $k_{d-\alpha+1}$ is replaced by the dependency on $j_{d-\alpha+1}$. See Algorithm 7 for an algorithmic description of this transition using a single block of $d+1$ nested loops. A possible sparsity of the matrices $T_{r_{d-\alpha+1}}^{(d-\alpha+1)}$ has been taken into account;, see Line **11**. The number of loops depends on d, and the choice of α governs how many k- and j-loops there are. This suggests a recursive implementation. The linear addresses of $v^{(\alpha)}(\ldots)$ and $v^{(\alpha-1)}(\ldots)$ with indices as given in Lines **11** and **12** can be obtained using index manuals without further computational effort at run time.

This procedure is then repeated for all different values of α sequentially, which yields d blocks of $d+1$ loops each for the full index transition $\mathbf{k} \to \mathbf{j}$. Eventually, this is applied for each term in the sum (3.4). Algorithm 8 summarizes the proceeding.

Algorithm 7: Computation of vector $\mathbf{v}^{(\alpha)}$ from $\mathbf{v}^{(\alpha-1)}$ using $d+1$ nested loops. Reversing the order of the j-indices makes the presentation more readable when doing the index transition $\mathbf{k} \to \mathbf{j}$: k-indices are replaced with j-indices having the same subscript.

```
1  function v^(α) = computeblock(v^(α−1); T^(d−α+1)_{r_{d−α+1}} ∈ C^{k^max_{α−1} × j^max_{α−1}})
2  for k_1 = 0 to k_1^max do
3      for k_2 = 0 to k_2^max do
4          ⋱
5          for k_{d−α} = 0 to k_{d−α}^max do
6              for j_d = 0 to j_d^max do
7                  ⋱
8                  for j_{d−α+1} = 0 to j_{d−α+1}^max do
9                      temp:= 0
10                     for k_{d−α+1} = 0 s.t. (T^(d−α+1)_{r_{d−α+1}})_{j_{d−α+1} k_{d−α+1}} ≠ 0 do
11                         temp:= temp +
                           (T^(d−α+1)_{r_{d−α+1}})_{j_{d−α+1} k_{d−α+1}} v^(α−1)(k_1, ..., k_{d−α+1}, j_{d−α+2}, ..., j_d)
12                     v^(α)(k_1, ..., k_{d−α}, j_{d−α+1}, ..., j_d) := temp
```

The terms we need to collect or compute in advance are the non-zeros of

$$(T^{(\alpha)}_{r_\alpha})_{j_\alpha k_\alpha} = \left(\varphi_{j_\alpha}, T^{(\alpha)}_{r_\alpha}\varphi_{k_\alpha}\right), \qquad \begin{aligned} &0 \le j_\alpha \le j^{\max}_\alpha, \\ &0 \le k_\alpha \le k^{\max}_\alpha, \\ &0 \le r_\alpha \le r^{\max}_\alpha, \end{aligned} \qquad \alpha = 1, \dots, d. \tag{3.6}$$

In case these terms are not analytically available, we employ an M-point 1D quadrature formula requiring

$$(r^{\max}_\alpha + 1) \cdot (j^{\max}_\alpha + 1) \cdot (k^{\max}_\alpha + 1) \cdot M \le (R+1) \cdot (K+1)^2 \cdot M$$

operations. We have to store $\sum_{\alpha=1}^{d}(r^{\max}_\alpha + 1) \le d \cdot (R+1)$ matrices of size at most $(j^{\max}_\alpha + 1) \cdot (k^{\max}_\alpha + 1) \le (K+1)^2$, which is negligible.

3.2.3 Reduced index sets

We now consider the case $\mathcal{K} \subsetneq \mathcal{K}_{\text{full}}$ of a reduced index set. The following is due to [90], who discuss the case of an additively reduced index set $\mathcal{K}_{\text{add}}(d, K)$, which is of size $\binom{K+d}{d} \approx \frac{1}{d!}K^d$. In addition, we also consider a hyperbolic reduction $\mathcal{K}_{\text{hyp}}(d, K)$, where we retain only $\mathcal{O}((K+1)\ln(K+1)^{d-1})$ multi-indices.

The difference between reduced and full index sets is that, in the former case, the upper bounds $k^{\max}_\beta$ and $j^{\max}_\beta$ of the sums in Algorithm 7 now depend on previous summation

Algorithm 8: Doing the matrix-vector product

$$\mathbf{w} = \mathbf{T}\mathbf{v}$$

termwise with sequential summations in each term.

```
1 function w = sequ-sum(v; T^{(d-α+1)}_{r_{d-α+1}} ∈ ℂ^{k^max_α × j^max_α}, 0 ≤ r_α ≤ r^max_α, 1 ≤ α ≤ d)
2 w := 0
3 for r ∈ ℛ do
4 |   w^r := v
5 |   for α = 1 to d do
6 |   |   w^r := computeblock(w^r; T^{(d-α+1)}_{r_{d-α+1}})
7 |   w := w + t̂_r(t) · w^r
```

indices. In case of $\mathcal{K}_{\mathrm{add}}$, we have

$$k_\beta^{\max}(k_1,\ldots,k_{\beta-1}) = K - \sum_{\gamma=1}^{\beta-1} k_\gamma, \qquad \beta = 1,\ldots,d-\alpha+1,$$

$$j_\beta^{\max}(j_{\beta+1},\ldots,j_d) = K - \sum_{\gamma=\beta+1}^{d} j_\gamma, \qquad \beta = d,\ldots,d-\alpha+1,$$

for all $\alpha = 1,\ldots,d$. In case of $\mathcal{K}_{\mathrm{hyp}}$, we have

$$k_\beta^{\max}(k_1,\ldots,k_{\beta-1}) = \left\lfloor (K+1)/\prod_{\gamma=1}^{\beta-1}(k_\gamma+1) - 1 \right\rfloor, \qquad \beta = 1,\ldots,d-\alpha+1,$$

$$j_\beta^{\max}(j_{\beta+1},\ldots,j_d) = \left\lfloor (K+1)/\prod_{\gamma=\beta+1}^{d}(j_\gamma+1) - 1 \right\rfloor, \qquad \beta = d,\ldots,d-\alpha+1.$$

For fixed α, the number of different k-multi-indices

$$(k_1,\ldots,k_{d-\alpha})$$

equals the size of the multi-index set $\mathcal{K}(d-\alpha,K)$. Taking into account a possible sparsity of $T^{(d-\alpha+1)}_{r_{d-\alpha+1}}$, e.g., that it bears at most $B^{(d-\alpha+1)}_{r_{d-\alpha+1}} \le k^{\max}_{d-\alpha+1}(k_1,\ldots,k_{d-\alpha})$ non-zeros in each row, the overall number of k-multi-indices is

$$B^{(d-\alpha+1)}_{r_{d-\alpha+1}} \cdot \mathcal{K}(d-\alpha,K).$$

Analogously, the number of different j-multi-indices $(j_{d-\alpha+1},\ldots,j_d)$ equals the size of the multi-index set $\mathcal{K}(\alpha,K)$. The complexity of Algorithm 7 therefore reduces to

$$\begin{aligned} B^{(d-\alpha+1)}_{r_{d-\alpha+1}} &\cdot \binom{K+d-\alpha}{d-\alpha} \cdot \binom{K+\alpha}{\alpha} \\ &\approx B^{(d-\alpha+1)}_{r_{d-\alpha+1}} \frac{1}{(d-\alpha)!\alpha!} K^d = B^{(d-\alpha+1)}_{r_{d-\alpha+1}} \cdot \binom{d}{\alpha} |\mathcal{K}_{\mathrm{add}}(d,K)| \end{aligned}$$

and

$$B_{r_{d-\alpha+1}}^{(d-\alpha+1)} \cdot (K+1)\ln(K+1)^{d-\alpha-1} \cdot (K+1)\ln(K+1)^{\alpha-1} = B_{r_{d-\alpha+1}}^{(d-\alpha+1)} \cdot \frac{K+1}{\ln(K+1)} \cdot |\mathcal{K}_{\text{hyp}}(d,K)|$$

in case of $\mathcal{K}_{\text{add}}$ and $\mathcal{K}_{\text{hyp}}$, respectively. In case of $\mathcal{K}_{\text{add}}$, this leads to an overall complexity of

$$\sum_{\mathbf{r}\in\mathcal{R}} \sum_{\alpha=1}^{d} B_{r_{d-\alpha+1}}^{(d-\alpha+1)} \binom{d}{\alpha} |\mathcal{K}_{\text{add}}(d,K)| \tag{3.7}$$

for the matrix-vector product. In case of $\mathcal{K}_{\text{hyp}}$, the overall complexity is

$$\sum_{\mathbf{r}\in\mathcal{R}} \sum_{\alpha=1}^{d} B_{r_{d-\alpha+1}}^{(d-\alpha+1)} \frac{K+1}{\ln(K+1)} |\mathcal{K}_{\text{hyp}}(d,K)|. \tag{3.8}$$

3.2.4 Comparison to the fast algorithm

We compare the sequential summations approach to the fast algorithm as given in Chapter 2 when applied to a potential matrix-vector product with the potential being approximated by Chebyshev interpolation, i.e.,

$$W \approx \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}} T_{\mathbf{r}};$$

see Section 1.4. Using the above notation, this yields an operator in SOP form with $T_{r_\alpha}^{(\alpha)}$ being the univariate Chebyshev polynomial of degree r_α with respect to the αth coordinate. As for the number of non-zeros in the 1D corresponding banded matrices, with the help of the 1D Chebyshev recurrence (1.25), it is readily seen that

$$B_r^{(\alpha)} = \frac{1}{2}(r+1) \qquad \forall\, 1 \le \alpha \le d, \forall\, 0 \le r \le R, \tag{3.9}$$

where the factor $\frac{1}{2}$ comes from the fact that $(\varphi_j, T_r\varphi_k)$ vanishes if $j+r+k$ is odd.

There is a caveat to the subsequent considerations: Sequential summations are an appropriate tool for a broader range of applications than the fast algorithm because they do not suffer from the restrictions imposed in Section 1.4. In situations where the joint assumptions on the regularity of the potential W and the solution ψ do not hold, the concept is stil applicable. Furthermore, one might want to combine it with a suitably reduced quadrature. The fast algorithm, in contrast, then fails to do us any service. We shall be more precise on these issues in the following.

The table given in Table 3.1 summarizes the key properties of both algorithms. We briefly comment on them in terms of what is required for the approaches to work, how the computational speed-up is actually achieved, in what way the computation unwinds, and to what extend they might be generalized.

- Both approaches require the operator T to be in **SOP form**. First, in both approaches, the product with the corresponding matrix representation is done for each

	fast algorithm	**sequ. summations**
matrix-free	✓	✓
scales linear with $\lvert\mathcal{K}\rvert$	✓	✓, unless reduction is too radical
requires SOP	✓	✓
m-v product done exactly	✗	✓
allows for $R \geq K$	✗	✓
large $\lvert\mathcal{R}\rvert$ feasible	✓, but expensive	✓, but expensive
employs recurrences	✓	✗
uses precomputations	✓ (index manuals)	✓ (1D matrix entries)
allows for index manuals	✓	✓
full-product quadrature	not employed, but equivalent; see 2.6	adaptation in [90]
Smolyak quadrature	not employed	adaptation in [2, 3, 4, 5, 6]
generalizsations	✓; see Part II	✓, T not specified

Table 3.1: Key features of sequ. summations and the fast algorithm.

	fast algorithm	**sequential summations**
complexity	version 1: $\sum_{\mathbf{r}\in\mathcal{R}} \lvert\mathbf{r}\rvert\lvert\mathcal{K}\rvert$	full index set: $\sum_{\mathbf{r}\in\mathcal{R}} \lvert\mathbf{r}+\mathbf{1}\rvert\lvert\mathcal{K}_{\text{full}}\rvert$
	version 2: $\lvert\mathcal{R}\rvert \cdot \lvert\mathcal{K}\rvert$	additive reduction: $\sum_{\mathbf{r}\in\mathcal{R}} \sum_{\alpha=1}^{d} (r_\alpha+1) \frac{d!}{(d-\alpha)!\alpha!} \lvert\mathcal{K}_{\text{add}}\rvert$
		hyperbolic reduction: $\sum_{\mathbf{r}\in\mathcal{R}} \lvert\mathbf{r}+\mathbf{1}\rvert \frac{(K+1)}{\ln(K+1)} \lvert\mathcal{K}_{\text{hyp}}\rvert$

Table 3.2: Time complexity of sequ. summations and the fast algorithm, where $\mathbf{1} = (1, \ldots, 1) \in \mathbb{N}^d$.

term in the sum expansion separately. Second, doing a single term, both approaches make use of the term being a product of univariate operators only. In the context of sequential summations, the latter yields to favorable block structure consisting of $(d+1)$ sums over a single integer index each; see Algorithm 7. For the fast algorithm, this enables us to use the 1D recurrence for the polynomial representation basis, viz., the Chebyshev recurrence (1.25), for each coordinate.

- Sequential summations is a means to compute the matrix-vector product **exactly**, while the fast algorithm does so only **approximately**. For the approximation error to decay spectrally, the latter approach requires $K \gg R$, as will be shown in Chapter 4. For sequential summations, there is **no constraint on how R and K are related**.

- Both approaches can in principle deal with a **large index set $\mathcal{R}$** for the polynomial representation of the potential, however, this is not an advisable setting for either

approach. As for sequential summations, the reason is that there exists a more suitable modification using Smolyak sparse grid quadrature; see the subsequent comments. In case of $|\mathcal{R}|$ coming close to $|\mathcal{K}|$, the essentially linear scaling is lost for both approaches.

- As mentioned above, the **1D recurrence** (1.10) constitutes the core ingredient to the fast algorithm in order to obtain its computational speed-up. In sequential summations, recurrences are never employed, and the speed-up is due to a clever index management.

- Technically, sequential summations is a **matrix-free** approach, although, during the **precomputation** process that is necessary in order to obtain the values $(T_r^{(\alpha)})_{jk}$, matrix entries are de facto computed. No matrix is actually assembled. Using the fast algorithm, there are no precomputations at all (but for the index manuals), and no matrix entries are ever computed.

- For both approaches, using **index manuals** is possible. An alternative to index manuals for the problem of addressing components of the input and output vectors as needed in Algorithm 7 is mentioned in [3], where an example in $d = 12$ is considered with an additive reduction. Since computing the index manual for an additively reduced index set in as large a dimension as this is tedious (in contrast to a hyperbolic reduction), the approach chosen in [3] is preferable. The explicit formula (2.13) constitutes a further alternative. Once the index manual has been computed, though, there are no additional costs at run time.

- While the approximation due to the fast algorithm is closely related to a specific kind of entrywise **Gauß–Hermite quadrature** (see Section 2.6) no quadrature approximation is actually ever computed. As for sequential summations, if $|\mathcal{R}|$ becomes large, a different approach is proposed in [90] that uses full-product quadrature with an index set $\mathcal{M}$ for the nodes to approximate every entry $(\varphi_{\mathbf{j}}, W\varphi_{\mathbf{k}})$, and no polynomial interpolation of W at all. The basic idea is to write the entries as

$$\sum_{\mathbf{k}\in\mathcal{K}} (\varphi_{\mathbf{j}}, W\varphi_{\mathbf{k}})_{\mathrm{quad}} v_{\mathbf{k}} = \sum_{\mathbf{m}\in\mathcal{M}} \varphi_{\mathbf{j}}(\xi_{\mathbf{m}})\omega_{\mathbf{m}} W(\xi_{\mathbf{m}}) \sum_{\mathbf{k}\in\mathcal{K}} \varphi_{\mathbf{k}}(\xi_{\mathbf{m}}) v_{\mathbf{k}}$$

  and to do two sequential summations in a row: First, one uses an index transition $\mathbf{k} \to \mathbf{m}$, then multiplies with precomputed values $W(\xi_{\mathbf{m}})$ in an intermediate step, and eventually does another index transition $\mathbf{m} \to \mathbf{j}$. This approach is generalized to **Smolyak sparse grid quadrature** in [2] and has been further applied in [3, 4, 5, 6]. Accounting for the computational costs of an evaluation of W at a quadrature node depends, however, on how expensive an evaluation of W actually is. We shall not discuss this modifcation of sequential summations in the present thesis.

- In the sequential summations approach, it is not specified what the operator T is supposed to be. E.g., derivatives, possibly even with non-constant coefficents, are a viable choice. We only need to be able to compute the quantites $(T_r^{(\alpha)})_{jk}$—whatever the operators $T_r^{(\alpha)}$ might be. As for the fast algorithm, we will also open up a **wider range of generalizations** in Chapter 6 and, most notably, in Part II.

The most crucial difference between the two approaches lies, of course, in their respective **computational complexity**. Time complexity for doing a single matrix-vector product

is summarized in Table 3.2. The number of operations for sequential summations has been obtained via (3.9) in combination with (3.7) and (3.8).

- The fast algorithm scales linearly with $|\mathcal{K}|$—in both versions and with whatever underlying index set. Using sequential summations, the actual scaling depends on the choice of index set reduction, and linear scaling with respect to $|\mathcal{K}|$ gets more and more lost the more the basis is reduced.
- In the best-case scenario for sequential summations, which is the full index set, the computational complexity comes close to Version 1 of the fast algorithm. In general, the fast algorithm scales more favorable than sequential summations. The more radical the reduction, the more favorable the complexity ratio from the point of view of the fast algorithm.
- On the other hand, sequential summations are more economical with respect to space complexity. They require the storage of only four vectors of size $|\mathcal{K}|$; see Algorithm 8.

In the next section, we shall present computation times for both approaches. The findings will corroborate the above complexity analysis.

3.3 Performance tests

We conclude this chapter by giving actual computation times for

- an explicit assembly of $\mathbf{W}^{\text{quad}}$ and multiplication with a vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$,
- doing this matrix-vector product via sequential summations as devised in Section 3.2,
- and the fast algorithm (both versions) as developed in Chapter 2

using a simple test example. Our aim is to corroborate the theoretical predictions for the computation times from the previous section and Chapter 2 for different choices of index sets $\mathcal{K}$. As an underlying potential, we consider the stretched torsional potential

$$W(x) = \sum_{\alpha=1}^{d} (1 - \cos(x_\alpha / S)), \qquad x \in \Omega = [-S, S]^d, \quad S = 16. \tag{3.10}$$

A plot is shown in Figure 3.1. Respecting $S \geq \sqrt{2(K+1)}+1$, this choice of S allows for $K \leq 111$; see (1.23). As W is separable, we can approximate it by multidimensional Chebyshev interpolation over Ω with nodes along the coordinate axes only. If we choose $R+1$ nodes on each axis, one of them being the origin, the overall number of nodes is thus

$$|\mathcal{R}| = R + 1 + R(d-1) = Rd + 1.$$

For $R=8$ and $d = 2, 3, 4$, e.g., this yields interpolation errors

$$\|W - W^{\text{pol}}\|_\infty, \|W - W^{\text{pol}}\| \leq 5\text{e-}9.$$

To obtain this information, we have approximated the L^∞- and L^2-norms by transformation of the interpolation coefficients to function values on a spatial grid with 100 grid points per coordinate. Even for a hyperbolically reduced index set $\mathcal{K}$, we can thus easily meet the requirements (1.26) and (1.27) for the fast algorithm to be applicable.

In the following, we list computation times for the matrix-vector products $\mathbf{W}^{\text{quad}}\mathbf{v}$ (explicit assembly and sequential summations) and $W^{\text{pol}}(\mathbf{X})\mathbf{v}$ (fast algorithm). All figures have been obtained on a desktop computer with an Intel Core 2 Duo E8400 3.00 GHz processor with 4 GB RAM using an implementation in `C` in double precision arithmetics. The vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$ consists of random entries $v_{\mathbf{k}} \in [0,1]$ generated with `rand()`.

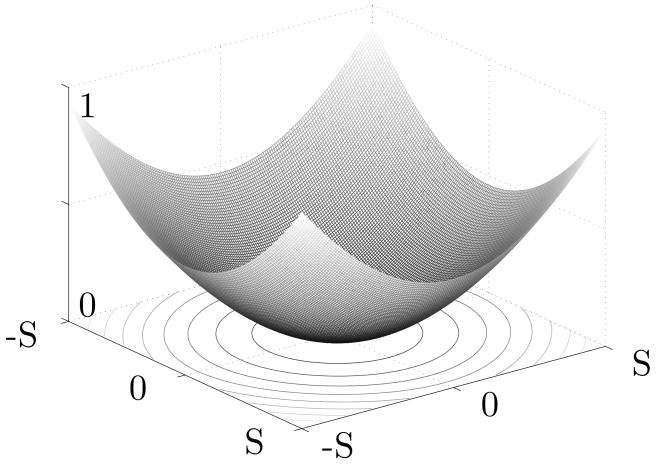


Figure 3.1: The torsional potential (3.10) for $d=2$.

Both for the explicit assembly and for sequential summations, we need to precompute the 1D quadrature formulas

$$(\varphi_j, T_r\varphi_k)_{\text{quad}} = \sum_{m=0}^{M} \omega_m \varphi_j(\xi_m) T_r(\xi_m) \varphi_k(\xi_m), \qquad 0 \le j,k \le K, \quad 0 \le r \le R;$$

see (3.2) and (3.6). In accordance with the equivalence stated in Lemma 2, we use $M=K$ when assembling the matrix $\mathbf{W}^{\text{quad}}$. For fixed r, the 1D integral matrix $(\varphi_j, T_r\varphi_k)_{jk}$ has at most $\frac{1}{2}(r+1)$ non-zeros it each row. In order to make use of this sparsity property when computing the innermost sum in Algorithm 7, these integrals need to be computed exactly, which necessitates

$$M = \left\lceil \frac{2K+R-1}{2} \right\rceil$$

for the sequential summations approach. Both for sequential summations and for the fast algorithm, we need to precompute index manuals. In the following experiments, neither the time necessary to precompute integrals nor the computational costs for setting up index manuals have been counted in.

Besides consuming enormous amounts of time, the computational burden for an explicit assembly of the matrix is equally due to its devastating memory requirements. In Table 3.3, for illustration purposes, we give the actual number of Galerkin basis functions for different choices of $\mathcal{K}$ together with the memory required to store the whole corresponding matrix $\mathbf{W}^{\text{quad}}$. For the sake of the argument, we have not taken into account zeros in $\mathbf{W}^{\text{quad}}$ or symmetries. We consider the case $d=3$ and thresholds for a small, an intermediate, and a large basis. For the sake of the argument, we have not taken into account zeros in $\mathbf{W}^{\text{quad}}$ or symmetries. As the figures reveal, for a full index cube, all but small bases and all but small choices of d are intractable. The choice $K=75$ yields an impressive memory requirement of 1.4 TB. For an additive reduction, using 4 GB RAM, we cannot hope to deal with large bases either, and the restriction to relatively low dimensions is already severe. Only a hyperbolic reduction seems to constitue a feasible way when trying to explicitly assemble the matrix. Anyway, assembling $\mathbf{W}^{\text{quad}}$ is obviously a poor idea.

$K \to$	25	50	75
full index cube	17576 2.3 GB	132651 131 GB	438976 1.4 TB
additive reduction	3276 81.9 MB	23426 4.1 GB	76076 41.3 GB
hyperbolic reduction	218 371.3 kB	573 2.5 MB	1000 7.6 MB

Table 3.3: Sizes $|\mathcal{K}|$ of Galerkin bases and explicitly assembled matrices $\mathbf{W}^{\text{quad}}$ for a full index cube and both additive and hyperbolic reductions in case $d=3$. We consider thresholds $K=25, 50, 75$ for a small, intermediate, and large basis. In each cell, the upper figure represents the number of basis functions, and the lower figure gives the approximate memory needed to store $|\mathcal{K}|^2$ 64-bit double precision numbers, i.e., for the whole matrix. On our machine, insufficient memory is expected for an additive reduction with $K=50$.

For the subsequent experiments, we start in reversed order with a hyperbolically reduced index set because only there can we hope to be able to assemble the corresponding matrix $\mathbf{W}^{\text{quad}}$ up to fairly large dimensions and with fairly large Galerkin bases in order to study the time behavior of the three approaches. We consider the additive reduction and the full index cube afterwards. In Figure 3.2, we show both computation times and ratios of computation times for an explicit assembly vs. sequential summations vs. the fast algorithm, where black color always represents the explicit assembly, and increasingly light gray stands for sequential summations and for the two versions of the fast algorithm, respectively. Different symbols stand for different choices of dimensions. We consider $d=2,\dots,5$. As for the choice of the threshold K, we start from $K=10$ and proceed in steps of 5 until $K=75$. The upper half of Figure 3.2 contains absolute computation times in seconds, while the lower part gives the corresponding ratios of computation times. Table 3.7 contains a selection of the underlying data (as well as data corresponding to the subsequent experiments for the additive reduction and for the full index sube). The hyperbolic reduction is the setting where the relative performance of the fast algorithm as compared to sequential summations is the strongest. When compared to an explicit assembly, its second version easily allows for a reduction of two orders of magnitude.

Next comes the additive reduction. In this case, it only takes a moderate choice of d for an explicit assembly to become infeasible. In accordance with the indication from Table 3.2, the choices $d=3$ and $K=50$ actually make us run out of memory. To circumvent at least this problem of lack of memory, we should better do the matrix-vector product entrywise by computing only a single row, then compute its inner product with the vector, store only the resulting scalar quantity, and proceed with the next row. Still, the problem of unbearably long computation times remains. In case $d=4$, what we have done instead is measure the computation time for only the first row of $\mathbf{W}^{\text{quad}}$ and multiply it by the number of rows. This way, we miss the time contribution from the subsequent matrix-vector multiplication. However, as we have experienced in cases where there is still enough memory, the bottleneck is actually the assembly, and the figures we present are only about 10–15% below the expected computation times when the matrix-vector multiplication is included. Thus, our little cheating yields the same qualitative behavior. We restrict our considerations to $d=2,3,4$. The experimental results are given in Figure 3.3 and Table

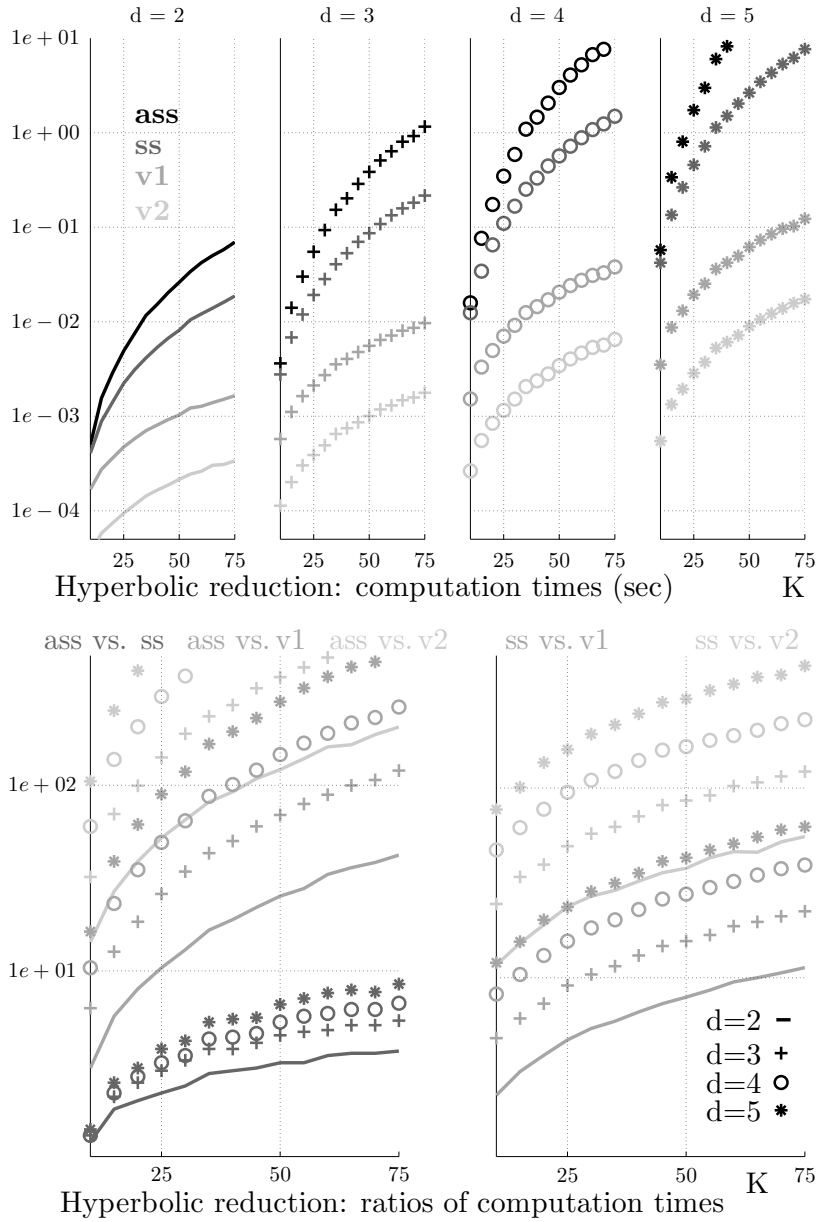


Figure 3.2: Hyperbolic reduction $\mathcal{K}_{\text{hyp}}$: Computation times for explicit assembly (ass, black color), sequential summations (ss, dark gray), and the fast algorithm in both versions (v1, middle gray, and v2, light gray). Solid lines indicate $d=2$. Plus symbols, circles, and asterisks stand for $d=3,4,5$, respectively. A selection of underlying data is provided in Table 3.7. Semi-logarithmic plots.

3.7. As the figures reveal, increasing the dimension by only 1 makes the time scale change from seconds to minutes to hours. It can be seen that the fast algorithm still outperforms sequential summations, but by a less significant margin, as we expect from the theoretical analysis given at the end of the previous chapter. On the other hand, the gap between assembly and sequential summations widens. Even in case $d=4$ and with a large threshold K, sequential summations take only a few minutes, while assembling the matrix would take several days.

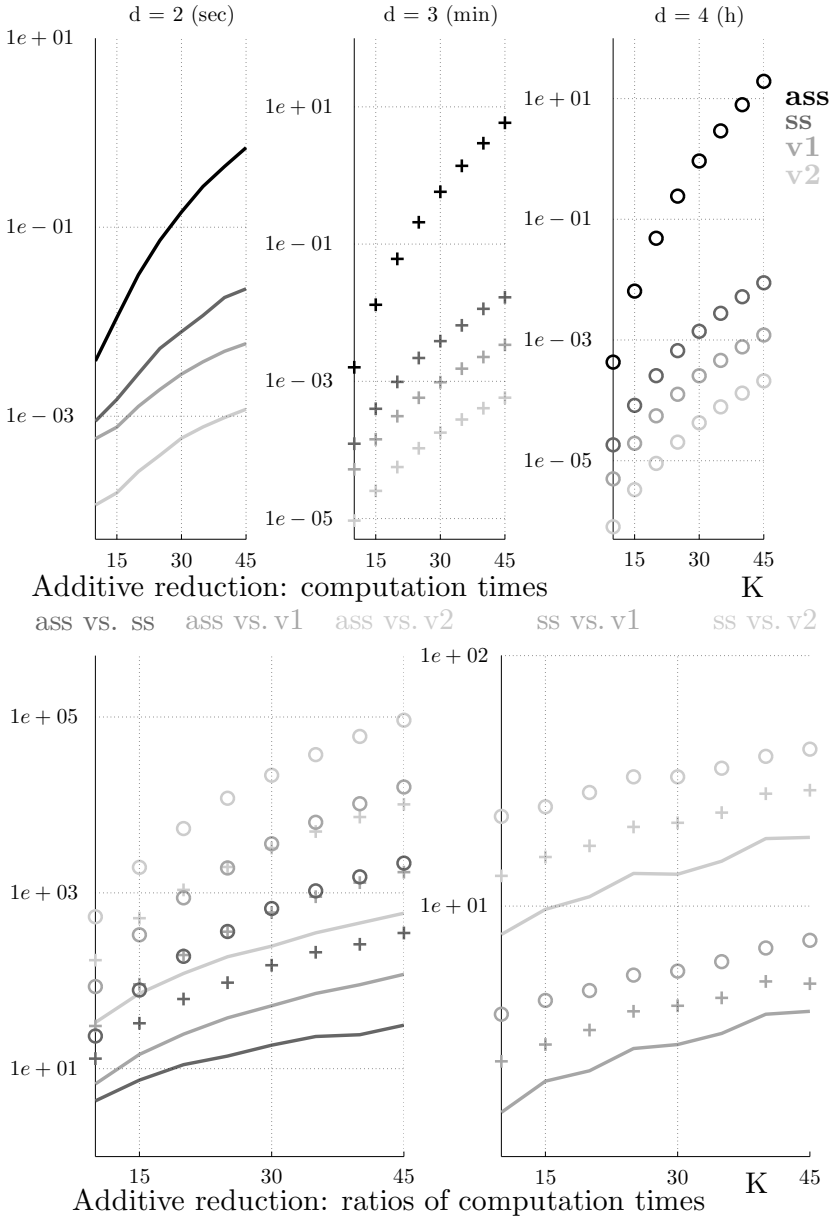


Figure 3.3: Additive reduction $\mathcal{K}_{\text{add}}$: Computation times and ratios as in the above Figure 3.2. Colors and symbols are the same as above. Again, a selection of underlying data is provided in Table 3.7. The time scale (sec, min, h) now depends on the choice of d. For $d = 2, 3$, we have assembled the matrix as above. Due to lack of memory, in $d = 4$, we have only computed the first row of $\mathbf{W}^{\text{quad}}$ and multiplied the computation time by the number of rows to obtain an approximate overall computation time.Semi-logarithmic plots.

We finally turn to the full index cube. In this case, there is no hope for the endeavor of explicitly assembling the full matrix. By taking an explicit assembly into consideration, we merely intend to give a flavor of how heavy a computationl burden the full index cube actually is. The performance is shown in the lower part of Table 3.7. We restrict ourselves to $K = 30$. In accordance with the above analysis, sequential summations perform not much

worse than the first version of the fast algorithm.

To conclude, we briefly mention the two main findings from the above experiments: First, from a computational point of view, there is still a considerable leeway beyond index set reduction. This motivates both the sequential summations approach and the fast algorithm. To put it differently, if index set reduction is not a means of choice due to missing regularity of the approximated function, direct operation procedures such as sequential summations or the fast algorithm may allow us to perform computations within a reasonable amount of time and with reasonable memory even though the underlying basis is relatively large. Second, the closer the chosen index set comes to the full index cube, the closer the computational performance of sequential summations and the fast algorithm. The more the basis is pruned, the more is sequential summations outperformed by the fast algorithm.

		hyperbolic reduction								
		time (sec)				**ratio**				
	K	ass	ss	v1	v2	$\frac{\text{ass}}{\text{ss}}$	$\frac{\text{ass}}{\text{v1}}$	$\frac{\text{ass}}{\text{v2}}$	$\frac{\text{ss}}{\text{v1}}$	$\frac{\text{ss}}{\text{v2}}$
$d=2$	25	4.90e-03	2.23e-03	4.72e-04	9.40e-05	2.2	10.4	52.1	4.7	23.7
	50	2.61e-02	8.13e-03	1.03e-03	2.15e-04	3.2	25.3	121.6	7.9	37.8
	75	6.95e-02	1.87e-02	1.65e-03	3.37e-04	3.7	42.1	206.2	11.3	55.5
$d=3$	25	5.50e-02	1.92e-02	2.12e-03	3.89e-04	2.9	26.0	141.4	9.1	49.4
	50	3.85e-01	8.65e-02	5.56e-03	1.01e-03	4.5	69.2	382.8	15.6	86.0
	75	1.16e+00	2.17e-01	9.67e-03	1.77e-03	5.4	120.1	655.2	22.4	122.2
$d=4$	25	3.48e-01	1.10e-01	7.05e-03	1.16e-03	3.2	49.3	300.9	15.6	95.1
	50	3.01e+00	5.68e-01	2.06e-02	3.43e-03	5.3	146.4	877.2	27.6	165.6
	75	1.00e+01	1.49e+00	3.80e-02	6.50e-03	6.7	264.1	1544.1	39.3	229.7
$d=5$	25	1.74e+00	4.57e-01	1.94e-02	2.86e-03	3.8	89.6	606.6	23.6	159.8
	50	1.74e+01	2.65e+00	6.18e-02	8.96e-03	6.6	282.1	1943.9	42.9	295.6
	75	6.55e+01	7.66e+00	1.23e-01	1.74e-02	8.5	533.6	3770.8	62.5	441.4

		additive reduction								
		time (sec)				**ratio**				
	K	ass	ss	v1	v2	$\frac{\text{ass}}{\text{ss}}$	$\frac{\text{ass}}{\text{v1}}$	$\frac{\text{ass}}{\text{v2}}$	$\frac{\text{ss}}{\text{v1}}$	$\frac{\text{ss}}{\text{v2}}$
$d=2$	15	1.11e-02	1.51e-03	7.67e-04	1.55e-04	7.4	14.5	71.7	2.0	9.7
	30	1.45e-01	7.84e-03	2.79e-03	5.85e-04	18.5	51.9	247.3	2.8	13.4
	45	6.95e-01	2.23e-02	5.88e-03	1.19e-03	31.2	118.3	586.3	3.8	18.8
$d=3$	15	7.83e-01	2.39e-02	8.59e-03	1.52e-03	32.8	91.2	514.7	2.8	15.7
	30	3.47e+01	2.31e-01	5.81e-02	1.07e-02	150.2	596.9	3229.2	4.0	21.5
	45	3.51e+02	1.01e+00	2.04e-01	3.46e-02	349.2	1719.3	10142.4	4.9	29.0
$d=4$	15	2.33e+01	2.97e-01	7.00e-02	1.19e-02	78.3	332.6	1950.8	4.2	24.9
	30	3.33e+03 ≈ 55.5 min	5.02e+00	9.20e-01	1.53e-01	662.8	3614.9	21753.9	5.5	32.8
	45	6.95e+04 ≈ 19.3 h	3.19e+01	4.35e+00	7.55e-01	2177.8	15983.7	92037.3	7.3	42.3

		full index cube								
		time (sec)				**ratio**				
d	K	ass	ss	v1	v2	$\frac{\text{ass}}{\text{ss}}$	$\frac{\text{ass}}{\text{v1}}$	$\frac{\text{ass}}{\text{v2}}$	$\frac{\text{ss}}{\text{v1}}$	$\frac{\text{ss}}{\text{v2}}$
2	30	5.44e-01	1.04e-02	4.48e-03	9.18e-04	52.4	121.4	592.6	2.3	11.3
3	30	8.54e+02 ≈ 14.2 min	7.24e-01	2.71e-01	5.01e-02	1179.9	3148.3	17029.1	2.7	14.4
4	30	1.74e+06 ≈ 20.4 days	5.79e+01	1.55e+01	2.81e+00	3.00e+04	1.12e+05	6.19e+05	3.7	20.6

Table 3.7: Table of data corresponding to Figures 3.2, 3.3, and for the full index cube.

4 Error analysis

The fast algorithm is a means to compute approximately the action of the matrix

$$\mathbf{W}(t)\mathbf{v}, \qquad \mathbf{W}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W(\cdot,t)\varphi_{\mathbf{k}}), \quad \mathbf{j},\mathbf{k} \in \mathcal{K}, \qquad t \geq 0,$$

on a vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$, for a sufficiently smooth potential $W(\cdot,t)$. As a first step, we need to approximate $W(\cdot,t)$ by a multivariate polynomial $W^{\mathrm{pol}}(\cdot,t)$. The error due to polynomial approximation of W is studied separately in Section 4.2.

The fast algorithm itself gives rise to an error due to quadrature and, in case $\mathcal{K} \subsetneq \mathcal{K}_{\mathrm{full}}$ is smaller than the full index cube, to an additional error due to index set reduction. Section 4.1 contains a detailed outline of the different error contributions and what techniques they are dealt with in the subsequent analysis, introducing all the necessary notation. For a full index cube, the error due to quadrature is discussed in Section 4.3. The case of $\mathcal{K}$ being reduced is considerably more difficult. An appropriate error decomposition is given in Section 4.4. We cast the regularity assumption on the exact solution as a componentwise decay assumption on the corresponding coefficient vector, see Section 4.5, which is then used in the analysis of the errors due to quadrature and index set reduction as given in Section 4.6 and 4.7, respectively.

4.1 Outline and main results

We repeat some notations together with general assumptions that are tacitly used throughout the subsequent analysis. Let

$$D = -\frac{1}{2}\left(\Delta - \sum_{\alpha=1}^{d} x_\alpha^2\right), \qquad W(x,t) = V(x,t) - \frac{1}{2}\sum_{\alpha=1}^{d} x_\alpha^2$$

denote the harmonic oscillator and the potential, respectively, with a smooth and possibly time-dependent potential $V(\cdot,t)$ as it occurs in (1.1); see Section 1.1.

Additionally, as said earlier, we assume that $W(\cdot,t)$ is sufficiently smooth such that it can be approximated by a polynomial with significantly fewer terms of significantly lower degree than the Galerkin approximate to the solution $\psi(\cdot,t)$ of the Schrödinger equation (1.1) wherever the latter does not essentially vanish, for all $t \geq 0$. More precisely, as explained in Section 1.4, we consider Chebyshev interpolation on a cube where ψ is assumed to be essentially supported for all times, viz.,

$$W(x,t) \approx W^{\mathrm{pol}}(x,t) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) T_{\mathbf{r}}(x/S), \qquad x \in \Omega = [-S,S]^d.$$

The multi-index set $\mathcal{R}(d,R) \subseteq \mathbb{N}^d$ is assumed to satisfy

$$|\mathcal{K}(d,K)| \gg |\mathcal{R}(d,R)|, \qquad K \gg R,$$

where $\mathcal{K}(d,K)$ is the index set for the spectral approximate to $\mathcal{K}$, and K is chosen such that $S \geq \sqrt{2(K+1)}+1$; see Section 1.4.

4.1.1 Solutions and their approximations

To facilitate the error analysis, we introduce the Hermite expansions

$$\psi(x,t) = \sum_{\mathbf{k}\in\mathbb{N}^d} u_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x), \qquad \psi^{\mathrm{pol}}(x,t) = \sum_{\mathbf{k}\in\mathbb{N}^d} u^{\mathrm{pol}}_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x),$$

as well as the truncated Hermite expansions

$$\psi^{\mathrm{quad}}_{\mathcal{K}}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}} c^{\mathrm{quad}}_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x), \qquad \psi^{\mathrm{fast}}_{\mathcal{K}}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}} c^{\mathrm{fast}}_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x),$$

where $\mathcal{K} = \mathcal{K}(d,K) \subseteq \mathcal{K}_{\mathrm{full}}(d,K)$ is a subset of the full multi-index cube that satisfies the closure condition (1.15).

The coefficients $u_{\mathbf{k}}$ are chosen such that ψ solves the original weakly formulated problem,

$$i(\varphi, \psi_t(\cdot,t)) = (\varphi, D\psi(\cdot,t)) + (\varphi, W(\cdot,t)\psi(\cdot,t)) \tag{4.1}$$

for all $\varphi \in H^1(\Omega)$ and for all $t \geq 0$, with initial data $\psi(x,0) = \psi^0(x)$.

Analogously, ψ^{pol} is the solution to the same problem, but with W replaced by W^{pol},

$$i(\varphi, \psi^{\mathrm{pol}}_t(\cdot,t)) = (\varphi, D\psi^{\mathrm{pol}}(\cdot,t)) + (\varphi, W^{\mathrm{pol}}(\cdot,t)\psi^{\mathrm{pol}}(\cdot,t)) \tag{4.2}$$

for all $\varphi \in H^1(\Omega)$ and for all $t \geq 0$, with the same initial data $\psi^{\mathrm{pol}}(x,0) = \psi^0(x)$. We recall the definition of the matrix representation of $W^{\mathrm{pol}}(\cdot,t)$,

$$\mathbf{W}^{\mathrm{pol}}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}(\cdot,t)\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K}, \qquad t \geq 0. \tag{4.3}$$

The function $\psi^{\mathrm{quad}}_{\mathcal{K}}$ is the Galerkin approximation to ψ^{pol}, where $(K{+}1)$-nodes full-product Gauß–Hermite quadrature has been taken into account for the potential part,

$$i(\varphi_{\mathbf{j}}, (\psi^{\mathrm{quad}}_{\mathcal{K}})_t(\cdot,t)) = (\varphi_{\mathbf{j}}, D\psi^{\mathrm{quad}}_{\mathcal{K}}(\cdot,t)) + (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}(\cdot,t)\psi^{\mathrm{quad}}_{\mathcal{K}}(\cdot,t))_{\mathrm{quad}} \tag{4.4}$$

for all $\mathbf{j} \in \mathcal{K}$ and $t \geq 0$ or, equivalently,

$$i\dot{\mathbf{c}}^{\mathrm{quad}}(t) = \mathbf{D}\mathbf{c}^{\mathrm{quad}}(t) + \mathbf{W}^{\mathrm{quad}}(t)\mathbf{c}^{\mathrm{quad}}(t), \tag{4.5}$$

where we set $\mathbf{c}^{\mathrm{quad}}(t) = (c^{\mathrm{quad}}_{\mathbf{k}}(t))_{\mathbf{k}\in\mathcal{K}}$, and

$$\mathbf{D} = \mathrm{diag}_{\mathbf{k}\in\mathcal{K}}\left(\sum_{\alpha=1}^{d}\left(k_\alpha + \frac{1}{2}\right)\right), \qquad \mathbf{W}^{\mathrm{quad}}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}(\cdot,t)\varphi_{\mathbf{k}})_{\mathrm{quad}}. \tag{4.6}$$

As an initialization, we choose the $L^2(\mathbb{R}^d)$-orthogonal projection of ψ^0, i.e., $\mathcal{P}_{\mathcal{K}}\psi^0$; cf. the definition of $\mathcal{P}_{\mathcal{K}}$ given in (1.5).

#	symbol	coefficients	equ.	init.	description
(1)	ψ	$u_{\mathbf{k}},\ \mathbf{k} \in \mathbb{N}^d$	(4.1)	ψ^0	original weak formulation
(2)	ψ^{pol}	$u^{\mathrm{pol}}_{\mathbf{k}},\ \mathbf{k} \in \mathbb{N}^d$	(4.2)	ψ^0	$W \approx W^{\mathrm{pol}}$
(3)	$\psi^{\mathrm{quad}}_{\mathcal{K}}$	$c^{\mathrm{quad}}_{\mathbf{k}},\ \mathbf{k} \in \mathcal{K}$	(4.4)	$\mathcal{P}_{\mathcal{K}}\psi^0$	$W \approx W^{\mathrm{pol}}$, Galerkin approximation, quadrature
(4)	$\psi^{\mathrm{fast}}_{\mathcal{K}}$	$c^{\mathrm{fast}}_{\mathbf{k}},\ \mathbf{k} \in \mathcal{K}$	(4.7)	$\mathcal{P}_{\mathcal{K}}\psi^0$	$W \approx W^{\mathrm{pol}}$, Galerkin approximation, insertion of $\mathbf{X}^{(\alpha)}$ into W^{pol}

Table 4.1: Overview: Steps of error analysis. (1) vs. (2) is the interpolation error; see Section 4.2. (2) vs. (3) is the error due to the fast algorithm over a full index cube; see Section 4.3. (2) vs. (4) is the error due to the fast algorithm over a reduced index set; see Sections 4.4–4.7.

Finally, $\psi^{\mathrm{fast}}_{\mathcal{K}} = \sum_{\mathbf{k}\in\mathcal{K}} c^{\mathrm{fast}}_{\mathbf{k}} \varphi_{\mathbf{k}}$ is the physical space representation of the solution $\mathbf{c}^{\mathrm{fast}}(t) = (c^{\mathrm{fast}}_{\mathbf{k}}(t))_{\mathbf{k}\in\mathcal{K}}$ to

$$i\dot{\mathbf{c}}^{\mathrm{fast}}(t) = \mathbf{D}\mathbf{c}^{\mathrm{fast}}(t) + W^{\mathrm{pol}}(\mathbf{X}, t)\mathbf{c}^{\mathrm{fast}}(t), \tag{4.7}$$

where, in the last term on the right-hand side, $W^{\mathrm{pol}}(\mathbf{X}, t)$ denotes again formal insertion of the coordinate matrices

$$\mathbf{X}^{(\alpha)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, x_\alpha \varphi_{\mathbf{k}}), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K},$$

in place of x_α into the polynomial W^{pol};, see Section 2.3. The initial value is again $\mathcal{P}_{\mathcal{K}}\psi^0$.

Equation (4.7) is the semidiscrete problem as we actually propagate it in time. Thus, our overall aim is to estimate the spatial error

$$\|\psi(\cdot, t) - \psi^{\mathrm{fast}}_{\mathcal{K}}(\cdot, t)\|, \qquad t \geq 0.$$

We do this using the above error decomposition. An overview over all steps of the subsequent error analysis is provided in Table 4.1, where we recap the above notation including a concise characterization of what the different quantities represent.

4.1.2 Organization of the analysis and main results

In Section 4.2, we discuss the error due to polynomial interpolation of the potential, i.e., the difference between ψ and ψ^{pol}. This involves standard techniques. We shall nevertheless go through the details of the analysis because, first, the proof serves as a sketch for the derivation of global error estimates in subsequent sections and, second, the presentation can afterwards focus more easily on essential novelties.

In Section 4.3, we consider the choice of a full index set. As we have seen in Lemma 2, in case of $\mathcal{K} = \mathcal{K}_{\mathrm{full}}$, approximation of the potential matrix $\mathbf{W}(t)$ by the kind of quadrature under consideration is actually equivalent to formal insertion of $\mathbf{X}^{(\alpha)}$ into the potential.

Thus, $\psi_{\mathcal{K}}^{\mathrm{quad}}$ and $\psi_{\mathcal{K}}^{\mathrm{fast}}$ coincide. We present an analysis that comprises the errors both due to a truncation of the Galerkin basis and due to quadrature, i.e., we compare ψ^{pol} and $\psi_{\mathcal{K}}^{\mathrm{quad}}$. This is done using adaptations of ideas developed in [13]; see the introductory remarks at the beginning of Section 4.3. Leaving aside the error due to polynomial interpolation of the potential as given in Lemma 3, we can summarize the error due to spatial discretization as given by our method as follows. For the full index cube, we find:

Theorem 1. (global error of spatial discretization, full index cube)
Let $\psi^{\mathrm{pol}} = \sum_{\boldsymbol{k}\in\mathbb{N}^d} u_{\boldsymbol{k}}^{\mathrm{pol}}\varphi_{\boldsymbol{k}}$ be the solution to (4.2), and let $\psi_{\mathcal{K}}^{\mathrm{quad}}$ be the solution to (4.4) with initialization $\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot,0) = \psi_{\mathcal{K}}^{\mathrm{quad}}(\cdot,0)$, where $\mathcal{K}(d,K)$ is the full index cube.
For all even integers s such that

$$|\psi^{\mathrm{pol}}(\cdot,t)|_s < \infty, \qquad t \geq 0,$$

the global error due to spatial discretization is then given by

$$\|\psi^{\mathrm{pol}}(\cdot,t) - \psi_{\mathcal{K}}^{\mathrm{quad}}(\cdot,t)\| \leq CK^{-s/2}\big(|\psi^{\mathrm{pol}}(\cdot,t)|_s + \int_0^t |\psi^{\mathrm{pol}}(\cdot,\tau)|_s \,\mathrm{d}\tau\big), \quad t \geq 0.$$

where $C = C(W^{\mathrm{pol}}, d, s)$.

This is simply a repetition of the below Theorem 3. See (1.19) for a definition of the $|\cdot|_s$-seminorm.

Reducing the index set makes the analysis considerably more intricate. In case of $\mathcal{K} \subsetneq \mathcal{K}_{\mathrm{full}}$, the solutions $\psi_{\mathcal{K}}^{\mathrm{quad}}$ and $\psi_{\mathcal{K}}^{\mathrm{fast}}$ no longer coincide. In Section 4.4, comparing ψ^{pol} and $\psi_{\mathcal{K}}^{\mathrm{fast}}$, we give an appropriate error decomposition that facilitates the analysis. In a specific way, the error can be split into a contribution due to quadrature and a contribution due to index set reduction itself. Based on a decay assumption on the coefficient vector of the exact solution ψ^{pol} introduced and explained in Section 4.5, the corresponding local errors are then analyzed in Sections 4.6 and 4.7, respectively. The analysis as it concerns the reduced case is due to [12]. Sections 4.4–4.7 contain an extended and slightly modified version of the material presented in [12]. We restrict our considerations to a hyperbolically reduced index set. After a thorough analysis, we eventually find:

Theorem 2. (global error of spatial discretization, hyperbolic reduction)
Let $\psi^{\mathrm{pol}} = \sum_{\boldsymbol{k}\in\mathbb{N}^d} u_{\boldsymbol{k}}^{\mathrm{pol}}\varphi_{\boldsymbol{k}}$ be the solution to (4.2), and let $\psi_{\mathcal{K}}^{\mathrm{fast}}$ be the solution to (4.7) with initialization $\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot,0) = \psi_{\mathcal{K}}^{\mathrm{fast}}(\cdot,0)$, where $\mathcal{K}(d,K)$ is a hyperbolically reduced index set.
For all integers s such that

$$|\psi^{\mathrm{pol}}(\cdot,t)|_s < \infty, \qquad t \geq 0,$$

the global error due to spatial discretization is then given by

$$\begin{aligned}\|\psi^{\mathrm{pol}}(\cdot,t) - \psi_{\mathcal{K}}^{\mathrm{fast}}(\cdot,t)\| \leq{}& C_1K^{-s/2}\big(|\psi^{\mathrm{pol}}(\cdot,t)|_s + \int_0^t |\psi^{\mathrm{pol}}(\cdot,\tau)|_s \,\mathrm{d}\tau\big) \\ &+ C_2(t)K^{d/2}\varepsilon\int_0^t |\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot,\tau)|_{s;\infty}\,\mathrm{d}\tau,\end{aligned} \qquad t \geq 0.$$

where

$$C_1 = C_1(W^{\mathrm{pol}}, d, s), \qquad C_2(t) = C_2(d, s, \mathcal{R}, W^{\mathrm{pol}}, S, t),$$

and the factor ε is defined as

$$\varepsilon = \min\{\varepsilon^{\mathrm{quad}}, \varepsilon^{\mathrm{red}}\}, \tag{4.8}$$

where

$$\varepsilon^{\mathrm{quad}} = \min_{\substack{\boldsymbol{k}\in\mathcal{K},\\ \exists\alpha: k_\alpha \geq K-R+2}} \prod_{\beta=1}^{d} \max\{1, k_\beta\}^{-s/2},$$

$$\varepsilon^{\mathrm{red}} = \min_{\substack{\boldsymbol{k}\in\mathcal{K}, \boldsymbol{r}\in\mathcal{R},\\ \boldsymbol{k}+\boldsymbol{r}\in\mathcal{K}_{\mathrm{full}}\setminus\mathcal{K}}} \prod_{\beta=1}^{d} \max\{1, k_\beta - r_\beta\}^{-s/2}$$

This follows directly from putting together the estimates given in the subsequent Theorems 4–6 according to the decomposition given in Section 4.4. The norm $|\cdot|_{s;\infty}$ is defined in (1.20).

Some comments are in order: In the estimate given in the previous Theorem 1, the error due to quadrature is subsumed in the standard projection error. In case of a reduced index set, this is no longer possible. Quite the contrary, as we can no longer invoke arguments that are valid only for a full index cube, not even the error contribution due to quadrature can be easily accounted for any more—not to mention the additional error due to index set reduction itself. Consequently, the error bound given in Theorem 2 exhibits another term, which is due to both quadrature and index set reduction. We shall comment on where the full cube strategy fails in case of a reduced index set in the course of the below analysis. The below Theorem 4 gives an estimate for the global error due to spatial discretization that summarizes the proceeding up to the point where the paths separate and new techniques need to be applied.

To solve the difficulties that arise from the failure of our previous strategy, first, we translate the regularity condition $|\psi^{\mathrm{pol}}(\cdot, t)|_s < \infty$ on the projected exact solution into a decay property for the corresponding coefficient vector $\mathbf{u}^{\mathrm{pol}}$. In Section 4.5, we discuss how exactly this decay property reflects the regularity of ψ^{pol}, and why this makes the norm $|\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot, t)|_{s;\infty}$ of the projected exact solution pop up in the estimate. Second, we employ new techniques based on binary trees to study both the error due to quadrature and due to index set reduction. The idea is to convert the Hermite and Chebyshev recurrence relations into binary trees in a suitable way, and to apply combinatorial arguments to these trees. This eventually leads to the somewhat complicated statement of the error decay as given above in eq. (4.8). The requirement that s be even is only for ease of presentation; see the remarks at the end of Section 4.5.

Each of the minima that occur in (4.8) has its specific meaning that we shall elucidate in Sections 4.6 and 4.7. The first argument in the outermost minimum stems from the local error contribution due to quadrature; see Theorem 5. The second argument comes from reducing the index set; see Theorem 6. As will become clear from the proofs to these theorems, the above definition of ε is a conservative bound. Using $K \gg R$, it is easily seen

$$\varepsilon \sim K^{-s/2},$$

thus, the overall error decays spectrally if $s > d$.

The dependency of the constant C_2 on S (and on t) comes from the Chebyshev coefficients of W^{pol} and does not affect the bound as a function of K.

In the following, for the sake of easier readability, we shall often drop explicit time-dependency whenever it is justified by the context. In order to avoid any confusion, be aware that we always consider a full product of $(K+1)$-nodes Gaussian quadrature formulas, which we refer to by a superscript or a subscript "quad", despite $\mathcal{K}(d, K)$ possibly being reduced.

4.2 Interpolation error

We estimate the error due to replacing the potential W by some polynomial approximation W^{pol} on $\Omega = [-S, S]^d$.

Lemma 3. (interpolation error)
If $\psi \in H^2(\Omega)$ is the solution to (4.1) and $\psi^{\mathrm{pol}} \in H^2(\Omega)$ is the solution to (4.2) with initialization $\psi(\cdot, 0) = \psi^{\mathrm{pol}}(\cdot, 0)$, the error is given by

$$\|\psi(\cdot,t) - \psi^{\mathrm{pol}}(\cdot,t)\| \leq \int_0^t \|W(\cdot,\tau) - W^{\mathrm{pol}}(\cdot,\tau)\| \|\psi(\cdot,\tau)\| \,\mathrm{d}\tau, \qquad t \geq 0.$$

Proof. The proof uses standard techniques; cf. [61], Theorem II.1.5. For ease of presentation, we omit explicit time-dependency. We define $e = \psi - \psi^{\mathrm{pol}}$ and subtract (4.2) from (4.1). Using

$$W\psi - W^{\mathrm{pol}}\psi^{\mathrm{pol}} = W^{\mathrm{pol}}(\psi - \psi^{\mathrm{pol}}) + (W - W^{\mathrm{pol}})\psi = W^{\mathrm{pol}}e + (W - W^{\mathrm{pol}})\psi,$$

this yields the error equation

$$i(\varphi, e_t) = (\varphi, (D + W^{\mathrm{pol}})e) + (\varphi, (W - W^{\mathrm{pol}})\psi) \tag{4.9}$$

for all $\varphi \in H^1(\Omega)$. Because W^{pol} is real-valued, we find

$$(\varphi, W^{\mathrm{pol}}\varphi) = \overline{(\varphi, W^{\mathrm{pol}}\varphi)}$$

for all $\varphi \in L^2(\Omega)$, thus, $\Re(-i(\varphi, W^{\mathrm{pol}}\varphi)) = 0$. Additionally, the operator $-\frac{1}{2}\Delta$ is self-adjoint with the domain $H^2(\mathbb{R}^d)$. Using (4.9) with $\varphi = e$ and taking the real part, this allows to compute

$$\begin{aligned}
\|e\|\frac{\mathrm{d}}{\mathrm{d}t}\|e\| &= \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|e\|^2 = \Re(e, e_t) \\
&= \underbrace{\Re(-i(e, (D + W^{\mathrm{pol}})e))}_{=0} + \Re(-i(e, (W - W^{\mathrm{pol}})\psi) \\
&\leq \|e\|\|(W - W^{\mathrm{pol}})\psi\|
\end{aligned}$$

Dividing by $\|e\|$ and integrating from 0 to t gives the desired result. □

The interpolation error $W - W^{\mathrm{pol}}$ can now be bounded in terms of the order R of the interpolant and of the derivatives of W using standard theory. We refer to the comments in Section 1.4 for the details.

4.3 Spatial discretization

In the present section, we restrict our considerations to the case $\mathcal{K} = \mathcal{K}_{\text{full}}$. The case of a reduced index set is more complicated and shall thus be delayed to Section 4.4 and onwards. On the full index cube, applying the fast algorithm is equivalent to $(K+1)$-nodes full-product Gauss–Hermite quadrature for the potential part, as seen in Section 2.6. Hence, we compare ψ^{pol} and $\psi_{\mathcal{K}}^{\text{quad}}$ in order to analyze the overall error due to spatial discretization. Again, let $\mathcal{P}_{\mathcal{K}}$ denote the $L^2(\mathbb{R}^d)$-orthogonal projection onto the approximation space $\operatorname{span}\{\varphi_{\mathbf{k}}\,;\,\mathbf{k} \in \mathcal{K}\}$ and $\mathcal{P}_{\mathcal{K}}^{\perp}$ its orthogonal complement; see (1.5). Following [86], the basic idea is to invoke a standard projection error estimate for the exact solution ψ^{pol}, and to compare the projected solutions $\mathcal{P}_{\mathcal{K}}\psi^{\text{pol}}$ and ψ^{quad} with and without quadrature separately; see the ansatz in the proof of Theorem 3. The latter error can be analyzed using an appropriate projection matrix that is related to the exactness properties of Gaussian quadrature; see the below Lemma 4. The technique has been developed in [13]. We present an adaptation to a Hermite basis. The errors are given in terms of Korobov seminorms of the solution ψ^{pol} to (4.2).

Theorem 3. (global error of spatial discretization, full index cube)
Let $\psi^{\text{pol}} = \sum_{\boldsymbol{k}\in\mathbb{N}^d} u_{\boldsymbol{k}}^{\text{pol}}\varphi_{\boldsymbol{k}}$ be the solution to (4.2), and let $\psi_{\mathcal{K}}^{\text{quad}}$ be the solution to (4.4) with initialization $\mathcal{P}_{\mathcal{K}}\psi^{\text{pol}}(\cdot,0) = \psi_{\mathcal{K}}^{\text{quad}}(\cdot,0)$, where $\mathcal{K}(d,K)$ is the full index cube.
For all integers s such that

$$|\psi^{\text{pol}}(\cdot,t)|_s < \infty, \qquad t \geq 0,$$

the global error due to spatial discretization is then given by

$$\|\psi^{\text{pol}}(\cdot,t) - \psi_{\mathcal{K}}^{\text{quad}}(\cdot,t)\| \leq CK^{-s/2}\big(|\psi^{\text{pol}}(\cdot,t)|_s + \int_0^t |\psi^{\text{pol}}(\cdot,\tau)|_s \,\mathrm{d}\tau\big), \quad t \geq 0.$$

where $C = C(W^{\text{pol}}, d, s)$.

Proof. Again, we omit time-dependency. A common way of decomposing the error is

$$\psi^{\text{pol}} - \psi_{\mathcal{K}}^{\text{quad}} = \underbrace{\big(\psi^{\text{pol}} - \mathcal{P}_{\mathcal{K}}\psi^{\text{pol}}\big)}_{=\rho} + \underbrace{\big(\mathcal{P}_{\mathcal{K}}\psi^{\text{pol}} - \psi^{\text{quad}}\big)}_{=\theta}; \tag{4.10}$$

see [86]. The error component ρ is bounded by the standard projection estimate

$$\|\psi^{\text{pol}} - \mathcal{P}_{\mathcal{K}}\psi^{\text{pol}}\| \leq C(d,s)K^{-s/2}|\psi^{\text{pol}}|_s; \tag{4.11}$$

see (1.18). The error component θ is controlled as follows. By orthogonality,

$$(\varphi_{\mathbf{j}}, (\mathcal{P}_{\mathcal{K}}^{\perp}\psi^{\text{pol}})_t) = \sum_{\mathbf{k}\notin\mathcal{K}} (\varphi_{\mathbf{j}}, \varphi_{\mathbf{k}}) \dot{u}_{\mathbf{k}}^{\text{pol}} = 0 \qquad \forall\, \mathbf{j} \in \mathcal{K}$$

and, using the eigenfunction relation,

$$\begin{aligned}(\varphi_{\mathbf{j}}, D\mathcal{P}_{\mathcal{K}}^{\perp}\psi^{\text{pol}}) &= \sum_{\mathbf{k}\notin\mathcal{K}} (\varphi_{\mathbf{j}}, D\varphi_{\mathbf{k}}) u_{\mathbf{k}}^{\text{pol}} \\ &= \sum_{\mathbf{k}\notin\mathcal{K}} (\varphi_{\mathbf{j}}, \varphi_{\mathbf{k}}) \sum_{\alpha=1}^{d} \big(k_\alpha + \frac{1}{2}\big) u_{\mathbf{k}}^{\text{pol}} = 0 \qquad \forall\, \mathbf{j} \in \mathcal{K}.\end{aligned}$$

Therefore, using (4.2) with $\varphi = \varphi_{\mathbf{j}}$, $\mathbf{j} \in \mathcal{K}$, we get

$$\begin{aligned} i(\varphi_{\mathbf{j}}, (\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}})_t) &= (\varphi_{\mathbf{j}}, D\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}) \\ &\quad + (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}) + (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}\mathcal{P}_{\mathcal{K}}^{\perp}\psi^{\mathrm{pol}}) \end{aligned} \qquad \forall\, \mathbf{j} \in \mathcal{K}. \tag{4.12}$$

Defining a defect

$$d_{\mathbf{j}} = (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}\mathcal{P}_{\mathcal{K}}^{\perp}\psi^{\mathrm{pol}})$$

and setting $\mathbf{d} = (d_{\mathbf{j}})_{\mathbf{j}\in\mathcal{K}}$, $\mathbf{u}^{\mathrm{pol}} = (u_{\mathbf{k}}^{\mathrm{pol}})_{\mathbf{k}\in\mathcal{K}}$, we can rewrite (4.12) equivalently as

$$i\dot{\mathbf{u}}^{\mathrm{pol}} = \mathbf{D}\mathbf{u}^{\mathrm{pol}} + \mathbf{W}^{\mathrm{pol}}\mathbf{u}^{\mathrm{pol}} + \mathbf{d}. \tag{4.13}$$

By Parseval's identity, the error $\mathbf{e} = \mathbf{u}^{\mathrm{pol}} - \mathbf{c}^{\mathrm{quad}}$ equals θ. What follows is a discrete analog of the proceeding given in the proof of Lemma 3. We subtract (4.5) from (4.13) to obtain the error equation

$$i\dot{\mathbf{e}} = \mathbf{D}\mathbf{e} + \mathbf{W}^{\mathrm{quad}}\mathbf{e} + (\mathbf{W}^{\mathrm{pol}} - \mathbf{W}^{\mathrm{quad}})\mathbf{u}^{\mathrm{pol}} + \mathbf{d}. \tag{4.14}$$

Multiplying (4.14) with $\mathbf{e}^*$, taking the real part, and using

$$\mathbf{e}^*\mathbf{D}\mathbf{e} = \sum_{\mathbf{j}} \underbrace{\mathbf{D}_{\mathbf{jj}}}_{\geq 0} |e_{\mathbf{j}}|^2 \in \mathbb{R}, \qquad \mathbf{e}^*\mathbf{W}^{\mathrm{quad}}\mathbf{e} = \overline{\mathbf{e}^*\mathbf{W}^{\mathrm{quad}}\mathbf{e}} \in \mathbb{R},$$

we find

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\mathbf{e}\| \leq \|(\mathbf{W}^{\mathrm{pol}} - \mathbf{W}^{\mathrm{quad}})\mathbf{u}^{\mathrm{pol}}\| + \|\mathbf{d}\|.$$

The term $(\mathbf{W}^{\mathrm{pol}} - \mathbf{W}^{\mathrm{quad}})\mathbf{u}^{\mathrm{pol}}$ can be estimated as in Lemma 4, see below. Using (4.11), the defect is controlled via

$$\|\mathbf{d}\|^2 \leq \|W^{\mathrm{pol}}\|^2 \|\mathcal{P}_{\mathcal{K}}^{\perp}\psi^{\mathrm{pol}}\|^2 \leq C(W^{\mathrm{pol}}, d, s)K^{-s}|\psi^{\mathrm{pol}}|_s^2. \tag{4.15}$$

Integrating from 0 to t then yields the desired result. □

The following Lemma gives an estimate for the error when approximating each entry in the matrix representation of the potential part using $(K+1)$-nodes full-product Gaussian quadrature.

Lemma 4. (error due to quadrature)
Let q be a multivariate polynomial with maximal univariate degree R such that $\mathrm{supp}(q) \subset \Omega = [-S, S]^d$. *Consider a function in Hermite representation,*

$$v = \sum_{\boldsymbol{k}\in\mathbb{N}^d} v_{\boldsymbol{k}}\varphi_{\boldsymbol{k}},$$

such that $|v|_s < \infty$ for some integer s, and matrices

$$\mathbf{Q}_{\boldsymbol{jk}} = (\varphi_{\boldsymbol{j}}, q\varphi_{\boldsymbol{k}}), \quad \mathbf{Q}_{\boldsymbol{jk}}^{\mathrm{quad}} = (\varphi_{\boldsymbol{j}}, q\varphi_{\boldsymbol{k}})_{\mathrm{quad}}, \qquad \boldsymbol{j}, \boldsymbol{k} \in \mathcal{K}_{\mathrm{full}},$$

where $\mathcal{K}(d, K) = \mathcal{K}_{\mathrm{full}}(d, K)$ is the full index cube with $K \gg R$. Setting $\mathbf{v} = (v_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{K}}$, we find

$$\|(\mathbf{Q} - \mathbf{Q}^{\mathrm{quad}})\mathbf{v}\| \leq CK^{-s/2}|v|_s,$$

where $C = C(q, d, s)$.

Proof. Consider an entry of the difference matrix $(\mathbf{Q} - \mathbf{Q}^{\text{quad}})_{\mathbf{jk}}$. The error vanishes if $|\mathbf{j} + \mathbf{k}|_\infty = \max_\alpha(j_\alpha + k_\alpha) \leq 2K + 1 - R$. Hence, defining a diagonal projection matrix $\mathbf{P}$ that is related to the exactness of Gaussian quadrature,

$$\mathbf{P}_{\mathbf{jj}} = \begin{cases} 1, & |\mathbf{j}|_\infty > K + 1 - R, \\ 0, & \text{else}, \end{cases}$$

we can write

$$\mathbf{Q} - \mathbf{Q}^{\text{quad}} = \mathbf{P}(\mathbf{Q} - \mathbf{Q}^{\text{quad}})\mathbf{P},$$

which allows for

$$\|(\mathbf{Q} - \mathbf{Q}^{\text{quad}})\mathbf{v}\| \leq \|\mathbf{P}\|\big(\|\mathbf{Q}\| + \|\mathbf{Q}^{\text{quad}}\|\big)\|\mathbf{P}\mathbf{v}\|,$$

where the matrix norm is the spectral matrix norm, and $\|\mathbf{P}\| = 1$. The formal extension of $\mathbf{P}$ to an infinite matrix, with ones on the new diagonal elements, can be interpreted as the matrix representation of an operator $\mathcal{P} : L^2 \to L^2$. With this operator, we can rewrite $\mathbf{P}\mathbf{v}$ to get

$$\|\mathbf{P}\mathbf{v}\| = \|\mathcal{P}\mathcal{P}_{\mathcal{K}} v\| = \|\mathcal{P}v - \mathcal{P}\mathcal{P}_{\mathcal{K}}^{\perp} v\| \leq \|\mathcal{P}v\| + \|\mathcal{P}_{\mathcal{K}}^{\perp} v\|,$$

and, by the projection estimate (4.11),

$$\|\mathbf{P}\mathbf{v}\| \leq C(d, s)(K - R)^{-s/2}|v|_s \leq C(d, s)K^{-s/2}|v|_s.$$

This bound is conservative, but still spectrally accurate with respect to K for $K \gg R$. It is only in this estimate that we make use of $K \gg R$. Since $\mathbf{Q}$ is symmetric, the matrix norm of $\mathbf{Q}$ is readily controlled using Rayleigh quotients,

$$\begin{aligned} \|\mathbf{Q}\| &= \sup_{\|\mathbf{w}\|=1} \sum_{\mathbf{j},\mathbf{k}} w_{\mathbf{j}} \mathbf{Q}_{\mathbf{jk}} w_{\mathbf{k}} = \sup_{\|\mathbf{w}\|=1} \big(\big[\sum_{\mathbf{j}\in\mathcal{K}} w_{\mathbf{j}} \varphi_{\mathbf{j}} \big], q\big[\sum_{\mathbf{k}\in\mathcal{K}} w_{\mathbf{k}} \varphi_{\mathbf{k}} \big] \big) \\ &\leq \max_{x\in\Omega} |q(x)|^2 \leq C(q), \end{aligned}$$

The matrix norm of $\mathbf{Q}^{\text{quad}}$ is estimated as follows: Using the factorization

$$\mathbf{Q}^{\text{quad}} = \mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(q(\xi_{\mathbf{m}}))\mathbf{U}, \tag{4.16}$$

and the fact that $\mathbf{U}$ is a unitary matrix, we find

$$\|\mathbf{Q}^{\text{quad}}\| \leq \max_{\mathbf{m}\in\mathcal{K}} |q(\xi_{\mathbf{m}})| \|\mathbf{U}^T\| \|\mathbf{U}\| \leq C(q).$$

Putting everything together proves the claim. □

For a reduced index set, the above factorization (4.16) of $\mathbf{Q}^{\text{quad}}$ is no longer valid, and neither is $\mathbf{U}$ unitary any more. We can therefore not expect an analogous procedure as in the proof of Lemma 4 to be viable in case $\mathcal{K} \subsetneq \mathcal{K}_{\text{full}}$. The remaining sections are dedicated to a remedy for this difficulty.

4.4 Error decomposition for reduced index sets

In the following sections, we restrict our attention to a hyperbolically reduced index set $\mathcal{K} = \mathcal{K}_{\mathrm{hyp}}$. As will be seen in the proof of Theorem 5, the subsequent analysis is valid for any kind of index set reduction that satisfies the vague condition

$$\forall\, \mathbf{k} \in \mathcal{K}, \alpha = 1, \ldots, d: \qquad k_\alpha \approx K \quad \Rightarrow \quad \forall\, \beta \neq \alpha : k_\beta \ll k_\alpha,$$

i.e., every index has at most one component that comes close to K. This is valid for a hyperbolic and for an additive reduction, but not for the full index cube. The adaptation of the following analysis to an additive reduction is straightforward.

As explained in Section 2.6, for a reduced index set, formal insertion of the coordinate matrices $\mathbf{X}^{(\alpha)}$ into the polynomial is no longer equivalent to $(K+1)$-nodes full-product Gauß–Hermite quadrature in every entry of the potential representation $\mathbf{W}^{\mathrm{pol}}(t)$. Therefore, the solutions $\psi_{\mathcal{K}}^{\mathrm{quad}}$ and $\psi_{\mathcal{K}}^{\mathrm{fast}}$ to (4.4) and (4.7), respectively, do no longer coincide, but exhibit different dynamics. As the proof of Lemma 4 is no longer valid, a comparison of ψ^{pol} and $\psi_{\mathcal{K}}^{\mathrm{fast}}$ cannot be given by simply smuggling in $\psi_{\mathcal{K}}^{\mathrm{quad}}$. In the present section, first, we briefly explain to what extant the above analysis carries over from the case of $\mathcal{K}$ being a full index cube. Second, we give an outline of how to fill the arising gap, i.e., the invalidity of an analog to Lemma 4. The latter is due to [12].

The very ansatz for the proof of Theorem 3 carries over when comparing ψ^{pol} to $\psi_{\mathcal{K}}^{\mathrm{fast}}$ instead of $\psi_{\mathcal{K}}^{\mathrm{quad}}$ together with most of the subsequent steps. In order to make all annoying terms vanish when multiplying the error equation by the error vector and taking the real part, we have to ensure that $\mathbf{e}^*(t)W^{\mathrm{pol}}(\mathbf{X}, t)\mathbf{e}(t)$ is a real quantity, where we now set $\mathbf{e}(t) = \mathbf{u}^{\mathrm{pol}}(t) - \mathbf{c}^{\mathrm{fast}}(t)$. This is readily seen from the fact that $W^{\mathrm{pol}}(\mathbf{X}, t)$ is a real and symmetric matrix. We thus find:

Theorem 4. (global error of spatial discretization, hyperbolic reduction)
Let $\psi^{\mathrm{pol}} = \sum_{\boldsymbol{k}\in\mathbb{N}^d} u_{\boldsymbol{k}}^{\mathrm{pol}} \varphi_{\boldsymbol{k}}$ be the solution to (4.2), and let $\psi_{\mathcal{K}}^{\mathrm{fast}}$ be the solution to (4.7) with initialization $\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot, 0) = \psi_{\mathcal{K}}^{\mathrm{fast}}(\cdot, 0)$, where $\mathcal{K}(d, K)$ is a hyperbolically reduced index set. For all integers s such that

$$|\psi^{\mathrm{pol}}(\cdot, t)|_s < \infty, \qquad t \geq 0,$$

the global error due to spatial discretization is then given by

$$\begin{aligned} \|\psi^{\mathrm{pol}}(\cdot, t) - \psi_{\mathcal{K}}^{\mathrm{fast}}(\cdot, t)\| \leq CK^{-s/2}\big(|\psi^{\mathrm{pol}}(\cdot, t)|_s + \int_0^t |\psi^{\mathrm{pol}}(\cdot, \tau)|_s \,\mathrm{d}\tau\big) \\ + \int_0^t \|(\mathbf{W}^{\mathrm{pol}}(\tau) - W^{\mathrm{pol}}(\mathbf{X}, \tau))\boldsymbol{u}^{\mathrm{pol}}(\tau)\| \,\mathrm{d}\tau, \end{aligned} \qquad t \geq 0.$$

where $C = C(W^{\mathrm{pol}}, d, s)$.

There is, however, no analog of Lemma 4 for a reduced index set, which is due to the matrix $\mathbf{U}$ being no longer unitary. Hence, the integrand

$$\|(\mathbf{W}^{\mathrm{pol}}(\tau) - W^{\mathrm{pol}}(\mathbf{X}, \tau))\mathbf{u}^{\mathrm{pol}}(\tau)\|$$

as occuring on the right-hand side of the estimate given in Theorem 4 needs to be dealt with in a different way. This is done in the remainder of the present chapter.

We start with some tools and technicalities. For now, let $\mathbf{X}_{\text{full}}^{(\alpha)}$ denote the $(|\mathcal{K}_{\text{full}}|\times|\mathcal{K}_{\text{full}}|)$-coordinate matrices with respect to the full index cube $\mathcal{K}_{\text{full}}(d,K)$, while $\mathbf{X}^{(\alpha)}$ are the smaller coordinate matrices with respect to the hyperbolically reduced index set $\mathcal{K}=\mathcal{K}_{\text{hyp}}(d,K)$. We define a cutting opertor

$$\sigma : \mathbb{C}^{|\mathcal{K}_{\text{full}}|\times|\mathcal{K}_{\text{full}}|} \to \mathbb{C}^{|\mathcal{K}|\times|\mathcal{K}|}, \quad \sigma(\mathbf{A}) = (\mathbf{A}_{\mathbf{jk}})_{\mathbf{j},\mathbf{k}\in\mathcal{K}} \tag{4.17}$$

that trims a fully indexed matrix to the size of the hyperbolically reduced index set by deleting rows and columns which bear indices from $\mathcal{K}_{\text{full}}\setminus\mathcal{K}$. Slightly overloading notation, let σ denote the analogous operator for vectors instead of matrices. A complementary operation is the blow-up

$$\sigma_+ \colon \mathbb{C}^{|\mathcal{K}|} \to \mathbb{C}^{|\mathcal{K}_{\text{full}}|}, \quad (\sigma_+(\mathbf{v}))_{\mathbf{j}} = \begin{cases} v_{\mathbf{j}}, & \mathbf{j}\in\mathcal{K}, \\ 0, & \mathbf{j}\notin\mathcal{K}, \end{cases} \tag{4.18}$$

which turns a vector of reduced size into a full-sized vector by adding zeros at indices missing in $\mathcal{K}$. We need the blow-up for vectors only, not for matrices. The following rather trivial lemma relates the two operations.

Lemma 5. (cutting and blowing up)
For any matrix $\mathbf{A}^{|\mathcal{K}_{\text{full}}|\times|\mathcal{K}_{\text{full}}|}$ and any vector $\mathbf{v}\in\mathbb{C}^{|\mathcal{K}|}$, it holds

$$\sigma(\mathbf{A})\mathbf{v} = \sigma(\mathbf{A}\sigma_+(\mathbf{v})).$$

For all vectors $\mathbf{v}\in\mathbb{C}^{|\mathcal{K}|}$ and all times $t\geq 0$, we can decompose the relevant error $(\mathbf{W}^{\text{pol}}(t) - W^{\text{pol}}(\mathbf{X},t))\mathbf{v}$ according to (omitting explicit time-dependency)

$$\begin{aligned}(\mathbf{W}^{\text{pol}} - W^{\text{pol}}(\mathbf{X}))\mathbf{v} &= (\mathbf{W}^{\text{pol}} - \mathbf{W}^{\text{quad}})\mathbf{v} + (\mathbf{W}^{\text{quad}} - W^{\text{pol}}(\mathbf{X}))\mathbf{v} \\ &= (\mathbf{W}^{\text{pol}} - \mathbf{W}^{\text{quad}})\mathbf{v} \\ &\quad + \underbrace{\big(\mathbf{W}^{\text{quad}} - \sigma(W^{\text{pol}}(\mathbf{X}_{\text{full}}))\big)}_{\star}\,\mathbf{v} + \big(\sigma(W^{\text{pol}}(\mathbf{X}_{\text{full}})) - W^{\text{pol}}(\mathbf{X})\big)\mathbf{v},\end{aligned}$$

where the difference $\star$ vanishes by virtue of Lemma 2. This separates the two sources of error, i.e., quadrature and index set reduction. The error due to quadrature is given by

$$\mathbf{e}^{\text{quad}}(\mathbf{v},t) = (\mathbf{E}_{\mathbf{jk}}(t))_{\mathbf{j},\mathbf{k}\in\mathcal{K}}\,\mathbf{v}, \qquad \mathbf{E}_{\mathbf{jk}}(t) = \mathbf{W}^{\text{pol}}_{\mathbf{jk}}(t) - \mathbf{W}^{\text{quad}}_{\mathbf{jk}}(t). \tag{4.19}$$

Using Lemma 5, the error due to grid reduction can be expressed as (omitting again time-dependency at various occurrences for the sake of readability)

$$\mathbf{e}^{\text{red}}(\mathbf{v},t) = (\mathbf{e}_{\mathbf{j}}(t))_{\mathbf{j}\in\mathcal{K}}, \qquad \begin{aligned}\mathbf{e}_{\mathbf{j}} &= \big(\mathbf{W}^{\text{quad}}\mathbf{v}\big)_{\mathbf{j}} - \big(W^{\text{pol}}(\mathbf{X})\mathbf{v}\big)_{\mathbf{j}} \\ &= \big(\sigma(W^{\text{pol}}(\mathbf{X}_{\text{full}}))\mathbf{v}\big)_{\mathbf{j}} - \big(W^{\text{pol}}(\mathbf{X})\mathbf{v}\big)_{\mathbf{j}} \\ &= \big(W^{\text{pol}}(\mathbf{X}_{\text{full}})\sigma_+(\mathbf{v})\big)_{\mathbf{j}} - \big(W^{\text{pol}}(\mathbf{X})\mathbf{v}\big)_{\mathbf{j}}.\end{aligned} \tag{4.20}$$

We eventually find the error decomposition

$$(\mathbf{W}^{\text{pol}}(t) - W^{\text{pol}}(\mathbf{X},t))\mathbf{v} = \mathbf{e}^{\text{quad}}(\mathbf{v},t) + \mathbf{e}^{\text{red}}(\mathbf{v},t).$$

The error (4.19) due to quadrature is analyzed in Section 4.6. The error (4.20) due to index set reduction is analyzed in Section 4.7.

4.5 Decay assumption

To facilitate the subsequent error analysis, we need to convert the regularity assumption on ψ^{pol} from Theorem 4, viz.,

$$|\psi^{\mathrm{pol}}(\cdot,t)|_s < \infty, \qquad t \geq 0,$$

for some $s \in \mathbb{N}$, into a decay condition on the coefficient vector $\mathbf{u}^{\mathrm{pol}} = (u_{\mathbf{k}}^{\mathrm{pol}}(t))_{\mathbf{k}\in\mathcal{K}}$ corresponding to $\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}$. This is done in the present section.

For a general vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$, we aim for a decay condition of the form

$$v_{\mathbf{k}} \leq C \prod_{\alpha=1}^{d} \max\{k_\alpha, 1\}^{-r}, \qquad \mathbf{k} \in \mathcal{K}, \tag{4.21}$$

with some decay parameter $r \in \mathbb{N}$. In the following, we motivate this assumption if $\mathbf{v}$ is interpreted as the coefficient vector of the Schwartz function

$$v = \sum_{\mathbf{k}\in\mathcal{K}} v_{\mathbf{k}}\varphi_{\mathbf{k}}$$

In particular, we comment on how the constant C depends on v and how the decay parameter r is related to the regularity of v. Briefly, the above decay condition (4.21) reflects the natural decay behavior of the coefficients in a product Hermite expansion of a sufficiently smooth function. Obviously, the larger the index components, the faster the decay. Explaining this in more detail, we start with the so-called ladder operators

$$(A^-\chi)(x) = \frac{1}{\sqrt{2}}\left(x + \frac{\mathrm{d}}{\mathrm{d}x}\right)\chi(x), \qquad (A^\dagger\chi)(x) = \frac{1}{\sqrt{2}}\left(x - \frac{\mathrm{d}}{\mathrm{d}x}\right)\chi(x). \tag{4.22}$$

Their name comes from the fact that they relate the univariate Hermite functions to one another via

$$\varphi_{k+1} = \frac{1}{\sqrt{k+1}}A^\dagger\varphi_k, \qquad \varphi_{k-1} = \frac{1}{\sqrt{k}}A^-\varphi_k. \tag{4.23}$$

We have already encountered the lowering operator A^- by the name of A (with no raising counterpart) in (1.12). The ladder operators (4.22) are adjoint to one another on the space $\mathcal{S}(\mathbb{R})$ of Schwartz functions, i.e.,

$$(A^\dagger\eta, \chi) = (\eta, A^-\chi) \qquad \forall\, \eta, \chi \in \mathcal{S}(\mathbb{R}), \tag{4.24}$$

which follows easily from integration by parts. In higher dimensions, defining A_α^- and $A_\alpha^\dagger$ to be the 1D ladder operators (4.22) with respect to the αth coordinate, we write

$$(A^-)^{\mathbf{r}} = (A_1^-)^{r_1}\ldots(A_d^-)^{r_d}, \qquad \mathbf{r} = (r_1,\ldots,r_d) \in \mathbb{N}^d,$$

and analogously for $(A^\dagger)^{\mathbf{r}}$. In particular, the corresponding 1D relations (4.23) immediately yield

$$\begin{aligned}
\varphi_{\mathbf{k}+\mathbf{e}_\alpha} &= \frac{1}{\sqrt{k_\alpha+1}}\left(A^\dagger\right)^{\mathbf{e}_\alpha}\varphi_{\mathbf{k}} = \frac{1}{\sqrt{k_\alpha+1}}A_\alpha^\dagger\varphi_{\mathbf{k}},\\
\varphi_{\mathbf{k}-\mathbf{e}_\alpha} &= \frac{1}{\sqrt{k_\alpha}}(A^-)^{\mathbf{e}_\alpha}\varphi_{\mathbf{k}} = \frac{1}{\sqrt{k_\alpha}}A_\alpha^-\varphi_{\mathbf{k}}
\end{aligned} \tag{4.25}$$

as higher-dimensional counterparts for the ladder relations. See the references given in Section 1.2 for the above facts. These tools allow to proof the following lemma, which is the key to an understanding of the above decay assumption.

Lemma 6. (decay of Hermite expansion coefficients)
Consider a function $v = \sum_{\boldsymbol{k}\in\mathcal{K}} v_{\boldsymbol{k}}\varphi_{\boldsymbol{k}} \in \mathcal{S}(\mathbb{R}^d)$ in Hermite representation. For every fixed integer s, the expansion coefficients decay as

$$|v_{\boldsymbol{k}}| \leq \prod_{\alpha=1}^{d} (1 + (k_\alpha - s)_+)^{-s/2} \, |v|_{s;\infty}.$$

For the definition of the $|\cdot|_{s;\infty}$-norm, see again (1.20).

Proof. The proof follows [61], Theorem III.1.5. For every $\mathbf{k}$, we define the multi-index $m(\mathbf{k})$ by the condition

$$k_\alpha - m(\mathbf{k})_\alpha = (k_\alpha - s)_+, \qquad \alpha = 1, \ldots, d,$$

where $a_+ = \max\{a, 0\}$. Then, $0 \leq m(\mathbf{k}) \leq s$. We introduce the shorthand notation

$$v(\mathbf{k}, s) = \prod_{\alpha=1}^{d} \left((1 + (k_\alpha - 1)_+) \ldots (1 + (k_\alpha - s)_+)\right)^{-1/2}.$$

By repeated application of (4.24) in combination with (4.25), we can then estimate

$$\begin{aligned}
|v_{\mathbf{k}}| = |(\varphi_{\mathbf{k}}, v)| &= v(\mathbf{k}, s) \left| \left(\left(A^\dagger\right)^{m(\mathbf{k})} \varphi_{\mathbf{k}-m(\mathbf{k})}, v \right) \right| \\
&= v(\mathbf{k}, s) \left| \left(\varphi_{\mathbf{k}-m(\mathbf{k})}, (A^-)^{m(\mathbf{k})} v, \right) \right| \\
&\leq v(\mathbf{k}, s) \left\| (A^-)^{m(\mathbf{k})} v \right\| \leq \prod_{\alpha=1}^{d} (1 + (k_\alpha - s)_+)^{-s/2} \left\| (A^-)^{m(\mathbf{k})} v \right\|.
\end{aligned}$$

□

We aim for $v_{\mathbf{k}} \leq C \prod_{\alpha=1}^{d} \max\{k_\alpha, 1\}^{-r}$. In the light of Lemma 6, for some $\mathbf{k} \in \mathcal{K}$, consider the inequalities

$$(1 + (k_\alpha - s)_+)^{-s/2} \leq k_\alpha^{-r}, \tag{4.26}$$

where s is some fixed integer. For all choices of α such that $k_\alpha \geq 2$, a solution r of (4.26) satisfies

$$r \leq \frac{s}{2} \log_{k_\alpha} \left(1 + (k_\alpha - s)_+\right) \leq \frac{s}{2}.$$

If $k_\alpha = 1$, the relation (4.26) holds true for any choice of r. Thus, if s is even, we can choose $r = \frac{s}{2}$, and we eventually obtain the desired decay property

$$v_{\mathbf{k}} \leq |v|_{s;\infty} \prod_{\alpha=1}^{d} \max\{1, k_\alpha\}^{-s/2},$$

for every even integer s. If s is odd, the choice is $r = \frac{s-1}{2}$.

We return to the question of how the regularity of ψ^{pol} translates into a decay condition on $\mathbf{u}^{\mathrm{pol}}$ of the desired form (4.21). If $|\psi^{\mathrm{pol}}(\cdot,t)|_s < \infty$ for some (say, even) $s \in \mathbb{N}$, where $|\cdot|_s$ is the seminorm defined in (1.19), it follows that $|\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot,t)|_{s;\infty} < \infty$, where $|\cdot|_{s;\infty}$ is the norm defined in (1.20). From the above considerations, we then have

$$u_{\mathbf{k}}^{\mathrm{pol}}(t) \leq |\mathcal{P}_{\mathcal{K}}\psi^{\mathrm{pol}}(\cdot,t)|_{s;\infty} \prod_{\alpha=1}^{d} \max\{1,k_\alpha\}^{-s/2}. \tag{4.27}$$

4.6 Local error due to quadrature (reduced index set)

In the present section, we analyze the error due to quadrature as defined in (4.19). At the core of the below analysis is a suitable conversion of the corresponding 1D quadrature error into a binary tree. The 1D error can then be estimated using combinatorial arguments. In higher dimensions, the hyperbolic reduction allows for a reduction of the analysis to the 1D case.

Theorem 5. (local error due to quadrature, hyperbolic reduction)
Consider a vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ that satisfies the decay assumption (4.27)*, viz.,*

$$v_{\boldsymbol{k}} \leq |v|_{s;\infty} \prod_{\alpha=1}^{d} \max\{1,k_\alpha\}^{-s/2},$$

for some even integer s, where $v = \sum_{\boldsymbol{k}\in\mathcal{K}} v_{\boldsymbol{k}}\varphi_{\boldsymbol{k}}$. Let $\mathcal{K}(d,K)$ be a hyperbolically reduced index set. Then, the error (4.19) *due to quadrature behaves as*

$$\begin{aligned}\|\boldsymbol{e}^{\mathrm{quad}}(\mathbf{v},t)\| &\leq K^{d/2} \max_{\boldsymbol{j}\in\mathcal{K}} \left| \left(\left(\mathbf{W}^{\mathrm{pol}}(t) - \mathbf{W}^{\mathrm{quad}}(t) \right) \mathbf{v} \right)_{\boldsymbol{j}} \right| \\ &\leq C(\mathcal{R}, W^{\mathrm{pol}}, S, t) \cdot |v|_{s;\infty} \cdot K^{d/2} \cdot \min_{\substack{\boldsymbol{k}\in\mathcal{K},\\ \exists\alpha: k_\alpha \geq K-R+2}} \prod_{\beta=1}^{d} \max\{1,k_\beta\}^{-s/2}, \end{aligned} \qquad t \geq 0,$$

where the matrices $\mathbf{W}^{\mathrm{pol}}(t)$ and $\mathbf{W}^{\mathrm{quad}}(t)$ are defined as in (4.3) *and* (4.6)*, respectively, and the constant $C(\mathcal{R}, W^{\mathrm{pol}}, S, t)$ depends only on $\mathcal{R}(d,R)$, the regularity of $W^{\mathrm{pol}}(\cdot,t)$, the parameter S, and time t.*

As for the min in the right-hand side estimate, for each $\mathbf{k} \in \mathcal{K}$, there is at most one α such that $k_\alpha \geq K - R + 2$. This will become clear in the following proof.

The first estimate is simply the compatibility of $\|\cdot\|$ and the maximum norm—a rough estimate, admittedly. For the following binary tree analysis, the maximum norm is the natural norm, though.

The restriction that s be even is merely for ease of presentation; see the above considerations in Section 4.5.

Proof. The proof is outlined as follows. We consider a termwise quadrature approximation of W^{pol}. In the first part, we discuss the case $d = 1$. In 1D, termwise error matrices can be turned into binary trees. The binary tree representation then allows for a combinatorial treatment of the 1D quadrature error. In the second part, we consider the quadrature

error in arbitrary dimensions, which we can reduce to the 1D case by the definition of the hyberbolical reduction and invoke the combinatorial results.

Separating the terms in W^{pol}, viz.

$$\mathbf{E}_{\mathbf{jk}}(t) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t)\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}}, \qquad \mathbf{j},\mathbf{k}\in\mathcal{K}, \quad t\geq 0, \tag{4.28}$$

gives rise to termwise error matrices

$$\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}(S\cdot), T_{\mathbf{r}}(\cdot/S)\varphi_{\mathbf{k}}(S\cdot)) - (\varphi_{\mathbf{j}}(S\cdot), T_{\mathbf{r}}(\cdot/S)\varphi_{\mathbf{k}}(S\cdot))_{\mathrm{quad}}\,.$$

The proof proceeds along the following lines: In Step 1.1, we convert the 1D quadrature error for a single term into a binary tree. A characterization of its non-vanishing leaves is given in Step 1.2. In Step 1.3, we count these leaves. The 1D error is finally accumulated in Step 1.4. Then, in multiple dimensions, we decompose the error into appropriate 1D error term (Step 2.1), before it can eventually be estimated in Step 2.2 by reduction to the 1D case.

Step 1.1: Conversion of 1D error into binary tree: In 1D, we invoke the Chebyshev recurrence (1.25) together with the Hermite recurrence (1.10) for the right-hand side argument to obtain a decomposition of Chebyshev–Hermite products

$$\begin{aligned}
&\varphi_j(x)T_r(x/S)\varphi_k(x) = \\
&\quad\left(\frac{\sqrt{2(k+1)}}{S}\varphi_j(x)T_{r-1}(x/S)\varphi_{k+1}(x)\right) \\
&\quad+\left(\frac{\sqrt{2k}}{S}\varphi_j(x)T_{r-1}(x/S)\varphi_{k-1}(x) - \varphi_j(x)T_{r-2}(x/S)\varphi_k(x)\right)
\end{aligned} \tag{4.29}$$

into three single terms, which we group as above. Due to $S \geq \sqrt{2(K+1)}+1$, see (1.23), all occurring coefficients are bounded by

$$\frac{\sqrt{2k}}{S} \leq \frac{\sqrt{2(k+1)}}{S} \leq 1. \tag{4.30}$$

Considering the total polynomial degrees of the Chebyshev–Hermite products in each term, we realize that the first term has an unaltered degree (shifting one power of x from the Chebyshev factor to the second Hermite factor), while the latter terms both exhibit a reduction of degree by 2. This is reflected by the bracketing. The observed changes in polynomial degrees when applying the recurrences is the starting point for converting (4.29) into a binary tree. Interpreting $\varphi_j T_r \varphi_k$ as the root of a binary tree, we define the first term on the right-hand side of (4.29) to be its left child, and the latter terms to be its right child. To each child, we apply the same procedure recursively. An r-fold application of (4.29) to each child then yields the binary tree pattern $\mathfrak{T}$ of depth r as given in Figure 4.1, where we omit all arguments and coefficients. Descending to the right reduces the polynomial degree by 2, descending to the left leaves it unaltered. In case $r > k$, we may define $\varphi_k(x) = 0$ for $k \leq -1$, preserving the recurrence relation (1.10) for negative indices. We expand termwise until each leaf bears a single term of the form $\varphi_j\varphi_{k+\lambda-\rho}$, where ρ and λ are the numbers of index-changing right and left descents, respectively, and $\lambda+\rho \leq r$. By the same procedure for the corresponding quadrature formulas instead of the exact integrals, we convert both

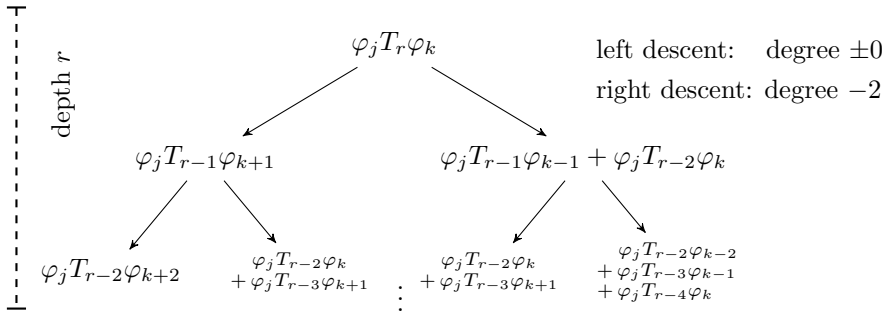


Figure 4.1: [12] Expansion of 1D quadrature error as a binary tree. In case $r=1$, we just use (1.10) on a term. In case $r=0$, terms are added to the left child without expanding.

terms in E^r_{jk} into a binary tree $\mathfrak{T}$ of depth r. Each leaf in $\mathfrak{T}$ then bears the difference of a 1D integral with the integrand being a product of two Hermite functions only (thus, the Chebyshev polynomial is gone by recursive application of (4.29)) and the corresponding quadrature formula.

As next steps, we characterize all non-vanishing leaves in $\mathfrak{T}$. As it turns out, at a non-vanishing leaf, the exact integral is zero. Then, we examnie how many these leaves are, and what quantity they sum up to.

Step 1.2: Characterization of non-vanishing leaves: Consider a fixed leaf in $\mathfrak{T}$ connected to the root E^r_{jk} by ρ and λ right and left descents, respectively. For the quadrature error not to vanish, we require $2(K+1) \leq j+k+\lambda-\rho$ for each single term, by the exactness properties of Gaussian quadrature. For the exact integral not to vanish by orthogonality of the Hermite functions, it is required $j = k+\lambda-\rho$. Together, this leads to the contradiction

$$2(K+1) \leq j+k+\lambda-\rho = 2j.$$

Thus, leaves with non-vanishing quadrature errors bear only the quadrature formulas, but not the exact integrals.

Step 1.3: Number of non-vanishing leaves: This is where the combinatorics comes into play. For a non-vanishing quadrature error at a particular leaf, we have

$$2(K+1) \leq j+k+\lambda-\rho \leq j+k+r-2\rho.$$

This motivates the definition

$$\rho_{\max}(j,k,r) = \left\lfloor \frac{j+k+r-2(K+1)}{2} \right\rfloor_+ \leq \frac{r}{2} - 1,$$

which is the maximal number of right descents that does not reduce the polynomial degree of the integrand to the extent that the quadrature becomes exact, where $a_+ = \max\{a, 0\}$.

In general, in an arbitrary full binary tree of depth r, the number of leaves connected to the root by a path containing exactly s right descents equals $\binom{r}{s}$. This can easily be seen as follows: Consider Figure 4.2, showing a full binary tree of height r with its left and right subtrees of height $r-1$ each attached to its root. Clearly, all relevant trees are those being connected by s right descents in the left and by $s-1$ right descents in the right subtree, thus, our statement is just a reformulation of the binomial recursion formula

$$\binom{r}{s} = \binom{r-1}{s} + \binom{r-1}{s-1}.$$

Hence, the number of non-vanishing leaves in $\mathfrak{T}$ is given by

$$a(j,k,r) = \sum_{s=0}^{\rho_{\max}(j,k,r)} \binom{r}{s} \tag{4.31}$$

We are looking for an upper bound for $a(j,k,r)$ that does not depend on j or k, but only on the polynomial degree r. To investigate this, consider the sum $\sum_{s=a}^{b}\binom{c}{s}$ with some $a,b,c \in \mathbb{N}$ such that $a \le 2b < c$. For $s \le b$, we find

$$\binom{c}{s} \Big/ \binom{c}{b} = \frac{b!(c-b)!}{s!(c-s)!} = \frac{b\cdot\ldots\cdot(s+1)}{(c-s)\cdot\ldots\cdot(c-b+1)} < \left(\frac{b}{c-b+1}\right)^{b-s}. \tag{4.32}$$

The assumption $c > 2b$ implies $b/(c-b+1) < 1$. Therefore, using (4.32),

$$\sum_{s=a}^{b}\binom{c}{s} = \binom{c}{b}\sum_{s=a}^{b}\binom{c}{s}\Big/\binom{c}{b} < \binom{c}{b}\sum_{s=a}^{b}\left(\frac{b}{c-b+1}\right)^{b-s} < \binom{c}{b}(b-a+1). \tag{4.33}$$

By definition, we have $2\rho_{\max} < r$, thus, using (4.33) for an estimation of (4.31), we find

$$a(j,k,r) < \binom{r}{\left\lfloor \frac{r}{2} - 1 \right\rfloor_+}\frac{r}{2} < C\frac{1}{\sqrt{2\pi}}2^r\sqrt{r}. \tag{4.34}$$

At best, $j+k+r = 2K+2$, and we have $a(j,k,r) = 1$. At worst, $j,k = K$, thus, $\rho_{\max} = \left\lfloor \frac{r}{2} - 1 \right\rfloor_+$, which makes the first estimate in (4.34) almost sharp. The second estimate is due to Stirling's formula.

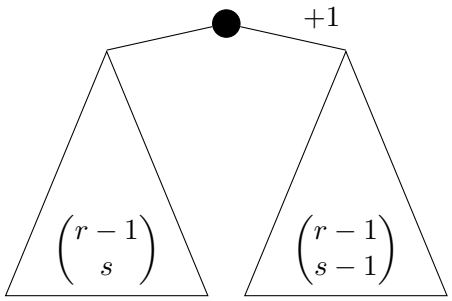


Figure 4.2: A binary tree of depth r with its left and right subtrees of height $r-1$ each attached to its root.

Step 1.4: Error accumulation in 1D: Taking into account the facts that the occurring coefficients in the binary tree are bounded by 1 (see (4.30)), and that exact integrals vanish at non-vanishing leaves, we sum up all non-vanishing leaves and, together with the bound (4.34), we obtain

$$\left|E_{jk}^r\right| \le a(j,k,r)\cdot\max_{j,k}\left|\sum_{m=0}^{K}\omega_m\varphi_j(\xi_m)\varphi_k(\xi_m)\right| \le C\frac{1}{\sqrt{2\pi}}2^r\sqrt{r} \tag{4.35}$$

where the maximum ranges over all j and k such that

$$0 \le j \le K, \qquad -r \le k \le K+r, \qquad j+k \ge 2K+2,$$

and the constant is independent of j, k, r, or K. Due to cancellation effects by Hermite function evaluations with rapidly alternating signs, the absolute value of the quadrature

formula is of size $\mathcal{O}(1)$. This concludes the first part of the proof
Step 2.1: Decomposition of error in multiple dimensions: We set

$$\mathcal{D} = \{1, \ldots, d\}$$

and consider the error matrices $\mathbf{E}^{\mathbf{r}}$ for $d \geq 2$. For arbitrary $\mathbf{j}, \mathbf{k} \in \mathcal{K}$ and $\mathbf{r} \in \mathcal{R}$,

$$\forall\, \alpha \in \mathcal{D}: \qquad k_\alpha \leq K - r_\alpha + 1 \quad \Rightarrow \quad j_\alpha + k_\alpha + r_\alpha \leq 2K + 1.$$

If the left-hand side condition holds for all $\alpha \in \mathcal{D}$, the full-product quadrature is exact in every coordinate, hence, $E^{\mathbf{r}}_{\mathbf{jk}} = 0$. Conversely, for fixed $\mathbf{j} \in \mathcal{K}$ and $\mathbf{r} \in \mathcal{R}$, if there is an index $\mathbf{k} \in \mathcal{K}$ such that $\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}} \neq 0$, we find a subset of components

$$\tilde{\mathcal{D}} = \tilde{\mathcal{D}}(\mathbf{k}) = \{\alpha \in \mathcal{D} \,:\, k_\alpha \geq K - r_\alpha + 2\} \subseteq \mathcal{D},$$

and the corresponding 1D quadrature errors $E^{r_\alpha}_{j_\alpha k_\alpha}$ do not vanish, for all $\alpha \in \tilde{\mathcal{D}}$. This allows for a decomposition (omitting the factor S^{-1} in the Chebyshev arguments)

$$\begin{aligned} E^{\mathbf{r}}_{\mathbf{jk}} &= (\varphi_{\mathbf{j}}, T_{\mathbf{r}}\varphi_{\mathbf{k}}) - (\varphi_{\mathbf{j}}, T_{\mathbf{r}}\varphi_{\mathbf{k}})_{\mathrm{quad}} \\ &= \left[\prod_{\alpha=1}^{d} (\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})\right] - \left[\prod_{\alpha=1}^{d} (\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})_{\mathrm{quad}}\right] \\ &= \Bigg\{ \underbrace{\left[\prod_{\alpha\in\tilde{\mathcal{D}}} (\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})\right]}_{\mathbf{A}} - \underbrace{\left[\prod_{\alpha\in\tilde{\mathcal{D}}} (\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})_{\mathrm{quad}}\right]}_{\mathbf{B}} \Bigg\} \underbrace{\left[\prod_{\alpha\notin\tilde{\mathcal{D}}} (\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})\right]}_{\mathbf{C}} \end{aligned} \tag{4.36}$$

of a non-vanishing entry $E^{\mathbf{r}}_{\mathbf{jk}}$. Because the indices from $\mathcal{D} \setminus \tilde{\mathcal{D}}$ stand for the coordinates with exact quadrature, the exact integral factors in $\mathbf{C}$ coincide with their quadrature counterparts.

By the definition of the hyperbolically reduced set $\mathcal{K}$, non-vanishing errors $\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}}$ have indices $\mathbf{k}$ that need to satisfy

$$\begin{aligned} K + 1 &\geq \prod_{\alpha\in\mathcal{D}} (k_\alpha + 1) \\ &= \left(\prod_{\alpha\notin\tilde{\mathcal{D}}} (k_\alpha + 1)\right) \left(\prod_{\alpha\in\tilde{\mathcal{D}}} (k_\alpha + 1)\right) \\ &\geq \left(\prod_{\alpha\notin\tilde{\mathcal{D}}} (k_\alpha + 1)\right) \left(\prod_{\alpha\in\tilde{\mathcal{D}}} (K - r_\alpha + 3)\right). \end{aligned}$$

For every $\mathbf{k} \in \mathcal{K}$, if $K \gg |\mathbf{r}|_\infty$, this is only possible if there is at most one large component k_α, thus,

$$|\tilde{\mathcal{D}}(\mathbf{k})| \leq 1.$$

Consequently, the terms $\mathbf{A}$ and $\mathbf{B}$ as occurring in the decomposition (4.36) consist of exactly one factor each, and $\mathbf{A} - \mathbf{B}$ equals the one-dimensional quadrature error $E^{r_{\alpha_0}}_{j_{\alpha_0} k_{\alpha_0}}$ for some

$\alpha_0 \in \tilde{\mathcal{D}}$. As a side note, all choices of index set reductions that yield indices with at most one large (i.e., close to K) index component are viable options in order to deduce $|\tilde{\mathcal{D}}| \leq 1$.
Step 2.2: Error estimate in multiple dimensions: Consider a non-vanishing entry $\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}}$ of the error matrix. By the above considerations for the one-dimensional case, the term $\mathbf{A}$ vanishes, since leaves with non-vanishing quadrature errors have vanishing exact integrals. Due to $|\tilde{\mathcal{D}}(\mathbf{k})| = 1$, there is $\alpha_0 \in \tilde{\mathcal{D}}$ such that $\mathbf{B}$ equals $E^{r_{\alpha_0}}_{j_{\alpha_0}k_{\alpha_0}}$. For a single factor in $\mathbf{C}$, using Cauchy–Schwarz, we find

$$|\,(\varphi_{j_\alpha}, T_{r_\alpha}\varphi_{k_\alpha})\,| \leq C(r_\alpha), \qquad \alpha \notin \tilde{\mathcal{D}}.$$

Thus, from (4.36) and (4.35), we have

$$\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}} \leq C E^{r_{\alpha_0}}_{j_{\alpha_0}k_{\alpha_0}} \leq C 2^{r_{\alpha_0}} \sqrt{r_{\alpha_0}}.$$

For a hyperbolically reduced index set $\mathcal{K}(d,K)$ with $R \leq \frac{K-1}{2} + 2$ (which we consider implied by $R \ll K$), it is easily seen that the total number of non-vanishing entries in $\mathbf{E}^{\mathbf{r}}$ is at most $\frac{1}{2}|\mathbf{r}|^2_\infty$. Multiplying the matrix with a rapidly decaying vector and using the decay condition (4.27) then yields

$$\begin{aligned}
(\mathbf{E}^{\mathbf{r}}\mathbf{v})_{\mathbf{j}} &\leq \sum_{\mathbf{k}\in\mathcal{K}} \mathbf{E}^{\mathbf{r}}_{\mathbf{jk}} v_{\mathbf{k}} \\
&\leq C \cdot \frac{1}{2}|\mathbf{r}|^2_\infty \cdot 2^{|\mathbf{r}|_\infty} |\mathbf{r}|^{1/2}_\infty \cdot \min_{\substack{\mathbf{k}\in\mathcal{K} \\ \mathbf{E}^{\mathbf{r}}_{\mathbf{jk}}\neq 0}} |v_{\mathbf{k}}| \\
&\leq C \cdot 2^{|\mathbf{r}|_\infty - 1} |\mathbf{r}|^{5/2}_\infty \cdot |v|_{s;\infty} \cdot \min_{\substack{\mathbf{k}\in\mathcal{K}, \\ \exists\alpha: k_\alpha \geq K - r_\alpha + 2}} \prod_{\beta=1}^{d} \max\{1, k_\beta\}^{-s/2}.
\end{aligned}$$

Due to $|\tilde{\mathcal{D}}(\mathbf{k})| \leq 1$, for every $\mathbf{k}$ with $\mathbf{E}^{\mathbf{r}}_{\mathbf{jk}} \neq 0$, there is exactly one α such that $k_\alpha \geq K - r_\alpha + 2$. Summing up all the terms in (4.28), the time- and S-dependent, rapidly decaying interpolation coefficients enter into the constant. This yields

$$\begin{aligned}
\mathbf{e}^{\mathrm{quad}}(\mathbf{v})_{\mathbf{j}} &\leq \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t)\, (\mathbf{E}^{\mathbf{r}}\mathbf{v})_{\mathbf{j}} \\
&\leq C \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) 2^{|\mathbf{r}|_\infty - 1} |\mathbf{r}|^{5/2}_\infty \cdot |v|_{s;\infty} \cdot \min_{\substack{\mathbf{k}\in\mathcal{K}, \\ \exists\alpha: k_\alpha \geq K - r_\alpha + 2}} \prod_{\beta=1}^{d} \max\{1, k_\beta\}^{-s/2},
\end{aligned} \tag{4.37}$$

which proves the claim. □

4.7 Local error due to index set reduction

In this last section, we analyze the error due to index set reduction as defined in (4.20). Once more, we employ the strategy of converting analytic error formulations into binary trees in order to obtain estimates by combinatorial arguments. This time, we do not reduce the multidimensional error to 1D (because there is no index set reduction in 1D), but construct more complicated binary trees dealing with the coordinates sequentially.

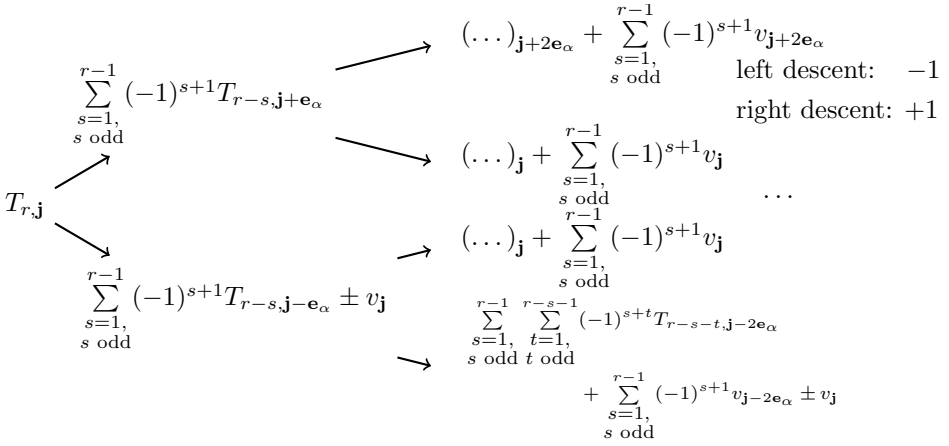


Figure 4.3: [12] Expansion of $T^{(\alpha)}_{r,\mathbf{j}}$ as a binary tree $\mathfrak{T}^{(\alpha)}(\mathbf{j})$ of depth r. We draw the case of r being even. Terms with reduced or unaltered index are attached to the left child, otherwise to the right child. The figure is rotated by 90° and all coefficients have been omitted.

Theorem 6. (local error due to index set reduction, hyperbolic reduction)
Consider a vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ *that satisfies the decay assumption* (4.27)*, viz.,*

$$v_{\boldsymbol{k}} \le |v|_{s;\infty} \prod_{\alpha=1}^{d} \max\{1, k_\alpha\}^{-s/2},$$

for some even integer s*, where* $v = \sum_{\boldsymbol{k}\in\mathcal{K}} v_{\boldsymbol{k}} \varphi_{\boldsymbol{k}}$*. Let* $\mathcal{K}(d,K)$ *be a hyperbolically reduced index set. Then, the error* (4.20) *due to index set reduction behaves as*

$$\begin{aligned} \|\boldsymbol{e}^{\mathrm{red}}(\mathbf{v},t)\| &\le K^{d/2} \max_{\boldsymbol{j}\in\mathcal{K}} \left| \left(W^{\mathrm{pol}}(\mathbf{X}_{\mathrm{full}},t)\sigma_+(\mathbf{v}) - W^{\mathrm{pol}}(\mathbf{X},t)\mathbf{v} \right)_{\boldsymbol{j}} \right| \\ &\le C(d,\mathcal{R},W^{\mathrm{pol}},S,t) \cdot |v|_{s;\infty} \cdot K^{d/2} \cdot \min_{\substack{\boldsymbol{j}\in\mathcal{K},\boldsymbol{r}\in\mathcal{R},\\ \boldsymbol{j}+\boldsymbol{r}\in\mathcal{K}_{\mathrm{full}}\setminus\mathcal{K}}} \prod_{\beta=1}^{d} \max\{1, j_\beta - r_\beta\}^{-s/2}, \end{aligned} \qquad t \ge 0,$$

where the matrices $W^{\mathrm{pol}}(\mathbf{X}_{\mathrm{full}},t)$ *and* $W^{\mathrm{pol}}(\mathbf{X},t)$ *result from formally inserting the coordinate matrices* $\mathbf{X}^{(\alpha)}_{\mathrm{full}}$ *(indexed over the corresponding full index cube* $\mathcal{K}_{\mathrm{full}}(d,K)$*) and* $\mathbf{X}^{(\alpha)}$ *into* W^{pol}*, respectively; see Section 2.3. The constant* $C(d,\mathcal{R},W,S,t)$ *depends only on* $\mathcal{R}(d,R)$*, the regularity of* W^{pol}*, the parameter* S*, and the time* t*—and on* d*, in contrast to the constant as occurring in the previous theorem.*

Proof. As in the previous theorem, we consider a termwise partition of the error

$$\mathbf{e}^{\mathrm{red}}(\mathbf{v}) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t) \left[T_{\mathbf{r}}\left(\tfrac{1}{S}\mathbf{X}_{\mathrm{full}}\right)\sigma_+(\mathbf{v}) - T_{\mathbf{r}}\left(\tfrac{1}{S}\mathbf{X}\right)\mathbf{v} \right],$$

see (4.20), and analyze the terms separately. We fix $\mathbf{r}$ for the rest of the proof.
Again, we split the proof into smaller parts: In Step 1, we explain how to construct binary tress both for the case of a reduced and of a full index set. The key idea is to consider differences of these trees (Step 2) and to characterize which leaves do not vanish in a suitable difference tree (Step 3). In Step 4, we count these non-vanishing leaves, before the overall

error can eventually be given by putting everything together in Step 5.
Step 1: Construction of binary trees: We use the abbreviations

$$T^{(\alpha)}_{r,\mathbf{j}} = \left(T_r\left(\tfrac{1}{S}\mathbf{X}^{(\alpha)}\right)\mathbf{v}\right)_{\mathbf{j}}, \qquad \begin{matrix}\alpha = 1,\dots,d,\\ r = 1,\dots,R.\end{matrix}$$

Fix α and r. The 1D Chebyshev recurrence (1.25) together with the direct operation procedure (2.5) yields

$$T^{(\alpha)}_{r,\mathbf{j}} = \frac{\sqrt{2j_\alpha}}{S}T^{(\alpha)}_{r-1,\mathbf{j}-\mathbf{e}_\alpha} + \frac{\sqrt{2(j_\alpha+1)}}{S}T^{(\alpha)}_{r-1,\mathbf{j}+\mathbf{e}_\alpha} - T^{(\alpha)}_{r-2,\mathbf{j}},$$

and, after repeated application with the right-most term,

$$T^{(\alpha)}_{r,\mathbf{j}} = \frac{\sqrt{2j_\alpha}}{S}\sum_{\substack{s=1,\\ s\text{ odd}}}^{r-1}(-1)^{s+1}T_{r-s,\mathbf{j}-\mathbf{e}_\alpha} + \frac{\sqrt{2(j_\alpha+1)}}{S}\sum_{\substack{s=1,\\ s\text{ odd}}}^{r-1}(-1)^{s+1}T_{r-s,\mathbf{j}+\mathbf{e}_\alpha} + \tau^{(\alpha)}_{r,\mathbf{j}}, \tag{4.38}$$

where

$$\tau^{(\alpha)}_{r,\mathbf{j}} = \begin{cases} v_{\mathbf{j}}, & r \text{ even and } 4 \mid r,\\ -v_{\mathbf{j}}, & r \text{ even and } 4 \nmid r,\\ \frac{1}{S}\left(\sqrt{\frac{j_\alpha}{2}}v_{\mathbf{j}-\mathbf{e}_\alpha} + \sqrt{\frac{j_\alpha+1}{2}}v_{\mathbf{j}+\mathbf{e}_\alpha}\right), & r \text{ odd and } 4 \mid (r+1),\\ -\frac{1}{S}\left(\sqrt{\frac{j_\alpha}{2}}v_{\mathbf{j}-\mathbf{e}_\alpha} + \sqrt{\frac{j_\alpha+1}{2}}v_{\mathbf{j}+\mathbf{e}_\alpha}\right), & r \text{ odd and } 4 \nmid (r+1). \end{cases}$$

We interpret $T^{(\alpha)}_{r,\mathbf{j}}$ as a node in a binary tree, where all the terms on the right-hand side of (4.38) with their index components j_α reduced by one or left unaltered are defined as the left child, while all the terms with their index components j_α incremented by 1 make up the right child. This procedure is repeated, and we end up with a binary tree $\mathfrak{T}^{(\alpha)}(\mathbf{j})$ of depth r that bears only sums of weighted components of $\mathbf{v}$ at its leaves. We illustrate the construction in Figure 4.3, where we omit all coefficients for the sake of simplicity. With each left or right descent, all indices of newly expanded terms have their αth component reduced or increased by 1, respectively. A simple calculation reveals that leaves exhibit sums of less than $\frac{1}{2}r(r-1)$ terms.

Starting from some $\mathbf{j} \in \mathcal{K}$, a binary tree $\mathfrak{T}(\mathbf{j})$ for the product

$$\left(T_{\mathbf{r}}\left(\tfrac{1}{S}\mathbf{X}\right)\mathbf{v}\right)_{\mathbf{j}} = \left(T_{r_1}\left(\tfrac{1}{S}\mathbf{X}^{(1)}\right)\cdot\left(\dots\left(T_{r_d}\left(\tfrac{1}{S}\mathbf{X}^{(d)}\right)\mathbf{v}\right)\dots\right)\right)_{\mathbf{j}}$$

is then obtained by attaching to each term in each leaf of $\mathfrak{T}^{(\alpha)}$ analogously defined trees $\mathfrak{T}^{(\alpha+1)}$ starting from $\alpha = 1$, such that a leaf in layer α is a root of a subtree in layer $\alpha+1$; see the pattern given in Figure 4.4, where the topmost and lowermost layers are numbered 1 and d, respectively. Along the path to a proper leaf in layer d (a *d-leaf*), let λ_α and ρ_α denote the number of left and right descents in layer α, respectively. Starting from $\mathbf{j}$, in layer α, only the αth component of $\mathbf{j}$ is changed. Thus, d-leaves exhibit sums of weighted components of $\mathbf{v}$ whose component indices are of the form

$$\mathbf{j} - \sum_{\alpha=1}^{d}\tilde{\lambda}_\alpha + \sum_{\alpha=1}^{d}\tilde{\rho}_\alpha, \qquad \tilde{\lambda}_\alpha, \tilde{\rho}_\alpha \in \{0,1\}.$$

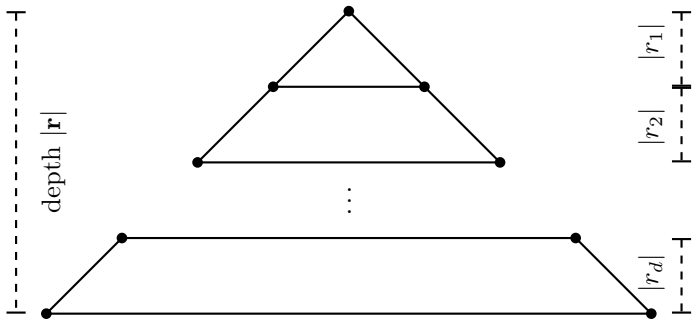


Figure 4.4: [12] Forming $\mathfrak{T}$ by layerwise attachments of $\mathfrak{T}^{(\alpha)}$.

The very same considerations also apply with an unreduced index set, i.e., with $\mathbf{X}^{(\alpha)}_{\text{full}}$ in place of $\mathbf{X}^{(\alpha)}$ and with $\sigma_+(\mathbf{v})$ in place of $\mathbf{v}$, yielding an analogously defined binary tree $\mathfrak{T}_{\text{full}}(\mathbf{j})$. By the definition (4.20) of the error due to index set reduction, we are interested in root indices $\mathbf{j} \in \mathcal{K}$.

Step 2: Difference of binary trees. By (4.20), we consider the difference tree

$$\mathfrak{D}(\mathbf{j}) = \mathfrak{T}(\mathbf{j}) - \mathfrak{T}_{\text{full}}(\mathbf{j}), \qquad \mathbf{j} \in \mathcal{K},$$

based on the vectors $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ and $\sigma_+(\mathbf{v})$, where we fix $\mathbf{j}$ as the root index in layer 1 from now on. Let us introduce some terminology: Consider an index appearing in a term somewhere along a path from the 1-root $\mathbf{j}$ to one of its d-leaves. If this index or any index of a predecessor term does not belong to $\mathcal{K}$ or $\mathcal{K}_{\text{full}}$, we say that the corresponding terms *vanish* in $\mathfrak{T}(\mathbf{j})$ or $\mathfrak{T}_{\text{full}}(\mathbf{j})$, respectively. A term in the difference tree $\mathfrak{D}(\mathbf{j})$ *vanishes* if corresponding terms in $\mathfrak{T}(\mathbf{j})$ and $\mathfrak{T}_{\text{full}}(\mathbf{j})$ vanish or do not vanish both at the same time in their respective trees. Finally, a d-leave in $\mathfrak{D}(\mathbf{j})$ is said to *vanish* if all terms in the leave vanish. As in the proof of the previous theorem, we examine non-vanishing d-leaves in $\mathfrak{D}(\mathbf{j})$. We state the following obvious, yet important observation: If a d-leave does not vanish, there is at least one term somewhere along the path connecting it to the root such that the corresponding index belongs to $\mathcal{K}_{\text{full}} \setminus \mathcal{K}$. In particular, for a fixed choice of $\mathbf{r}$, the existence of a non-vanishing d-leaf implies $\mathbf{j} + \mathbf{r} \in \mathcal{K}_{\text{full}} \setminus \mathcal{K}$.

Step 3: Characterization of non-vanishing leaves: We consider a root index $\mathbf{m}$ in some fixed layer α, where $m_\alpha = j_\alpha$. Our aim is to characterize indices of single α-leaf terms by means of the minimal and maximal numbers of left descents within the layer such that these leaves do not vanish. We have the following necessary condition for a term depending on $\mathbf{m}$ not to vanish in $\mathfrak{T}_{\text{full}}(\mathbf{j})$ or $\mathfrak{T}(\mathbf{j})$, respectively:

($\mathfrak{T}_{\text{full}}$) For the αth component index, it is required that

$$0 \overset{!}{\leq} j_\alpha + \rho_\alpha - \lambda_\alpha \leq j_\alpha + r_\alpha - 2\lambda_\alpha \overset{!}{\leq} K,$$

which gives the bounds

$$\lambda_\alpha^{\min}(\mathbf{r}, \mathbf{j}) = \left\lfloor \frac{j_\alpha + r_\alpha - K}{2} \right\rfloor \leq \lambda_\alpha \leq \left\lfloor \frac{j_\alpha + r_\alpha}{2} \right\rfloor = \lambda_\alpha^{\max}(\mathbf{r}, \mathbf{j}).$$

($\mathfrak{T}$) The upper bound is the same as in ($\mathfrak{T}_{\text{full}}$). By the definition of $\mathcal{K}$, one needs

$$j_\alpha + r_\alpha - 2\lambda_\alpha + 1 \overset{!}{\leq} (K+1)\Big(\prod_{\substack{\beta=1\\ \beta\neq\alpha}}^{d}(1+m_\beta)\Big)^{-1}. \tag{4.39}$$

From $\mathbf{m} \in \mathcal{K}$, it follows that

$$1 + j_\alpha \leq (K+1)\Big(\prod_{\substack{\beta=1\\ \beta\neq\alpha}}^{d}(1+m_\beta)\Big)^{-1},$$

thus, by subtraction, $r_\alpha - 2\lambda_\alpha \overset{!}{\leq} 0$. This means that the leaf term in $\mathfrak{T}(\mathbf{j})$ does not vanish for

$$\lambda_\alpha \geq \left\lceil \frac{r_\alpha}{2} \right\rceil = \lambda_\alpha^{\min,\text{hyp}}(\mathbf{r}). \tag{4.40}$$

If a leave term index in the difference tree $\mathfrak{D}(\mathbf{j})$ does not vanish, it satisfies ($\mathfrak{T}_{\text{full}}$), but it does not satisfy the more restrictive ($\mathfrak{T}$). The converse is not true, since a leaf term index violating ($\mathfrak{T}$) might still fulfill (4.39), and thus vanish in $\mathfrak{D}(\mathbf{j})$. This complicates a characterization of terms vanishing in $\mathfrak{D}(\mathbf{j})$. To make things simpler, we consider the more handy condition (4.40) to constitute the lower bound in ($\mathfrak{T}$) . Obviously,

$$\lambda_\alpha^{\min} \leq \lambda_\alpha^{\min,\text{hyp}}.$$

Step 4: Number of non-vanishing leaves: Summing up as in the proof of the previous theorem, we have at most

$$\sum_{s=\lambda_\alpha^{\min}}^{\lambda_\alpha^{\max}} \binom{r_\alpha}{s} - \sum_{s=\lambda_\alpha^{\min,\text{hyp}}}^{\lambda_\alpha^{\max}} \binom{r_\alpha}{s} = \sum_{s=\lambda_\alpha^{\min}}^{\lambda_\alpha^{\min,\text{hyp}}-1} \binom{r_\alpha}{s} = a_\alpha(\mathbf{r},\mathbf{j})$$

non-vanishing α-leaves. Once more, we use (4.33) with

$$a = \lambda_\alpha^{\min}, \qquad b = \lambda_\alpha^{\min,\text{hyp}} - 1, \qquad c = r_\alpha.$$

By virtue of $\lambda_\alpha^{\min,\text{hyp}} = \left\lceil \frac{r_\alpha}{2} \right\rceil$, we have $2(\lambda_\alpha^{\min,\text{hyp}} - 1) < r_\alpha$, hence,

$$a_\alpha(\mathbf{r},\mathbf{j}) < \binom{r_\alpha}{\lfloor \frac{r_\alpha}{2} - 1 \rfloor} \frac{r_\alpha}{2} < C\frac{1}{\sqrt{2\pi}} 2^r \sqrt{r} = a(r_\alpha).$$

Step 5: Error accumulation: As explained at the end of Step 2, a non-vanishing error yields $\mathbf{j} + \mathbf{r} \in \mathcal{K}_{\text{full}} \setminus \mathcal{K}$. In the following, we assume that our choices of $\mathbf{j}$ and $\mathbf{r}$ are such. Along the path to a non-vanishing d-leaf, at any α-leaf, the most unfavorable weight, viz.,

$$2^{r_\alpha/2} \prod_{s=1}^{r_\alpha} \frac{(j_\alpha + s)^{1/2}}{S} \leq 2^{r_\alpha/2} = b(r_\alpha)$$

comes from descending right only. Using the decay assumption (4.27) on $\mathbf{v}$, the d-leaf can be bounded by

$$\begin{aligned} c(\mathbf{r},\mathbf{j}) &\leq \prod_{\alpha=1}^{d} \frac{1}{2} r_\alpha (r_\alpha - 1) \cdot \left\{ \max_{-r_1 \leq s_1 \leq r_1} \cdot \ldots \cdot \max_{-r_d \leq s_d \leq r_d} \left| v_{\mathbf{j} - \sum_{\alpha=1}^{d} s_\alpha \mathbf{e}_\alpha} \right| \right\} \\ &\leq \prod_{\alpha=1}^{d} \frac{1}{2} r_\alpha (r_\alpha - 1) \cdot |v|_{s;\infty} \cdot \prod_{\beta=1}^{d} \max\{1, j_\beta - r_\beta\}^{-s/2}. \end{aligned}$$

Hence, the error over all layers can be bounded by

$$\begin{aligned} |E_{\mathbf{j}}^{\mathbf{r}}| &\leq \prod_{\alpha=1}^{d} \{a(r_\alpha) \cdot b(r_\alpha)\} \cdot c(\mathbf{r},\mathbf{j}) \\ &\leq C \prod_{\alpha=1}^{d} \left\{ \frac{1}{\sqrt{2\pi}} 2^{r_\alpha} \sqrt{r_\alpha} \cdot 2^{r_\alpha/2} \cdot \frac{1}{2} r_\alpha (r_\alpha - 1) \right\} \cdot |v|_{s;\infty} \cdot \prod_{\beta=1}^{d} \max\{1, j_\beta - r_\beta\}^{-s/2} \\ &\leq C d \; 2^{3/2|\mathbf{r}|_\infty} |\mathbf{r}|_\infty^{5/2} \cdot |v|_{s;\infty} \cdot \prod_{\beta=1}^{d} \max\{1, j_\beta - r_\beta\}^{-s/2}. \end{aligned}$$

Finally, we sum up and obtain

$$\mathbf{e}^{\mathrm{red}}(\mathbf{v},t)_{\mathbf{j}} \leq C \sum_{\mathbf{r} \in \mathcal{R}} \hat{w}_{\mathbf{r}}(t) \, d \; 2^{3/2|\mathbf{r}|_\infty} |\mathbf{r}|_\infty^{5/2} \cdot |v|_{s;\infty} \cdot \prod_{\beta=1}^{d} \max\{1, j_\beta - r_\beta\}^{-s/2}. \tag{4.41}$$

This proves the claim. □

4.8 Remarks on the actual decay behavior

A few remarks concerning the error in case of a reduced index set are in order. In particular, we shall comment on the actual error behavior of the local errors as analyzed in Theorems 5 and 6 when considering different components of the error vectors $\mathbf{e}^{\mathrm{quad}}(\mathbf{v},t)$ and $\mathbf{e}^{\mathrm{red}}(\mathbf{v},t)$.

- First, in the maximum norm, using $K \gg R$, the local errors $\mathbf{e}^{\mathrm{quad}}(\mathbf{v},t)$ and $\mathbf{e}^{\mathrm{red}}(\mathbf{v},t)$ both exhibit an asymptotic decay of

  $$C(\mathcal{R}, W^{\mathrm{pol}}, S, t) K^{-s/2} \quad \text{and} \quad C(d, \mathcal{R}, W^{\mathrm{pol}}, S, t) K^{-s/2},$$

  respectively.

- The next comment concerns the actual order of decay for a specific choice of error component from the local error due to index set reduction. As for the componentwise error estimates (4.37) and (4.41), it is only the decay in the estimate for $(\mathbf{e}^{\mathrm{red}}(\mathbf{v},t))_{\mathbf{j}}$ that actually depends on the component index $\mathbf{j}$.
  We have to distinguish three cases: First, for fixed $\mathbf{r} \in \mathcal{R}$, if $\mathbf{j} + \mathbf{r} \in \mathcal{K}$ (roughly, if there are only small index components in $\mathbf{j}$), all paths in the difference tree $\mathfrak{D}(\mathbf{j})$ from the proof of Theorem 6 cancel out, and the error

  $$T_{\mathbf{r}}\left(\tfrac{1}{S}\mathbf{X}\right)\mathbf{v} - T_{\mathbf{r}}\left(\tfrac{1}{S}\mathbf{X}_{\mathrm{full}}\right)\sigma_+(\mathbf{v})$$

vanishes. As is seen from Part 5 of the above proof, the actual error behavior depends on the number of large index components in $\mathbf{j}$: Roughly, if there is more than one large component in $\mathbf{j}$, say,

$$N(\mathbf{r},\mathbf{j}) \geq 2,$$

where $N(\mathbf{r},\mathbf{j}) \in \{1,\dots,d\}$ is the number of components j_α such that $K \approx j_\alpha \gg r_\alpha$, we find

$$c(\mathbf{r},\mathbf{j}) \leq C \prod_{\alpha=1}^{d} \frac{1}{2} r_\alpha (r_\alpha - 1) \cdot |v|_{s;\infty} \cdot K^{-Ns/2}.$$

Therefore, if $\tilde{N} = \min_{\mathbf{r}\in\mathcal{R}} N(\mathbf{r},\mathbf{j})$,

$$\mathbf{e}^{\mathrm{red}}(\mathbf{v},t)_{\mathbf{j}} \leq C \sum_{\mathbf{r}\in\mathcal{R}} \hat{w}_{\mathbf{r}}(t)\, d\; 2^{3/2|\mathbf{r}|_\infty} |\mathbf{r}|_\infty^{5/2} \cdot |v|_{s;\infty} \cdot K^{-\tilde{N}s/2}.$$

This is the second case. Third, if there are only intermediate-sized index components, neither of the above considerations applies. Thus, the error decay is worst away from the corners of the index cube. This behavior shall be reproduced and visualized in the numerical experiments given in Section 5.1; see Figure 5.3, in particular.

- Finally, which of the two local errors do we expect to decay faster? The constants in the termwise local error estimates (4.37) and (4.41) appear to be similar. However, the constant in the estimate for $\mathbf{e}^{\mathrm{red}}(\mathbf{v})$ depends on d, in contrast to the d-independent constant in the estimate for $\mathbf{e}^{\mathrm{quad}}(\mathbf{v})$. We therefore expect that the local error due to index set in the maximum norm worsens for increasing d, whereas the local error due to quadrature remains unaffected. Additionally, in (4.41), the reduction by r_β in the factors $\max\{1, j_\beta - r_\beta\}^{-s/2}$ slows down the decay. We therefore expect the error due to quadrature to be smaller than the error due to index set reduction. This behavior shall be reproduced in the numerical experiments; see Figure 5.1, in particular.

5 Numerical experiments

In this chapter, we present some numerical experiments that corroborate the theoretical predictions from the above error analyis. In Section 5.1, we illustrate the local error behavior when approximating the matrix-vector product

$$\mathbf{W}^{\mathrm{quad}}\mathbf{v} \approx W^{\mathrm{pol}}(\mathbf{X})\mathbf{v}, \qquad \mathbf{v} \in \mathbb{C}^{|\mathcal{K}|},$$

where

$$\mathbf{W}^{\mathrm{quad}}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, W^{\mathrm{pol}}\varphi_{\mathbf{k}})_{\mathrm{quad}}, \qquad \mathbf{j},\mathbf{k} \in \mathcal{K},$$

denotes the matrix due to entrywise quadrature approximation of the representation of the polynomially approximated potential W by $(K+1)$-nodes full-product Gauß–Hermite quadrature, and $W^{\mathrm{pol}}(\mathbf{X})$ denotes formal insertion of the coordinate matrices $\mathbf{X}^{(\alpha)}$ in place of x_α into the polynomial W^{pol}; see Section 2.6 and 2.3 for the respective details. The effect on the Galerkin approximation based on the Hermitian Lanczos process to the matrix exponentials

$$\exp(-ih(\mathbf{D} + \mathbf{W}^{\mathrm{quad}}))\mathbf{v} \quad \text{vs.} \quad \exp(-ih(\mathbf{D} + W^{\mathrm{pol}}(\mathbf{X})))\mathbf{v}$$

is shown in Section 5.2. In both sections, the potential under consideration is the time-independent torsional potential. Finally, in Section 5.3, we propagate the semidicrete systems

$$\begin{aligned} i\dot{\mathbf{c}}^{\mathrm{quad}}(t) &= \mathbf{D}\mathbf{c}^{\mathrm{quad}}(t) + \mathbf{W}^{\mathrm{quad}}(t)\mathbf{c}^{\mathrm{quad}}(t), \\ i\dot{\mathbf{c}}^{\mathrm{fast}}(t) &= \mathbf{D}\mathbf{c}^{\mathrm{fast}}(t) + W^{\mathrm{pol}}(\mathbf{X},t)\mathbf{c}^{\mathrm{fast}}(t); \end{aligned}$$

see (4.5) and (4.7), respectively, using the Magnus integrators given in Section 1.5 and compare the dynamics. The potential is the Hénon–Heiles potential with a time-dependent linear perturbation.

Throughout this section, we have mostly chosen a hyperbolic reduction for the underlying index set $\mathcal{K}$. The specific choice is always indicated.

Again, all figures have been obtained on a desktop computer with an Intel Core 2 Duo E8400 3.00 GHz processor with 4 GB RAM using an implementation in `C`. The code is provided on the author's webpage:

`https://na.uni-tuebingen.de/~brumm/`

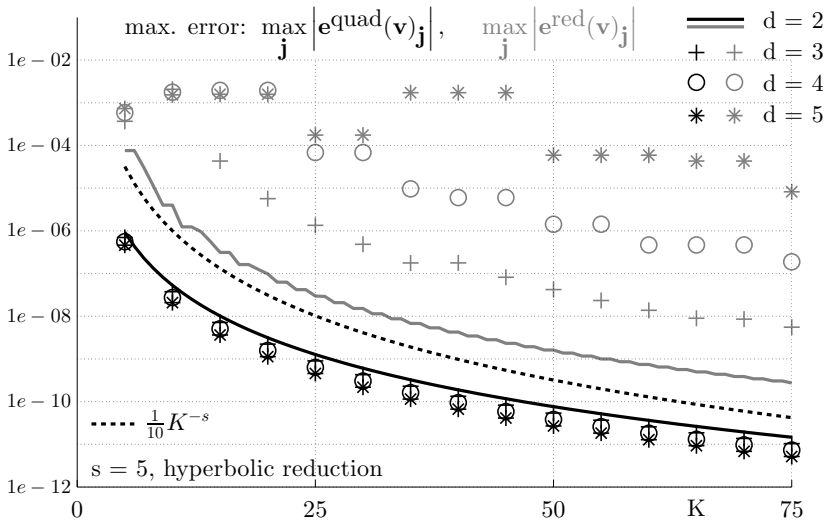


Figure 5.1: Errors $\mathbf{e}^{\mathrm{quad}}(\mathbf{v})$ (black) and $\mathbf{e}^{\mathrm{red}}(\mathbf{v})$ (gray) for a hyperbolic reduction and a vector decaying according to (5.2) ($S = 16$, $R = 8$, $s = 5$). The solid lines represent $d = 2$, selected errors in cases $d = 3, 4, 5$ are indicated by plus signs, circles, and asterisks, respectively. The dashed line represents $\frac{1}{10}K^{-s}$. Increasing d worsens the constant in $\mathbf{e}^{\mathrm{red}}(\mathbf{v})$; cf. Theorem 6. In contrast, $\mathbf{e}^{\mathrm{quad}}(\mathbf{v})$ is unaffected. Semi-logarithmic plot. Figure taken from [12] and extended.

quadrature, index set reduction				
$K \downarrow$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
25	1.270e-09	8.978e-10	6.347e-10	4.559e-10
	2.975e-08	1.345e-06	6.834e-05	1.744e-04
50	7.628e-11	5.392e-11	3.812e-11	2.695e-11
	1.621e-09	4.207e-08	1.426e-06	5.907e-05
75	1.466e-11	1.037e-11	7.328e-12	5.181e-12
	2.729e-10	5.540e-09	1.879e-07	8.227e-06

5.1 Local errors due to quadrature and index set reduction

We consider the aforementioned stretched torsional potential as given in (3.10), viz.

$$W(x) = \sum_{\alpha=1}^{d} (1 - \cos(x_\alpha / S)), \qquad x \in [-S, S]^d, S = 16, \tag{5.1}$$

approximated by Chebyshev interpolation with $R = 8$ nodes on each coordinate axis. In cases $d = 2, 3, 4$, as mentioned in Section 3.3, this yields L^2- and L^∞-interpolation errors not larger than 5e-09. This error contribution will thus be dominated by the error due to index set reduction unless the spatial resolution is chosen particularly fine.

To illustrate the local error behavior of the fast algorithm, we consider a vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ that satisfies the decay assumption (4.21), and do the corresponding matrix-vector products in the following ways. We set

$$v_{\mathbf{k}} = \prod_{\alpha=1}^{d} \max\{k_\alpha, 1\}^{-s}, \qquad \text{for some given } s \in \mathbb{N}, \tag{5.2}$$

and then normalize such that $\|\mathbf{v}\| = 1$.

First, we illustrate the error due to quadrature,

$$\max_{\mathbf{j} \in \mathcal{K}} \left|\mathbf{e}^{\mathrm{quad}}(\mathbf{v})_{\mathbf{j}}\right|, \qquad \mathbf{e}^{\mathrm{quad}}(\mathbf{v}) = (\mathbf{W}^{\mathrm{pol}} - \mathbf{W}^{\mathrm{quad}})\mathbf{v},$$

as a function of K; see (4.19). As shown in Theorem 5, the error is expected to decay spetrally, i.e., $\sim K^{-s}$, for all choices of d and K, where the constant does not depend on the dimension. This is shown in Figure 5.1 for a hyperbolic reduction with $s = 5$ (black lines and symbols). We start from $K = 10$ and proceed in steps of 1 ($d = 2$) and 5 ($d \geq 2$) until the fine resolution of $K = 75$. For different choices of d, the error behavior is indeed almost identical.

Throughout most of the following experiments, in case $d \geq 3$, we face the difficulty that both matrix-vector products $\mathbf{W}^{\mathrm{pol}}\mathbf{v}$ and $\mathbf{W}^{\mathrm{quad}}\mathbf{v}$ are too expensive to compute in a naive way, and we might thus easily lack the quantities the fast algorithm needs to be compared to. In this situation, sequential summations do us a good service and constitute an efficient means to obtain these matrix-vector products exactly when combined with a suitable choice of Gaussian quadrature, viz.,

$$M^{\mathrm{pol}} = \left\lceil \frac{2K + R - 1}{2} \right\rceil, \qquad M^{\mathrm{quad}} = K + 1.$$

With these choices of full-product quadrature thresholds, the entries of $\mathbf{W}^{\mathrm{pol}}$ and $\mathbf{W}^{\mathrm{quad}}$ are computed exactly.

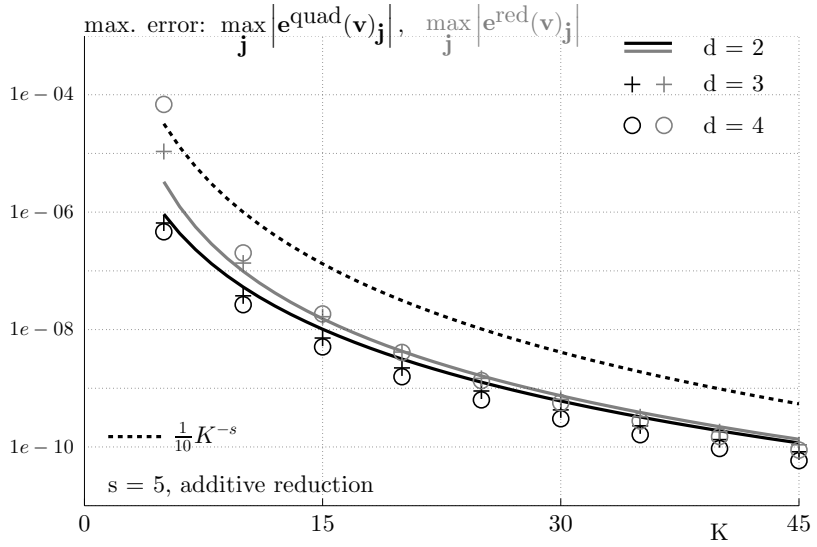
Next, we illustrate the error due to index set reduction,

$$\max_{\mathbf{j} \in \mathcal{K}} \left| \mathbf{e}^{\mathrm{red}}(\mathbf{v})_{\mathbf{j}} \right|, \qquad \mathbf{e}^{\mathrm{red}}(\mathbf{v}) = (\mathbf{W}^{\mathrm{quad}} - W^{\mathrm{pol}}(\mathbf{X}))\mathbf{v},$$

as a function of K; see (4.20) and the discussion at the end of Section 4.4. As shown in Theorem 6, the error is expected to decay spetrally as well, i.e., $\sim K^{-s}$, for all choices of d and K, but the constant depends on the dimension and worsens for increasing d. This is also shown in Figure 5.1 for a hyperbolic reduction and $s = 5$ (gray lines and symbols).

Besides the hyperbolic reduction, we also consider the additively reduced index set. Analogous local error results are shown in Figure 5.2. While the error due to quadrature is of a similar size as for the hyperbolic reduction, the error due to index set reduction now decays faster. This difference can be elucidated by recourse to the analysis given in Theorem 6 and to the subsequent remarks from Section 4.8: As explained there, for K being sufficiently large, the error $\mathbf{e}^{\mathrm{red}}(\mathbf{v})_{\mathbf{j}}$ is not evenly distributed, but varies according to the choice of $\mathbf{j}$. If $\mathbf{j}$ consists of small components j_α only (i.e., increasing by R in any component does not lead out of the index set), we expect no error due to index set reduction at all. Wherever the error does not vanish, it decreases the faster the more components are large as compared to R. Roughly spoken, for the choice of vector decay (5.2) under consideration, we expect an error decay $\sim K^{-Ns}$, where N is the number of components that are large as compared to R. For a hyperbolic reduction, this yields the largest error contributions for indices with moderately sized components only. In 2D and for not too large a choice of K, this is the region close to the boundary of the index set, but away from its flanks. For an additive reduction, errors corresponding to indices close to the flanks are expected to decay slightly slowlier than indices close to the center of the boundary line because moderately sized indices from the half-cube still bear indices that are all large as compared to R. Figure 5.3 illustrates this decay behavior in the individual components of $\mathbf{e}^{\mathrm{red}}(\mathbf{v})$ for a hyperbolic and for an additive reduction, respectively, with $d = 2$ and $s = 3$.

A final issue worth mentioning is whether one should prefer a hyperbolic or an additive reduction from the point of view of their respective local error behaviors. There is no general answer to this question. In Figure 5.4, we plot the observed errors due to index



quadrature				index set reduction			
$K \downarrow$	$d=2$	$d=3$	$d=4$	$K \downarrow$	$d=2$	$d=3$	$d=4$
15	1.013e-08	7.160e-09	5.062e-09	15	1.527e-08	1.654e-08	1.845e-08
30	6.061e-10	4.284e-10	3.029e-10	30	7.465e-10	6.469e-10	5.663e-10
45	1.170e-10	8.271e-11	5.847e-11	45	1.351e-10	1.088e-10	8.854e-11

Figure 5.2: Errors $\mathbf{e}^{\mathrm{quad}}(\mathbf{v})$ (black) and $\mathbf{e}^{\mathrm{red}}(\mathbf{v})$ (gray) for an additive reduction and a vector decaying according to (5.2) ($S = 16,\ R = 8,\ s = 5$). The symbols are the same as in Figure 5.1. Semi-logarithmic plot.

set reduction vs. the respective computation times for both kinds of reductions and both versions of the fast algorithm and $d=2,3,4$ using the data from Figures 5.1 and 5.2. As for the first version, the hyperbolic reduction is seen to be always superior over the additive one. As for the second version, the hyperbolic reduction is still preferable for all choices of K in case $d=2$, but the higher the dimension the larger a choice of K it takes for the hyperbolic reduction to outperform its additive counterpart. These observations are not directly related to the respective approximation qualities of both index set reductions, but are merely due to the fact that the hyperbolically reduced index sets comes with indices that contribute at most a single negative power of K to the local error bound, while the additive reduction can give as more.

5.2 Matrix exponentials

The matrices $\mathbf{D} + \mathbf{W}^{\mathrm{quad}}$ and $\mathbf{D} + W^{\mathrm{pol}}(\mathbf{X})$ both are Hermitian, and the respective matrix exponentials times a unit vector $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$ can be approximated by a Galerkin ansatz,

$$\exp(-ih(\mathbf{D} + \mathbf{W}^{\mathrm{quad}}))\mathbf{v} \approx \mathbf{V}_m^{\mathrm{quad}} \exp(-ih\mathbf{T}_m^{\mathrm{quad}})e_1,$$
$$\exp(-ih(\mathbf{D} + W^{\mathrm{pol}}(\mathbf{X})))\mathbf{v} \approx \mathbf{V}_m^{\mathrm{fast}} \exp(-ih\mathbf{T}_m^{\mathrm{fast}})e_1,$$

where the matrices $\mathbf{V}_m^{\mathrm{quad}}$, $\mathbf{T}_m^{\mathrm{quad}}$, $\mathbf{V}_m^{\mathrm{fast}}$, and $\mathbf{T}_m^{\mathrm{fast}}$ are obtained by an m-step Hermitian Lanczos process starting from $\mathbf{v}_1^{\mathrm{quad}} = \mathbf{v}_1^{\mathrm{fast}} = \mathbf{v}$; see Section 1.5 for the details. The real

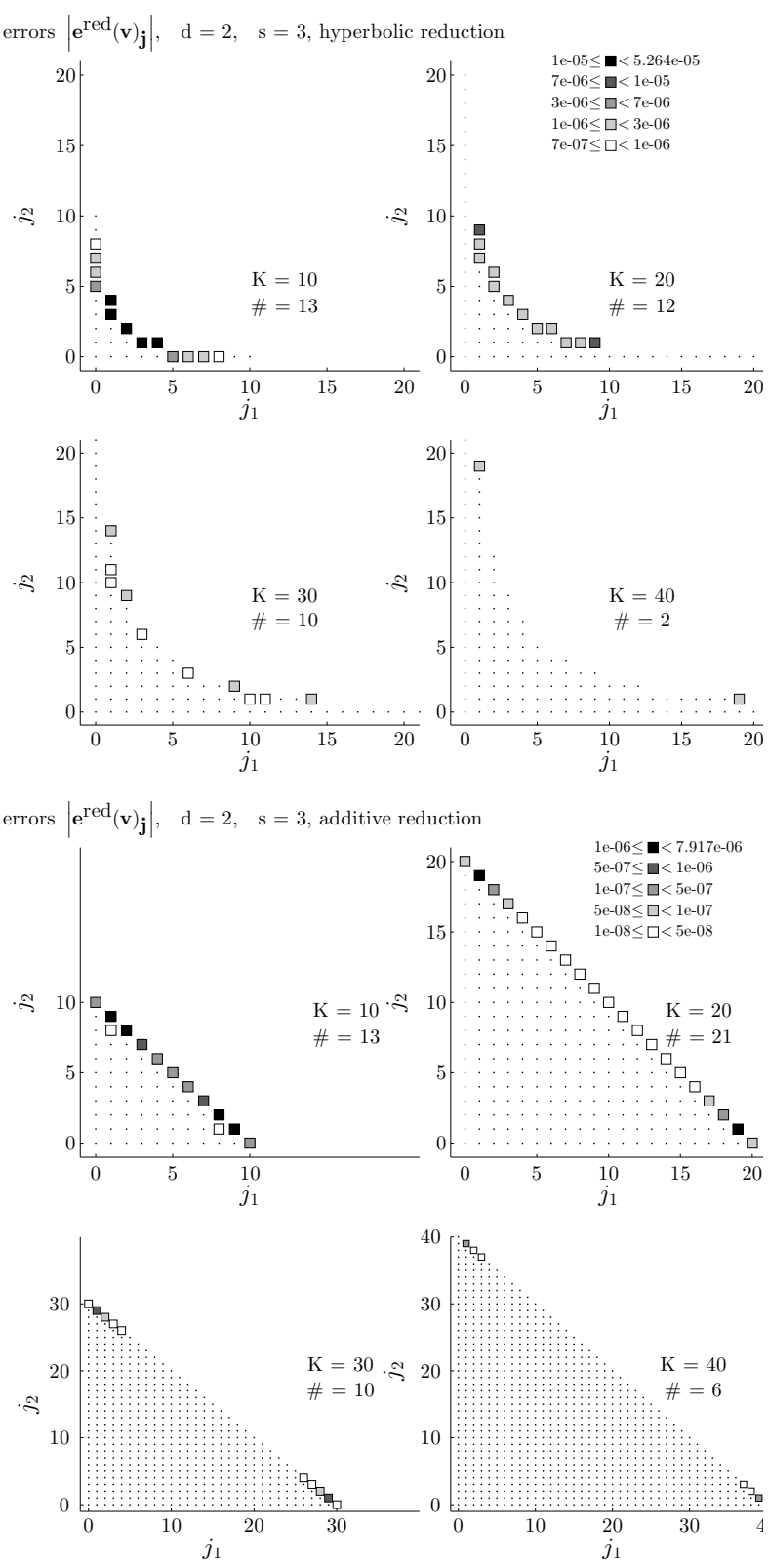


Figure 5.3: [12] $\mathbf{e}^{\mathrm{red}}(\mathbf{v})_{\mathbf{j}}$ due to index set reduction for the torsional potential (5.1) and a vector decaying according to (5.2) with $d=2$ and different choices of K (as above, $S = 16$, $R = 8$, $s = 3$). Each dot or box represents an error vector component. Upper figure: hyperbolic reduction. Lower figure: additive reduction. Errors being small with respect to the largest observed error component $e_{\max} \approx$ 5.264e-05 (hyperbolic) and $e_{\max} \approx$ 7.917e-06 (additive), respectively, are simply indicated by a dot; indices carrying larger errors are indicated by a gray box. The darker the box, the closer the error to $e_{\max}$. In the upper figure, the pictures corresponding to $K = 30, 40$ show an enlarged view. The symbol # points to the number of large error components. The errors decrease with growing K as indicated by a decreasing number of increasingly light boxes. As for the hyperbolic reduction, the error is concentrated in the region with only "intermediate" index components. As explained in the text, the additive reduction exhibits a somewhat complementary behavior.

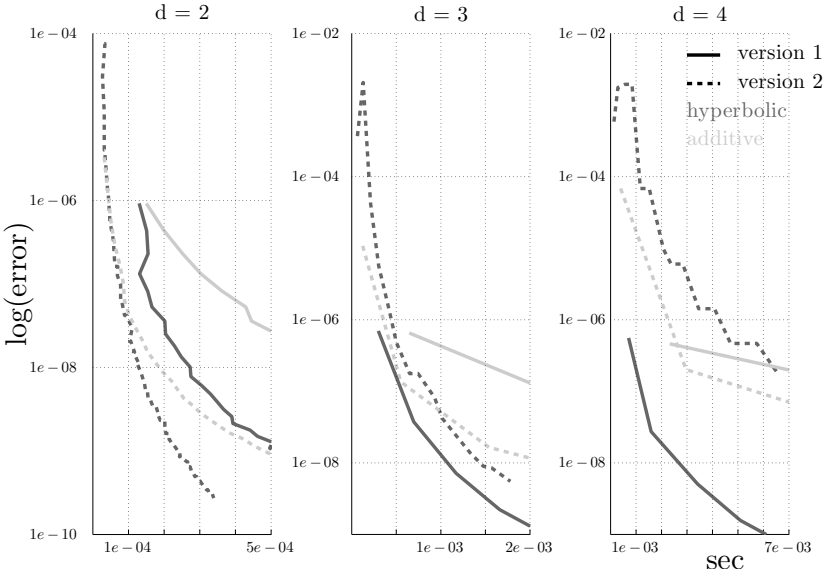


Figure 5.4: Computation times vs. errors for the hyperbolic (dark gray) and for the additive reduction (light gray). The two versions of the fast algorithm are represented by a solid line and by a dashed line, respectively. Semi-logarithmic plot.

quantity h is the time step size, and $\mathbf{v}$ is assumed to decay according to (5.2) for some given s. In this section, we illustrate the effect of replacing $\mathbf{W}^{\text{quad}}$ by $W^{\text{pol}}(\mathbf{X})$ when computing the above matrix exponentials. It is only in case of a full index cube that we have $\mathbf{W}^{\text{quad}} = W^{\text{pol}}(\mathbf{X})$, and the two matrix exponentials coincide; cf. Lemma 2 from Section 2.6. In case of a hyperbolic reduction, in particular, this is false. As it turns out, even a fairly small number of Lanczos iterations makes the error due to index set reduction dominate the error due to Lanczos itself.

We set

$$\mathbf{A}^{\text{quad}} = \mathbf{D} + \mathbf{W}^{\text{quad}}, \qquad \mathbf{A}^{\text{fast}} = \mathbf{D} + W^{\text{pol}}(\mathbf{X})$$

and decompose the error for the matrix exponentials according to

$$\begin{aligned}\big(\exp(-ih\mathbf{A}^{\text{quad}}) - \exp(-ih\mathbf{A}^{\text{fast}})\big)\mathbf{v} = \\ &\big(\exp(-ih\mathbf{A}^{\text{quad}})\mathbf{v} - \mathbf{V}_m^{\text{quad}} \exp(-ih\mathbf{T}_m^{\text{quad}})e_1\big) && (5.3)\\ &+ \big(\mathbf{V}_m^{\text{fast}} \exp(-ih\mathbf{T}_m^{\text{fast}})e_1 - \exp(-ih\mathbf{A}^{\text{fast}})\mathbf{v}\big) && (5.4)\\ &+ \big(\mathbf{V}_m^{\text{quad}} \exp(-ih\mathbf{T}_m^{\text{quad}})e_1 - \mathbf{V}_m^{\text{fast}} \exp(-ih\mathbf{T}_m^{\text{fast}})e_1\big). && (5.5)\end{aligned}$$

The errors (5.3) and (5.4) are the errors due to Lanczos approximation of the respective matrix exponentials. The error (5.5) is the error due to a perturbation of Lanczos by replacing $\mathbf{W}^{\text{quad}}$ by $W^{\text{pol}}(\mathbf{X})$. Our aim is to compare (5.3) and (5.5).

Lemma 7. (perturbation error)
The error (5.5) *due to a perturbation of Lanczos is given by*

$$\begin{aligned}\big\|\mathbf{V}_m^{\text{quad}} \exp(-ih\mathbf{T}_m^{\text{quad}})e_1 - \mathbf{V}_m^{\text{fast}} \exp(-ih\mathbf{T}_m^{\text{fast}})e_1\big\| \\ \leq h\|\mathbf{F}\| C(\mathbf{A}^{\text{quad}}, \mathbf{F}, h),\end{aligned} \tag{5.6}$$

where C decreases for decreasing h or increasing K, and the matrix $\mathbf{F}$ satisfies $\|\mathbf{F}\| \to 0$ for $K \to \infty$.

Proof. We set

$$\begin{aligned}\mathbf{A}^{\text{fast}}\mathbf{v}_k^{\text{fast}} &= \mathbf{A}^{\text{quad}}\mathbf{v}_k^{\text{quad}} + \mathbf{A}^{\text{fast}}\big(\mathbf{v}_k^{\text{fast}} - \mathbf{v}_k^{\text{quad}}\big) + \big(\mathbf{A}^{\text{fast}} - \mathbf{A}^{\text{quad}}\big)\mathbf{v}_k^{\text{quad}} \\ &= \mathbf{A}^{\text{quad}}\mathbf{v}_k^{\text{quad}} + \tilde{\mathbf{f}}_k\end{aligned}$$

for all $1 \leq k \leq m$, and

$$\tilde{\mathbf{F}} = \big(\tilde{\mathbf{f}}_1 | \ldots | \tilde{\mathbf{f}}_m\big).$$

Thus, $\|\tilde{\mathbf{F}}\| \to 0$ for $K \to \infty$; cf. Theorem 6. Using the Lanczos identities

$$\begin{aligned}\mathbf{V}_m^{\text{quad}}\mathbf{T}_m^{\text{quad}}(\mathbf{V}_m^{\text{quad}})^* &= \mathbf{A}^{\text{quad}}\mathbf{V}_m^{\text{quad}}(\mathbf{V}_m^{\text{quad}})^* - \tau_{m+1,m}^{\text{quad}}\mathbf{v}_{m+1}^{\text{quad}}e_m^T(\mathbf{V}_m^{\text{quad}})^*, \\ \mathbf{V}_m^{\text{fast}}\mathbf{T}_m^{\text{fast}} &= \mathbf{A}^{\text{fast}}\mathbf{V}_m^{\text{fast}} - \tau_{m+1,m}^{\text{fast}}\mathbf{v}_{m+1}^{\text{fast}}e_m^T\end{aligned}$$

(see Section 1.5 and [61], Chapter III.2.2) together with the above relation

$$\mathbf{A}^{\text{quad}}\mathbf{V}_m^{\text{quad}}(\mathbf{V}_m^{\text{quad}})^* = \mathbf{A}^{\text{fast}}\mathbf{V}_m^{\text{fast}}(\mathbf{V}_m^{\text{quad}})^* + \tilde{\mathbf{F}}(\mathbf{V}_m^{\text{quad}})^*,$$

we find

$$\begin{aligned}\mathbf{V}_m^{\text{quad}}\mathbf{T}_m^{\text{quad}}(\mathbf{V}_m^{\text{quad}})^* &= \mathbf{A}^{\text{fast}}\mathbf{V}_m^{\text{fast}}(\mathbf{V}_m^{\text{quad}})^* + \tilde{\mathbf{F}}(\mathbf{V}_m^{\text{quad}})^* \\ &\quad - \tau_{m+1,m}^{\text{quad}}\mathbf{v}_{m+1}^{\text{quad}}e_m^T(\mathbf{V}_m^{\text{quad}})^* \\ &= \mathbf{V}_m^{\text{fast}}\mathbf{T}_m^{\text{fast}}(\mathbf{V}_m^{\text{quad}})^* + \tilde{\mathbf{F}}(\mathbf{V}_m^{\text{quad}})^* \\ &\quad + (\tau_{m+1,m}^{\text{fast}}\mathbf{v}_{m+1}^{\text{fast}} - \tau_{m+1,m}^{\text{quad}}\mathbf{v}_{m+1}^{\text{quad}})e_m^T(\mathbf{V}_m^{\text{quad}})^* \\ &= \mathbf{V}_m^{\text{fast}}\mathbf{T}_m^{\text{fast}}(\mathbf{V}_m^{\text{fast}})^* + \mathbf{V}_m^{\text{fast}}\mathbf{T}_m^{\text{fast}}\big((\mathbf{V}_m^{\text{quad}})^* - (\mathbf{V}_m^{\text{fast}})^*\big) + \tilde{\mathbf{F}}(\mathbf{V}_m^{\text{quad}})^* \\ &\quad + (\tau_{m+1,m}^{\text{fast}}\mathbf{v}_{m+1}^{\text{fast}} - \tau_{m+1,m}^{\text{quad}}\mathbf{v}_{m+1}^{\text{quad}})e_m^T(\mathbf{V}_m^{\text{quad}})^* \\ &= \mathbf{V}_m^{\text{fast}}\mathbf{T}_m^{\text{fast}}(\mathbf{V}_m^{\text{fast}})^* + \mathbf{F},\end{aligned}$$

where $\|\mathbf{F}\| \to 0$ for $K \to \infty$. This gets us

$$\begin{aligned}&\mathbf{V}_m^{\text{quad}}\exp(-ih\mathbf{T}_m^{\text{quad}})e_1 - \mathbf{V}_m^{\text{fast}}\exp(-ih\mathbf{T}_m^{\text{fast}})e_1 \\ &= \mathbf{V}_m^{\text{quad}}\exp(-ih\mathbf{T}_m^{\text{quad}})(\mathbf{V}_m^{\text{quad}})^*\underbrace{\mathbf{V}_m^{\text{quad}}e_1}_{=\mathbf{v}} - \mathbf{V}_m^{\text{fast}}\exp(-ih\mathbf{T}_m^{\text{fast}})(\mathbf{V}_m^{\text{fast}})^*\underbrace{\mathbf{V}_m^{\text{fast}}e_1}_{=\mathbf{v}} \\ &= \Big[\exp(-ih\mathbf{V}_m^{\text{quad}}\mathbf{T}_m^{\text{quad}}(\mathbf{V}_m^{\text{quad}})^*) - \exp(-ih(\mathbf{V}_m^{\text{quad}}\mathbf{T}_m^{\text{quad}}(\mathbf{V}_m^{\text{quad}})^* - \mathbf{F})\Big]\,\mathbf{v}.\end{aligned}$$

By the sensitivity analysis for the matrix exponential given in [89], we can then establish the desired result. □

Having established this theoretical estimate for how the Lanczos process is perturbed by replacing $\mathbf{W}^{\text{quad}}$ with $W^{\text{pol}}(\mathbf{X})$ in each step, we turn to experimental tests. We consider the matrix $\mathbf{D} + \mathbf{W}^{\text{quad}}$ where W is again the torsional potential,

$$W(x) = \sum_{\alpha=1}^{d}(1 - \cos(x_\alpha/S)) - \frac{1}{2}\sum_{\alpha=1}^{d}x_\alpha^2, \qquad x \in [-S,S]^d, S = 16,$$

$h\to$	1/10	1/20	1/40	1/80
$d=2, K=10$	5.266e-06	2.632e-06	1.316e-06	6.581e-07
$d=2, K=20$	7.946e-07	3.954e-07	1.976e-07	9.882e-08
$d=2, K=30$	3.040e-07	1.485e-07	7.411e-08	3.705e-08
$d=2, K=40$	1.636e-07	7.689e-08	3.823e-08	1.911e-08
$d=3, K=40$	6.598e-07	3.253e-07	1.625e-07	8.125e-08
$d=4, K=40$	5.321e-06	2.662e-06	1.331e-06	6.655e-07

Table 5.1: [12] Perturbation error (5.6) depending linearly on h for fixed $m = 5$ and various choices of d and K (torsional potential, $S = 16$, $R=8$, $s=3$).

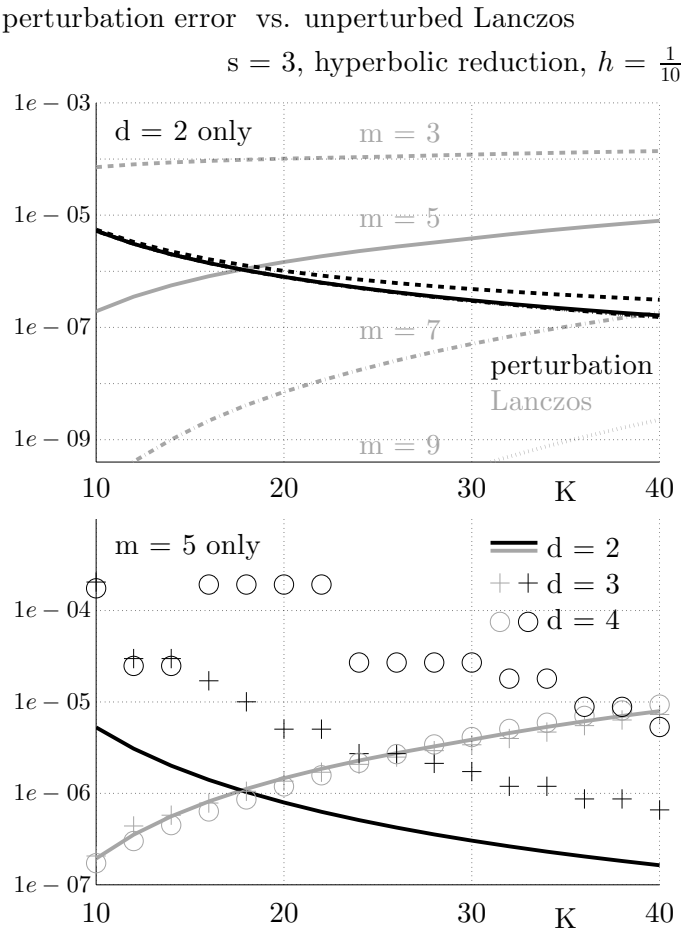
approximated by Chebyshev interpolation with $R=8$ as above. The underlying index set $\mathcal{K}$ is hyperbolically reduced. The linear decay of the error (5.6) with respect to h is shown in Table 5.1. Additionally, for a fixed choice of h, the error is seen to become worse for constant K and increasing d (as predicted by Theorem 6, where the constant in the error estimate depends on the dimension), and it becomes arbitrarily small for constant d and a sufficiently large choice of K (because $\|\mathbf{F}\|$ becomes smaller).

Next, we compare the perturbation error (5.6) to the unperturbed Lanczos error (5.3),

$$\begin{aligned}\left\| \exp(-ih(\mathbf{D}+\mathbf{W}^{\mathrm{quad}}))\mathbf{v} - \mathbf{V}_m^{\mathrm{quad}} \exp(-ih\mathbf{T}_m^{\mathrm{quad}})e_1 \right\| \\ \leq 8\left(\exp(1-(\omega/2m)^2)\frac{\omega}{2m}\right)^m,\end{aligned} \tag{5.7}$$

where $\omega = h(b-a)/2$ and $[a,b]$ is the interval that contains all the eigenvalues of the Hermitian matrix $\mathbf{D}+\mathbf{W}^{\mathrm{quad}}$; see, e.g., [61], Theorem III.2.10. The analogous error (5.4) yields almost the same figures, so we restrict our attention to a comparison of (5.6) to (5.7). We illustrate the error behavior for various choices of d, K, and m and for a fixed choice of h in Figure 5.5. The errors (5.6) and (5.7) are printed in black and gray color, respectively. In the upper part, we fix $d=2$ and vary m. In the lower part, we fix $m=5$ and vary d. We first consider the upper part. As the figures reveal, there is an antagonism: On the one hand, increasing m improves on (5.7), while increasing K worsens it due to the eigenvalues of $\mathbf{D}+\mathbf{W}^{\mathrm{quad}}$ becoming larger. On the other hand, the error (5.6) is hardly affected by an increase in m at all, while a larger choice of K improves on it, as seen above. What we wish for is an optimal combination of m and K such that neither error contribution is strongly dominant (for fixed choices of h and d). Presently, we lack further analytical insight into how to relate m, K, h, and d in an optimal way. For the choice $m=7$, the perturbation error dominates until K is about 40. Thus, for small or moderate choices of K, it does not take too large a choice of m for the perturbation error to become the dominant contribution, and a larger choice of m is not advisable, because it yields only additional costs and no additional benefits. We note that increasing m is always possible, though. The lower part shows the harmful effect on (5.6) of an increase in the dimension, while the error (5.7) is not affected, when $m=5$ (which happens to be a good choice for most of our examples) is fixed.

To obtain $\exp(-ih\mathbf{A}^{\mathrm{quad}})\mathbf{v}$, we have assembled $\mathbf{A}^{\mathrm{quad}}$ as explained in Section 3.1 and used the procedure `zheev` from the `LAPACK` `C`-library.



$K\rightarrow$	10	20	30	40
$d=2$	7.162e-05	1.015e-04	1.212e-04	1.386e-04
$m=3$	5.525e-06	1.015e-07	4.837e-07	3.116e-07
$d=2$	1.938e-07	1.464e-06	3.86e-06	7.935e-06
$m=5$	5.266e-06	7.946e-07	3.04e-07	1.636e-07
$d=2$	1.64e-10	7.103e-09	5.089e-08	1.848e-07
$m=7$	5.265e-06	7.906e-07	2.966e-07	1.532e-07
$d=2$	5.381e-14	1.712e-11	3.172e-10	2.271e-09
$m=9$	5.265e-06	7.906e-07	2.964e-07	1.529e-07
$d=3$	2.063e-07	1.359e-06	3.399e-06	7.288e-06
$m=5$	2.049e-04	5.04e-06	1.732e-06	6.598e-07
$d=4$	1.724e-07	1.192e-06	4.132e-06	9.352e-06
$m=5$	1.747e-04	1.922e-04	2.695e-05	5.321e-06

Figure 5.5: Errors (5.6) (black) and (5.7) (gray) as functions of K for fixed $h=1/10$ and various choices of d and m (torsional potential, $S=16$, $R=8$, $s=3$). Upper figure: $d=2$, $m=3,\ldots,9$ (dashed, solid, chain dotted, and dotted line, respectively). Lower figure: $d=2,3,4$, $m=5$ (plus signs, circles, and asterisks, respectively). In each cell of the table, the upper and lower figure corresponds to (5.7) and (5.6), respectively. Semi-logarithmic plot. Figure taken from [12] and modified.

5.3 Time integration

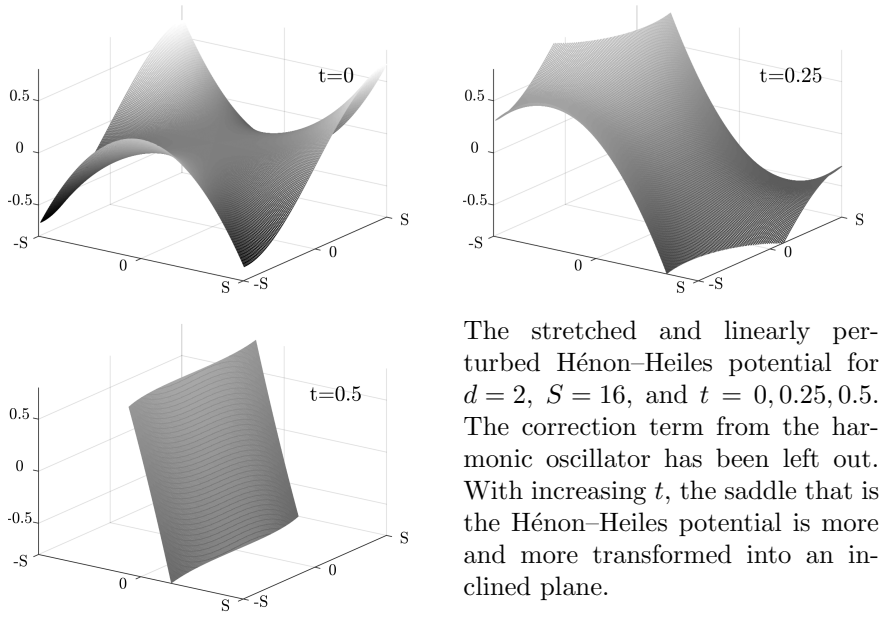
We conclude this chapter on numerical experiments with a propagation in time of the semidicrete system

$$i\dot{\mathbf{c}}^{\text{fast}}(t) = \mathbf{D}\mathbf{c}^{\text{fast}}(t) + W^{\text{pol}}(\mathbf{X}, t)\mathbf{c}^{\text{fast}}(t), \qquad t \in [0, 1]; \tag{5.8}$$

see (4.7). The underlying potential W is a stretched Hénon–Heiles potential with a linear time-dependent perturbation,

$$W(x,t) = \sum_{\alpha=1}^{d-1} \left[(x_\alpha/S)^2 (x_{\alpha+1}/S) - \frac{1}{3}(x_{\alpha+1}/S)^3 \right] - \sin^2(t)x_1 - \frac{1}{2}\sum_{\alpha=1}^{d} x_\alpha^2,$$

($x \in [-S, S]^d$, $S = 16$) exactly represented by fully-indexed Chebyshev interpolation with $R = 3$ nodes in every direction. This models the interaction of an atom or a molecule with a high-intensity CW laser in x_1-direction (see [71], with a quantum harmonic oscillator in place of a Hénon–Heiles potential). Some plots of the Hénon–Heiles potential without the harmonic oscillator correction term at different points in time are shown in Figure 5.6.



The stretched and linearly perturbed Hénon–Heiles potential for $d = 2$, $S = 16$, and $t = 0, 0.25, 0.5$. The correction term from the harmonic oscillator has been left out. With increasing t, the saddle that is the Hénon–Heiles potential is more and more transformed into an inclined plane.

Figure 5.6: $W(x) + \frac{1}{2}\sum_{\alpha=1}^{d} x_\alpha^2 = \sum_{\alpha=1}^{d-1} \left[(x_\alpha/S)^2 (x_{\alpha+1}/S) - \frac{1}{3}(x_{\alpha+1}/S)^3 \right] - \sin^2(t)x_1$

We want to demonstrate that the fast algorithm can be succesfully employed to (5.8) even in case the underlying Galerkin basis is drastically reduced, and therefore choose a hyperbolically reduced index set. When comparing the exact solution $\mathbf{c}(t)$ of (5.8) to the solution of the corresponding spatially continuous problem, our task is to come up with choices of parameters and initial data such that

- the truncation error does still not dominate the error due to a subsequent discretization in time (for fixed K and reasonable choices of time step size h),

- the error due to a perturbation of Lanczos does not dominate either (which, for fixed h, requires not too small a choice of K),
- the corresponding reference solutions are still computable with no excessive memory or time consumption (which requires not too large a choice both of h and K),
- and the resulting dynamics is still interesting enough. In particular, for the fast algorithm to be applicable, the solution needs to be significantly less smooth than the evolving potential surface.

Our aim is to show numerically the expected order of convergence with respect to h and to illustrate the error behavior if m, K, and d vary individually. We have observed the following choices to do us a good service in case $d=2,3,4$, and also to yield a good illustration of the difficulties involved.

As an initialization, we choose

$$\mathbf{c}^{\mathrm{fast}}(0) = \tilde{\mathbf{c}}(0)/\left\|\tilde{\mathbf{c}}(0)\right\|, \qquad \tilde{c}_{\mathbf{k}} = \begin{cases} 1, & \mathbf{k}=\mathbf{0} \text{ or } \mathbf{k}=(0,\dots,0,1,1), \\ 0, & \text{else.} \end{cases}$$

We start with a visualization of the exact solution $\psi(\cdot,t)$ to the corresponding spatially continuous problem in case $d=2$. For this purpose, we consider (5.8) based on the full index cube with $K=50$, and propagate this system in time using the scheme (1.33) with time step size $h=$1e-03 and $m=20$ Lanczos steps in each time step. Using a full index set, we can safely employ the fast algorithm as an equivalent, but computationally much cheaper alternative to quadrature. With as large a choice of K as this, we can expect the error due to quadrature to be negligible. Additionally, these careful choices of h and m yields a negligibly small time discretization error. We then transform the obtained coefficients to a grid of 100 equidistant points in $[-5,5]^2$ and plot the squared modulus $|\psi(\cdot,t)|^2$ evaluated at these points. This is shown in Figure 5.7 for different choices of t. As one can see, the function actually has support within $[-5,5]^2$. From the values at these grid points, we compute approximations to the L^2-norm of $|\psi(\cdot,t)|^2$. Transforming coefficients to this spatial grid constitutes a means to compare different choices of underlying index sets.

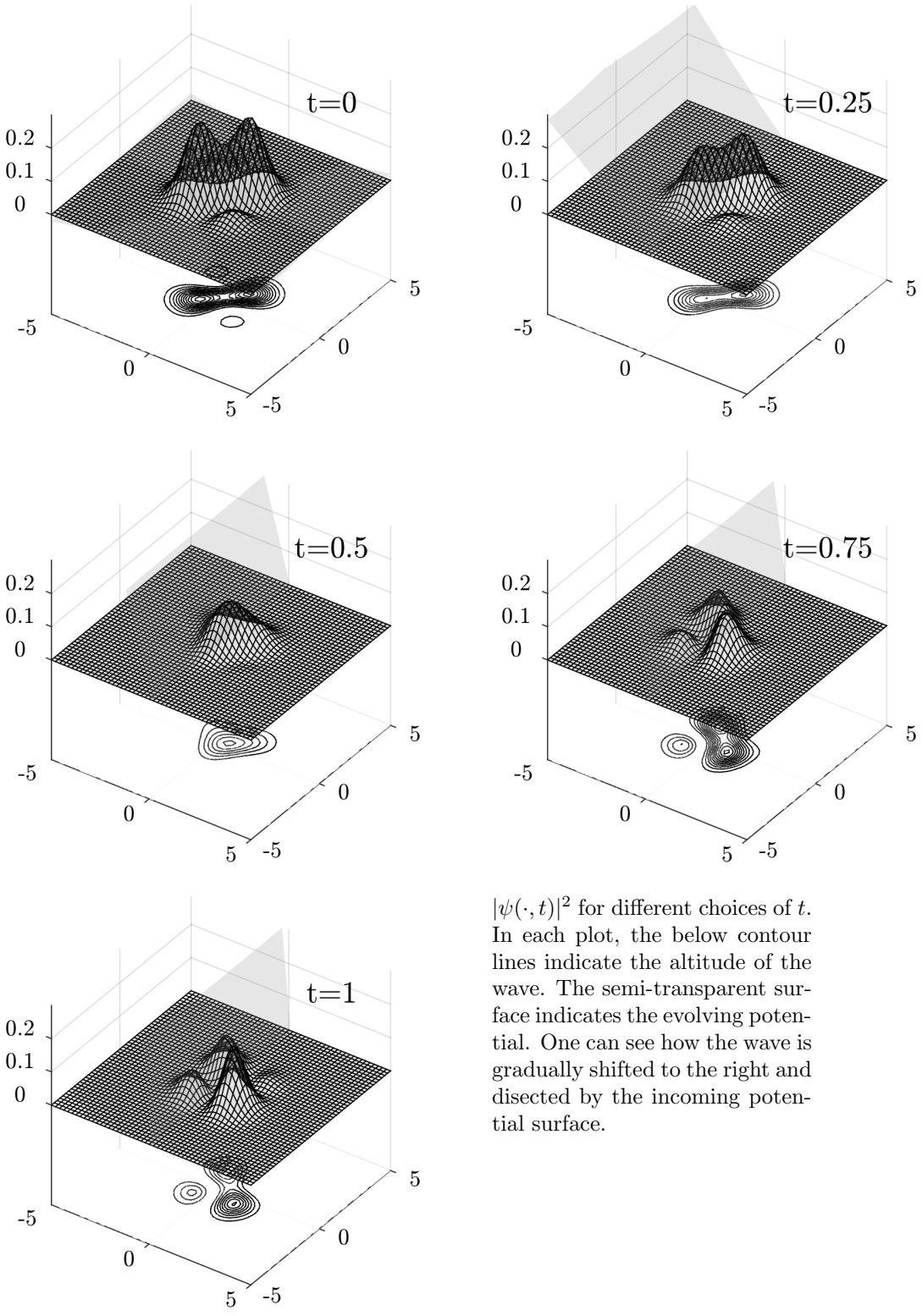
Before considering time discretization errors, we need to examine how sound an idea using a hyperbolically reduced grid in order to approximate ψ is in the first place. For this purpose, we consider the corresponding unperturbed system

$$i\dot{\mathbf{c}}^{\mathrm{quad}}(t) = \mathbf{D}\mathbf{c}^{\mathrm{quad}}(t) + \mathbf{W}^{\mathrm{quad}}(t)\mathbf{c}^{\mathrm{quad}}(t), \tag{5.9}$$

with the same initialization $\mathbf{c}^{\mathrm{quad}}(0) = \mathbf{c}^{\mathrm{fast}}(0)$; see (4.5). The underlying index set is the hyperbolically reduced index set with a varying threshold K. Time propagation is done using again the scheme (1.33) with $h=$1e-03 and $m=20$. Transforming the obtained coefficients to the above spatial grids yields an approximate solution $\psi_{\mathcal{K}}^{\mathrm{quad}}$, which is almost exact in time, but not in space. As we use a sufficiently high order for the Gauß–Hermite quadrature involved, the error stems exclusively from the reduction of the underlying Galerkin basis. Table 5.2 gives the corresponding L^2-errors

$$\left\|\psi(\cdot,t) - \psi_{\mathcal{K}}^{\mathrm{quad}}(\cdot,t)\right\| \tag{5.10}$$

at time $t=1$ for $d=2,3$ and various choices of K. As can be seen, in 2D, we can expect the decay in time errors not to be dominated by these spatial truncation errors unless h



$|\psi(\cdot, t)|^2$ for different choices of t. In each plot, the below contour lines indicate the altitude of the wave. The semi-transparent surface indicates the evolving potential. One can see how the wave is gradually shifted to the right and disected by the incoming potential surface.

Figure 5.7: Exact solution to the linearly perturbed Hénon–Heiles problem.

is considerably small. However, it is already the choice $d=3$ that does not as easily allow for a small choice of h as in 2D, while for higher choices of d, the approximation quality of the hyperbolic reduction is doubtful for the problem under consideration. We come back to this issue in the subsequent experiments.

$K\to$	10	15	20	25
$d=2$	1.388e-03	2.697e-05	2.113e-06	3.825e-07
$K\to$	10	20	30	40
$d=3$	1.280e-01	1.384e-02	1.779e-03	1.604e-04

Table 5.2: Error $\|\psi(\cdot,t) - \psi_{\mathcal{K}}^{\mathrm{quad}}(\cdot,t)\|$ due to spatial discretization at time $t=1$ for various choices of d and K.

Next, we eventually propagate (5.8) in time using the scheme (1.32) of order 2 with step sizes between $\frac{1}{10}$ and $\frac{1}{160}$ in case $d=2$. The Galerkin basis is again hyperbolically reduced. As a reference, we propagate the unperturbed system (5.9) using the scheme (1.33) with $h=$ 1e-03 and $m=20$, and $(K+1)$-nodes Gauß–Hermite quadrature. The latter is a safe choice, since, when applying the fast algorithm, the error due to quadrature is dominated by the error due to index set reduction; see the error analysis given in Sections 4.6 and 4.7. The respective approximations after n time steps are denoted by

$$\big(\mathbf{c}^{\mathrm{fast}}\big)^{(n)} \approx \mathbf{c}^{\mathrm{fast}}(t^n), \qquad \big(\mathbf{c}^{\mathrm{quad}}\big)^{(n)} \approx \mathbf{c}^{\mathrm{quad}}(t^n), \qquad t^n = nh, n = 0,1,\dots$$

The left picture in Figure 5.8 shows the error

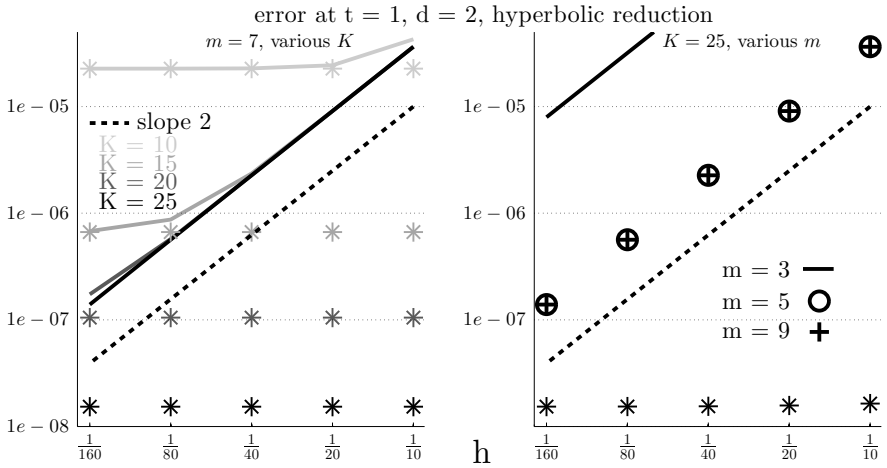
$$\Big\|\big(\mathbf{c}^{\mathrm{fast}}\big)^{(h^{-1})} - \big(\mathbf{c}^{\mathrm{quad}}\big)^{(1\mathrm{e}+03)}\Big\| \tag{5.11}$$

at time $t=1$. The choice of K varies between 10 (lightest gray) and 25 (black). In each time step, we employ the Lanczos process and $m=7$ Lanczos steps together with the fast algorithm. Additionally, we use asterisks to indicate the difference in the dynamics, viz.,

$$\Big\|\big(\mathbf{c}^{\mathrm{fast}}\big)^{(h^{-1})} - \big(\mathbf{c}^{\mathrm{quad}}\big)^{(h^{-1})}\Big\|, \tag{5.12}$$

where (5.9) has been propagated using the same integrator with exactly the same time step size. The latter error is the error due to a perturbation of Lanczos by replacing $\mathbf{W}^{\mathrm{quad}}$ by $W^{\mathrm{pol}}(\mathbf{X})$. This error contribution has been studied in Section 5.2. There are three noticeable observations: First, the perturbation error (5.12) becomes eventually dominant for decreasing h, as predicted in the last section, but it decreases arbitrarily for increasing K and fixed h. Second, unless (5.12) dominates, the figures reveal the expected order of convergence as visualized by the dashed line, which represents the slope. In the corresponding table, the columns "ratio" give the quotients of two consecutive errors. Third, in case $K=25$ and $h=\frac{1}{160}$, the error (5.11) is of similar size as the truncation error given in Table 5.2. For all other combinations of K and h, the truncation error is easily dominated by the error (5.11), and the hyperbolic reduction reveals to be a proper choice.

Let us examine a bit closer the role of the perturbation of Lanczos. The right picture shows the same error as on the left in case of fixed $K=25$, and of various choices of m as indicated by a solid line, circles, and plus signs, respectively. We use again the scheme (1.32). The perturbation error (5.12) for the choice $m=3$ is marked by asterisks. Being of almost exactly the same size, the perturbations for the other choices of m have been left



$m=7$	$K\rightarrow$	10		15		20		25	
	$h\downarrow$	(5.11)	ratio	(5.11)	ratio	(5.11)	ratio	(5.11)	ratio
	1/10	4.278e-05	–	3.638e-05	–	3.637e-05	–	3.637e-05	–
	1/20	2.439e-05	1.75	9.098e-06	4.00	9.074e-06	4.01	9.074e-06	4.01
	1/40	2.278e-05	1.07	2.360e-06	3.85	2.267e-06	4.00	2.265e-06	4.01
	1/80	2.268e-05	1.00	8.715e-07	2.71	5.731e-07	3.96	5.636e-07	4.02
	1/160	2.267e-05	1.00	6.794e-07	1.28	1.734e-07	3.30	1.390e-07	4.06

$K=25$	$m\rightarrow$	3		5		9		(5.12), $m=3$
	$h\downarrow$	(5.11)	ratio	(5.11)	ratio	(5.11)	ratio	
	1/10	2.035e-03	–	3.640e-05	–	3.637e-05	–	1.640e-08
	1/20	5.087e-04	4.00	9.074e-06	4.11	9.074e-06	4.01	1.573e-08
	1/40	1.272e-04	4.00	2.265e-06	4.03	2.267e-06	4.01	1.547e-08
	1/80	3.180e-05	4.00	5.636e-07	4.03	5.636e-07	4.02	1.538e-08
	1/160	7.951e-06	4.00	1.390e-07	4.08	1.390e-07	4.06	1.536e-08

Figure 5.8: Error (5.11) at $t=1$ when propagating (5.8) using the scheme (1.32) of order 2. Left: $d=2$, $m=7$, various K. Right: $d=2$, $K=25$, various m. The perturbation error (5.12) is indicated by asterisks. Log–log plot.

out. As can be seen, all choices of m between 5 and 9 yield almost exactly the same figures, and the perturbation does not interfere with the expected time convergence for our choices of h.

Next, we consider fixed choices of $K=40$, $m=7$, and vary d. Some convergence results in cases $d=2,3,4$ are shown in the upper half of Table 5.3. When d is increased, there are two separate issues we need to address: First, as stated above, the hyperbolic reduction ceases to be a good approximation for our example, and the truncation error (5.10) dominates. This is a general point of criticism of the hyperbolic reduction that does not directly affect the fast algorithm itself. Second, the perturbation error (5.12) (not explicitly given in the table)

becomes dominant more easily, and we fail to observe the expected order of convergence before the contribution from (5.12) outweighs. The order of the method (1.32) is therefore not fully revealed.

	$d\rightarrow$	2		3		4	
	$h\downarrow$	(5.11)	ratio	(5.11)	ratio	(5.11)	ratio
hyp., $K=40$	1/10	3.637e-05	–	2.684e-05	–	2.713e-05	–
	1/20	9.074e-06	4.01	6.705e-06	4.00	8.022e-06	3.38
	1/40	2.265e-06	4.01	1.711e-06	3.92	4.810e-06	1.67
	1/80	5.634e-07	4.02	5.556e-07	3.08	4.552e-06	1.06
	1/160	1.381e-07	4.08	3.827e-07	1.45	4.540e-06	1.00
add., $K=20$	1/10	3.637e-05	–	2.684e-05	–	2.786e-05	–
	1/20	9.074e-06	4.01	6.695e-06	4.01	6.713e-06	4.15
	1/40	2.265e-06	4.01	1.671e-06	4.01	1.672e-06	4.02
	1/80	5.634e-07	4.02	4.157e-07	4.02	4.157e-07	4.02
	1/160	1.381e-07	4.08	1.019e-07	4.08	1.019e-07	4.08

Table 5.3: Error (5.11) at $t=1$ when propagating (5.8) using the scheme (1.32) of order 2. Upper half: Hyperbolic reduction, $m=7$, $K=40$, various d. Lower half: Additive reduction, $m=7$, $K=20$, various d.

Finally, in view of these findings, we turn to the additive reduction. The lower half of Table 5.3 shows the errors (5.11) for fixed choices of $K=20$, $m=7$, and varying d. Even with only half the threshold as in the hyperbolic case, the 2D errors are exactly reproduced, and the expected order of convergence still is observable in case $d=3,4$. As an additional comment that cannot be seen from the table, for all these choices of d, the truncation error (5.10) is of size $\sim$ 1e-14, and the observed time-error ratios have been similar to those given in Figure 5.4. For the example under consideration, at least in case $d \geq 4$, the additive reduction is thus preferable over the hyperbolic reduction.

To conclude with, we briefly summarize the two main lessons to draw from these experiments: First, given any fixed kind of reduction, the fast algorithm is seen to work well. In principle, a reduction as radical as the hyperbolic one is not an obstacle for a succesful application. Second, in all our experiments, the error induced by the fast algorithm due to a reduction of the index set is not larger than the truncation error itself. Thus, *if* a specific kind of index reduction is feasible with a specific example, the fast algorithm can be safely applied. To put it differently, we do not expect a situation where a specific reduction yields a reasonable approximation, but the fast algorithm does not allow for a succesful propagation in time. All our observations nicely reflect the theoretical predictions.

6 Further applications

We conclude the first part of this thesis with a brief discussion of crucial assumptions for the fast algorithm as given above to be applicable vs. non-essential choices that were either made to ease the presentation, or could easily be dropped without too much additional efforts to circumvent the minor issues that arise from dropping them. Furthermore, we present further applications of the fast algorithm.

Section 6.1 gives a concise list of those crucial assumptions. In Section 6.2, we consider the case of a Hermite basis together with a differential operator that does not allow for a simple replacement of spatial derivatives by a diagonal eigenvalue matrix. It is shown how the fast algorithm can be applied with differential operators other than the harmonic oscillator. In Section 6.3, we comment on the applicability of the fast algorithm with a set of moving wavepacket basis functions, which is a more natural choice in case the wave function can be expected to be localized. Finally, in Section 6.4, we show how to apply the fast algorithm to the Schrödinger equation with a cubic nonlinearity.

6.1 Essentials and non-essentials

When developing the fast algorithm in Chapter 2, not all choices we made were necessary ones. The fast algorithm actually comes closer to being a general tool to deal with the intricacies of spectral discretizations of high-dimensional PDEs than a narrowly-taylored procedure for the linear Schrödinger equation with a specific kind of potential only. A presentation of the wider range of applications of the fast algorithm shall be postponed until Part II of this thesis. In the present section, we point out to what are the essential ingredients for the above procedure to be applicable, and to where there is still some leeway for generalizations.

- As pointed out multiply, the fast algorithm is applicable in arbitrary dimensions and with any kind of index set

  $$\mathcal{K}(d, K) \subseteq \mathcal{K}_{\text{full}}(d, K),$$

  as long as $\mathcal{K}$ satisfies the weak closure property (1.15).

- Additionally, the index sets for the Galerkin basis and the one for the polynomial approximation of the potential, viz., $\mathcal{R}(d, R)$, must be related via

  $$|\mathcal{K}| \gg |\mathcal{R}|, \qquad K \gg R, \tag{6.1}$$

  which is implied by the requirement that the polynomial $W(\cdot, t)$ as defined in (1.14) be significantly smoother than the solution $\psi(\cdot, t)$ to the original problem (1.1) wherever

ψ does not essentially vanish, for all $t \geq 0$. The two assumptions given in (6.1) each play a specific role: The assumption $|\mathcal{K}| \gg |\mathcal{R}|$ guarantees essentially linear scaling of the procedure, while $K \gg R$ guarantees convergence of the approximation.

- The chosen basis for a polynomial representation of the potential W is arbitrary. In Section 1.4, we choose Chebyshev polynomials, and the fast algorithm in both its versions thus employs the 1D Chebyshev recurrence (2.9) for the factors in each term of the multivariate Chebyshev expansion of W^{pol}; see line **13** of Algorithm 4 and line **10** of Algorithm 5, respectively. The particular benefits of Chebyshev interpolation notwithstanding, for an orthogonal polynomial basis other than Chebyshev polynomials, these lines need to be adjusted accordingly. Even a monomial representation is viable, which reduces the computations to direct applications of $\mathbf{X}^{(\alpha)}$ on vectors only.
  The fast algorithm as given in Chapter 2 requires a multivariate polynomial with not too many terms of not too high a degree for the approximation of W. Non-polynomials might also be applicable as a representation basis. In Section 6.4, when discussing nonlinearities, we consider a Hermite function approximation (i.e, polynomials times squares of a weight function) to the nonlinearity, and give it a treatment similar to the one for the multiplicative potential. Besides, the present thesis does not investigate any further into the issue of using non-polynomial approximation, though.

- Amongst their generic features, the defining three-term recurrence (1.10) of the Hermite functions is the one we put into use the most prominently, besides them being mutually orthogonal with respect to the scalar product that underlies the Galerkin approximation and being bounded independently of their index. Actually, any set of tensor products of univariate orthogonal basis functions that come with a recurrence of a K-independent number of terms does the trick. Classical orthogonal polynomials (or their corresponding functions, i.e., the polynomials times the square of the corresponding weight function and possibly an appropriate normalization factor) are thus a natural candidate for a Galerkin basis that allows for an application of the fast algorithm. A zoo of these polynomials and functions, including the underlying general theory, is readily found in various textbooks; see the references given in the introduction to Part I, page 8. In Part II, e.g., we shall consider the choice of a Legendre basis in place of Hermite functions. All that needs to be changed is the application of the specific recurrence relation in the direct operation procedure for the action of $\mathbf{X}^{(\alpha)}$ on a vector; see Algorithm 3.

The above considerations are summed up in Table 6.1.

6.2 Derivatives

There is an obvious objection to considering orthogonal polynomials or functions other than Hermite functions as a Galerkin basis: In Section 1.1, our first move was to get rid of spatial derivatives as they occur in the Hamiltonian operator H by invoking the eigenfunction relation the chosen Galerkin basis provides. In our case, this trick was facilitated by the interplay of the harmonic oscillator and the Hermite functions. Will the fast algorithm still be applicable with basis functions other than Hermite function? Will it be applicable with the same Hermite basis, but with a differential operator that does not yields the harmonic

item	minimal requirement
Galerkin basis index set $\mathcal{K}(d,K)$	any subset of $\mathcal{K}_{\text{full}}(d,K)$ such that (1.15) holds
polynomial approximation index set $\mathcal{R}(d,R)$	any subset of $\mathcal{R}_{\text{full}}(d,R)$ such that $\lvert\mathcal{K}\rvert \gg \lvert\mathcal{R}\rvert$ and $K \gg R$
basis for $W^{\text{pol}} \approx W$	classical orthogonal polynomials, monomials
Galerkin basis	small recurrence, orthogonality, boundedness

Table 6.1: Essentials and non-essentials for the fast algorithm.

oscillator by suitably smuggling in terms? To both questions, the answer is positive. As for a change of Galerkin basis, see the above comments at the end of Section 6.1 and, primarily, Part II of this thesis. In this section, we consider only the second issue: We retain the Hermite basis, but study more general kinds of differential operators. Even though the right-hand side of our PDE might not allow to invoke the Hermite eigenfunction property (1.11) such that the task of evaluating spatial derivatives can easily be circumvented, the methodology of the fast algorithm can still be applied. We briefly sketch the proceeding. For second-order differential operators, a more systematic way of doing spatial derivatives including an error analysis shall be presented in Part II.

6.2.1 Differential operators

The follwing observations apply with any Hamiltonian operator of the form $H = D + W$ where W is a multiplicative potential as above and D is a sum of products, viz.,

$$D = \sum_{\mathbf{r}\in\mathcal{R}} a_{\mathbf{r}}(x)\partial_x^{\mathbf{r}} = \sum_{\mathbf{r}\in\mathcal{R}} a_{\mathbf{r}}(x) \prod_{\alpha=1}^{d} \frac{\partial^{r_\alpha}}{\partial x_\alpha^{r_\alpha}} \tag{6.2}$$

with real-valued, position-depedendent coefficient functions $a_{\mathbf{r}}(x)$, $x \in \mathbb{R}^d$, which we approximate by polynomials $a_{\mathbf{r}}^{\text{pol}}$. Note that $\lvert\mathcal{R}\rvert$ is the number of terms in the operator D, not the number of terms in a polynomial approximation. For the sake of simplicity, we restrict ourselves to the special case

$$D = \sum_{\alpha=1}^{d} a_\alpha(x)\frac{\partial}{\partial x_\alpha},$$

i.e., $\mathcal{R} = \{\mathbf{e}_\alpha\,;\, 1 \le \alpha \le d\}$. Even without the harmonic oscillator trick given in Sections 1.1 and 1.2, the following proceeding then easily allows for a treatment of, e.g., the Fokker–Planck equation. Again, we consider termwise evaluation of the matrix-vector product with the corresponding matrix representation of the derivative operator, viz.,

$$\mathbf{D}^{\text{pol}}\mathbf{v} = \sum_{\alpha=1}^{d} \big(\varphi_{\mathbf{j}}, a_\alpha^{\text{pol}}\frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}\big)_{\mathbf{j},\mathbf{k}\in\mathcal{K}}\mathbf{v} = \sum_{\alpha=1}^{d} \mathbf{D}^{(\alpha')}\mathbf{v}, \qquad \mathbf{v} \in \mathbb{C}^{\lvert\mathcal{K}\rvert}.$$

The Hermite functions come with a three-term recurrence for their spatial derivatives,

$$\varphi_k'(x) = -\sqrt{\frac{k+1}{2}}\varphi_{k+1}(x) + \sqrt{\frac{k}{2}}\varphi_{k-1}(x), \qquad k \geq 0,$$

where we set $\varphi_{-1} \equiv 0$. This enables us to write

$$\begin{aligned}\big(\mathbf{D}^{(\alpha')}\mathbf{v}\big)_{\mathbf{j}} &= \sum_{\mathbf{k}\in\mathcal{K}} \big(\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}} \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}\big) v_{\mathbf{k}} \\ &= \sum_{\mathbf{k}\in\mathcal{K}} (\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}+\mathbf{e}_\alpha})\big(-\sqrt{\frac{k_\alpha+1}{2}}v_{\mathbf{k}}\big) + \sum_{\mathbf{k}\in\mathcal{K}} (\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}-\mathbf{e}_\alpha})\big(\sqrt{\frac{k_\alpha}{2}}v_{\mathbf{k}}\big) \\ &= \big(\tilde{\mathbf{D}}^{(+\alpha)}\mathbf{v}^{(+)}\big)_{\mathbf{j}} + \big(\mathbf{D}^{(-\alpha)}\mathbf{v}^{(-)}\big)_{\mathbf{j}},\end{aligned} \tag{6.3}$$

where

$$\big(\tilde{\mathbf{D}}^{(+\alpha)}\big)_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}+\mathbf{e}_\alpha}), \qquad \big(\mathbf{D}^{(-\alpha)}\big)_{\mathbf{jk}} = \begin{cases}(\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}-\mathbf{e}_\alpha}), & k_\alpha > 0, \\ 0, & k_\alpha = 0,\end{cases}$$

and

$$v_{\mathbf{k}}^{(+\alpha)} = -\sqrt{\frac{k_\alpha+1}{2}}v_{\mathbf{k}}, \qquad v_{\mathbf{k}}^{(-\alpha)} = \sqrt{\frac{k_\alpha}{2}}v_{\mathbf{k}}.$$

We now explain how the latter matrix-vector products with $\tilde{\mathbf{D}}^{(+\alpha)}$ and $\mathbf{D}^{(-\alpha)}$ can be reduced to matrix-vector products with $\mathbf{D}^{(\alpha)}$,

$$\big(\mathbf{D}^{(\alpha)}\big)_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K},$$

which enables us to use the fast algorithm with the polynomially approximated coefficient function a_α^{pol} in place of a polynomial potential.

6.2.2 Shifting vectors

Consider the following simple observation in 1D: Given an arbitrary matrix

$$A = (A_{jk})_{0\leq j,k\leq K} \in \mathbb{C}^{(K+1)\times(K+1)},$$

we define

$$A_{jk}^{(+)} = \begin{cases}A_{j,k+1}, & 0 \leq k \leq K-1, \\ 0, & k = K+1,\end{cases} \qquad 0 \leq j \leq K,$$

$$A_{jk}^{(-)} = \begin{cases}A_{j,k-1}, & 1 \leq k \leq K, \\ 0, & k = 0,\end{cases} \qquad 0 \leq j \leq K,$$

i.e., the matrix is shifted to the left and to the right, respectively, and the free column is filled with zeros. Using lower and upper diagonal unit matrices

$$I_{jk}^{(\pm)} = \delta_{j,k\pm1}, \qquad 0 \leq j,k \leq K,$$

we readily find

$$A^{(\pm)} \cdot \mathbf{v} = A \cdot I^{(\pm)}\mathbf{v}$$

for the products of the shifted matrices with a vector $\mathbf{v} \in \mathbb{C}^{(K+1)}$. Thus, we can express these products as products of the original matrix with shifted vectors.

In higher dimensions, for each $1 \leq \alpha \leq d$, we set

$$\mathbf{I}_{\mathbf{jk}}^{(\pm\alpha)} = \delta_{\mathbf{j},\mathbf{k}\pm\mathbf{e}_\alpha}, \qquad \mathbf{j},\mathbf{k} \in \mathcal{K}.$$

The analogous shifted $(|\mathcal{K}| \times |\mathcal{K}|)$-matrix $\mathbf{A}$ reads

$$\mathbf{A}_{\mathbf{jk}}^{(\pm\alpha)} = \begin{cases} \mathbf{A}_{\mathbf{j},\mathbf{k}\pm\mathbf{e}_\alpha}, & \mathbf{k} \in \mathcal{K} \setminus \{\mathbf{k}\,;\, \mathbf{k}\pm\mathbf{e}_\alpha \notin \mathcal{K}\}, \\ 0, & \mathbf{k}\pm\mathbf{e}_\alpha \notin \mathcal{K}, \end{cases} \qquad \mathbf{j},\mathbf{k} \in \mathcal{K},$$

for all $1 \leq \alpha \leq d$. We then find

$$\begin{aligned} \left(\mathbf{A}\mathbf{I}^{(\pm\alpha)}\right)_{\mathbf{jk}} &= \sum_{\mathbf{m}\in\mathcal{K}} \mathbf{A}_{\mathbf{jm}}\mathbf{I}_{\mathbf{mk}}^{(\pm\alpha)} = \sum_{\mathbf{m}\in\mathcal{K}} \mathbf{A}_{\mathbf{jm}}\delta_{\mathbf{m},\mathbf{k}\pm\mathbf{e}_\alpha} \\ &= \begin{cases} \mathbf{A}_{\mathbf{j},\mathbf{k}\pm\mathbf{e}_\alpha}, & \mathbf{k}\pm\mathbf{e}_\alpha \in \mathcal{K} \\ 0, & \mathbf{k}\pm\mathbf{e}_\alpha \notin \mathcal{K}, \end{cases} \qquad \mathbf{j},\mathbf{k} \in \mathcal{K}. \\ &= \mathbf{A}_{\mathbf{jk}}^{(\pm\alpha)}, \end{aligned}$$

Again, products with shifted matrices can be turned into products of unshifted matrices with shifted vectors. If $\mathbf{A}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, a^{\mathrm{pol}}\varphi_{\mathbf{k}})$ with a multivariate polynomial a^{pol}, we can thus approximate the matrix-vector product with the shifted matrix according to

$$\mathbf{A}^{(\pm\alpha)}\mathbf{v} = \mathbf{A}\big(\mathbf{I}^{(\pm\alpha)}\mathbf{v}\big) \approx a^{\mathrm{pol}}(\mathbf{X})\big(\mathbf{I}^{(\pm\alpha)}\mathbf{v}\big)$$

using the fast algorithm for the formal insertion of the coordinate matrices into the polynomial a^{pol}.

6.2.3 Doing derivatives by shifts

Applying these ideas to (6.3), we first compute $\mathbf{v}^{(\pm\alpha)}$ from $\mathbf{v}$, then shift $\mathbf{v}^{(\pm\alpha)}$ to obtain $\mathbf{I}^{(\pm\alpha)}\mathbf{v}^{(\pm\alpha)}$, all in in $\mathcal{O}(|\mathcal{K}|)$, and then use the fast algorithm with $a_\alpha^{\mathrm{pol}}(\mathbf{X})$ and $\mathbf{I}^{(\pm\alpha)}\mathbf{v}^{(\pm\alpha)}$, which takes $\mathcal{O}(R_\alpha|\mathcal{K}|)$ operations if R_α is the number of terms in the polynomial expansion of a_α. This needs to be done for every choice of α. As a minor issue, the matrix $\tilde{\mathbf{D}}^{(+\alpha)}$ does not equal the matrix

$$\mathbf{D}_{\mathbf{jk}}^{(+\alpha)} = \begin{cases} (\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}+\mathbf{e}_\alpha}), & \mathbf{k}+\mathbf{e}_\alpha \in \mathcal{K}, \\ 0, & \mathbf{k}+\mathbf{e}_\alpha \notin \mathcal{K}, \end{cases}$$

and

$$\big(\tilde{\mathbf{D}}^{(+\alpha)}\mathbf{v}^{(+\alpha)}\big)_{\mathbf{j}} = \big(\mathbf{D}^{(+\alpha)}\mathbf{v}^{(+\alpha)}\big)_{\mathbf{j}} + \sum_{\mathbf{k}+\mathbf{e}_\alpha\notin\mathcal{K}} (\varphi_{\mathbf{j}}, a_\alpha^{\mathrm{pol}}\varphi_{\mathbf{k}+\mathbf{e}_\alpha})v_{\mathbf{k}}^{(+\alpha)}. \tag{6.4}$$

The sum on the right-hand side of (6.4) contains all the boundary elements in the index set $\mathcal{K}(d,K)$. It has $|\mathcal{K}(d-1,K)|$-many terms and can be dealt with using an adaptation of the sequential summations approach given in Section 3.2.

As an afternote, consider the idea of introducing shifted coordinate matrices to operate on the original (i.e., not shifted) vector only instead of using shifted vectors. The matrices

$$\mathbf{X}^{(\pm\alpha)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, x_\alpha \varphi_{\mathbf{k}\pm\mathbf{e}_\alpha}), \qquad \mathbf{U}^{(\pm\alpha)}_{\mathbf{jk}} = \sqrt{\omega_{\mathbf{j}}}\varphi_{\mathbf{k}\pm\mathbf{e}_\alpha}(\xi_{\mathbf{j}}),$$

however, do not yield a unitary diagonalization of $\mathbf{X}^{(\alpha)}$ analogous to (2.17), because

$$\left((\mathbf{U}^{(\pm\alpha)})^T\mathbf{U}^{(\pm\alpha)}\right)_{\mathbf{jk}} = \sum_{\mathbf{m}} \omega_{\mathbf{m}}\varphi_{\mathbf{j}\pm\mathbf{e}_\alpha}(\xi_{\mathbf{m}})\varphi_{\mathbf{k}\pm\mathbf{e}_\alpha}(\xi_{\mathbf{m}}),$$

which does neither equal the corresponding exact integral for all choices of $\mathbf{j}$ and $\mathbf{k}$ (even for a full-product Gaussian quadrature, due to excessive degrees), nor does it evaluate to $\delta_{\mathbf{jk}}$ as required.

6.3 Moving wavepackets

The fast algorithm has first been proposed in [23] in the context of a time-splitting procedure for the Schrödinger equation in the semiclassical regime. Instead of Hermite functions with a fixed localization around zero, the underlying basis consists of parameterized, moving wavepackets. Such a basis that adapts to the localization of the solution could also be used in the classical regime if the solution is known in advance to exhibit a strong localization. We shall briefly introduce the moving wavepacket basis, give a rough presentation of the time-splitting procedure together with a sketch of how the fast algorithm needs to be adapted, and comment on how (and to what extent) the above error analysis carries over.

6.3.1 Hagedorn wavepackets

The Hagedorn wavepackets are a multidimensional generalizsation of Hermite functions; see [41, 42, 43]. They allow for more flexibility with respect to their position and momentum localization. Starting from the parameterized complex Gaussian wavepacket

$$\varphi^\varepsilon_{\mathbf{0}}(x) = (\pi\varepsilon)^{-d/4}\det(Q)^{-1/2}\exp\left(\frac{i}{2\varepsilon}(x-q)^TPQ^{-1}(x-q) + \frac{i}{\varepsilon}p^T(x-q)\right), \qquad x\in\mathbb{R}^d,$$

centered in position $q\in\mathbb{R}^d$ and momentum $p\in\mathbb{R}^d$ with invertible matrices $Q,P\in\mathbb{C}^{d\times d}$ that satisfy a specific symplecticity condition, the Hagedorn wavepackets $\varphi^\varepsilon_{\mathbf{k}}$, $\mathbf{k}\in\mathbb{N}^d$, can be constructed recursively via

$$Q\left(\sqrt{k_\alpha+1}\varphi^\varepsilon_{\mathbf{k}+\mathbf{e}_\alpha}(x)\right)_{\alpha=1}^d = \sqrt{\frac{2}{\varepsilon}}(x-q)\varphi^\varepsilon_{\mathbf{k}}(x) - \bar{Q}\left(\sqrt{k_\alpha}\varphi^\varepsilon_{\mathbf{k}-\mathbf{e}_\alpha}(x)\right)_{\alpha=1}^d. \tag{6.5}$$

Therefore, $\varphi^\varepsilon_{\mathbf{k}} = \varphi^\varepsilon_{\mathbf{k}}[q,p,Q,P]$ is the product of a polynomial of degree $\sum_{\alpha=1}^d k_\alpha$ times a complex Gaussian, viz.,

$$\varphi^\varepsilon_{\mathbf{k}}(x) = \left(2^{\sum_\alpha k_a}\prod_\alpha k_\alpha!\right)^{-1/2} p^\varepsilon_{\mathbf{k}}(x)\varphi^\varepsilon_{\mathbf{0}}(x);$$

see [58], Prop. 2, for the latter identity. If Q is real and symmetric, the polynomials $p^\varepsilon_{\mathbf{k}}$ turn out to be products of suitably evaluated Hermite polynomials. The scale parameter $\varepsilon > 0$ determines the width of a wavepacket to be of size $\mathcal{O}(\varepsilon^{-1/2})$ and its wavelength to be of size $\mathcal{O}(\varepsilon)$. The family $\{\varphi^\varepsilon_{\mathbf{k}}(x)\}_{\mathbf{k}\in\mathbb{N}^d}$ forms an $L^2(\mathbb{R}^d)$-orthonormal basis of highly oscillatory functions.

6.3.2 Semiclassical splitting and the fast algorithm

In case of a quadratic (possibly time-dependent) potential V, the Hagedorn wavepackets multiplied with a phase factor $\exp\big(\frac{i}{\varepsilon}S(t)\big)$ turn out to solve the semiclassical Schrödinger equation

$$\begin{aligned} i\varepsilon\psi_t(x,t) &= (H^\varepsilon\psi)(x,t) \\ &= -\frac{\varepsilon^2}{2}(\Delta\psi)(x,t) + V(x,t)\psi(x,t), \end{aligned} \qquad x = (x_1,\dots,x_d)\in\mathbb{R}^d, \quad 0<\varepsilon\ll 1, \tag{6.6}$$

exactly if the parameters $q(t), p(t), Q(t), P(t)$ follow a set of classical and linearized equations of motions, respectively, and if $S(t)$ is the classical action; see [43], or (in our notation) [61], Chapter V.

In [23], this is used to devise a fully discrete, explicit, and time-reversible time-stepping algorithm to propagate the parameters, the phase, and the coefficients $c_{\mathbf{k}}$ in the ansatz

$$\psi(x,t) \approx \psi_{\mathcal{K}}(x,t) = \exp\big(\frac{i}{\varepsilon}S(t)\big)\sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}(t)\varphi^\varepsilon_{\mathbf{k}}[q(t),p(t),Q(t),P(t)](x),$$

where ψ is the solution to (6.6) and $\mathcal{K}(d,K)\subseteq\mathcal{K}_{\text{full}}(d,K)$ is a d-dimensional index set. The idea is to split the Hamiltonien operator H^ε into its kinetic part and its potential part, and to further subdivide V into a local quadratic approximation around the current position and the non-quadratic remainder W. The free Schrödinger equation and the quadratic potential part can then each be solved exactly with unaltered coefficients $c_{\mathbf{k}}$, and the non-quadratic potential equation is approximated using a Galerkin ansatz on the linear space spanned by the functions $\varphi^\varepsilon_{\mathbf{k}}$ with fixed parameters q,p,Q,P and varying coefficients $c_{\mathbf{k}}$. The kinetic and potential parts are trivially propagated with negligible costs, and the non-quadratic remainder part requires to compute the action of the matrix exponential $\exp\big(-\frac{ih}{\varepsilon}\mathbf{W}\big)$ on a vector $\mathbf{v}\in\mathbb{C}^{|\mathcal{K}|}$ in each time step. This can be accomplished using a few Lanczos steps in combination with our fast algorithm.

As already mentioned, using the fast algorithm for the above task has actually been proposed in [23] without further details. Let us briefly explicate this: The overall computational procedure for the approximation

$$\mathbf{W}\mathbf{v} \approx W^{\text{pol}}(\mathbf{X}^{(\alpha;\varepsilon)})\mathbf{v}, \tag{6.7}$$

where W^{pol} is a polynomial approximate of the non-quadratic remainder of the potential, remains almost unaltered when replacing tensor products of Hermite functions by Hagedorn wavepackets. By virtue of the recurrence (6.5), a direct operation with the ε-dependent coordinate matrices

$$\mathbf{X}^{(\alpha;\varepsilon)}_{\mathbf{jk}} = (\varphi^\varepsilon_{\mathbf{j}}, x_\alpha\varphi^\varepsilon_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k}\in\mathcal{K}, \quad \alpha = 1,\dots,d,$$

can be performed via

$$\Big((\mathbf{X}^{(\alpha;\varepsilon)}\mathbf{v})_{\mathbf{j}}\Big)_{\alpha=1}^{d} = v_{\mathbf{j}}q + \sqrt{\frac{\varepsilon}{2}}Q\left(\sqrt{j_\alpha}v_{\mathbf{j}-\mathbf{e}_\alpha}\right)_{\alpha=1}^{d} + \sqrt{\frac{\varepsilon}{2}}\bar{Q}\left(\sqrt{j_\alpha+1}v_{\mathbf{j}+\mathbf{e}_\alpha}\right)_{\alpha=1}^{d}; \qquad (6.8)$$

cf. Algorithm 3. Note that, due to the presence of the matrices Q and $\bar{Q}$, this recurrence formula no longer consists of only a single row, but of d rows. The complexity of $\mathbf{X}^{(\alpha;\varepsilon)}\mathbf{v}$ is thus $\mathcal{O}(d|\mathcal{K}|)$. Apart from the above direct operation procedure (6.8), the rest of the fast algorithm is exactly the same as before. And so is the complexity, but for an additional factor of d.

6.3.3 Error

An error estimate for a splitting scheme even more elaborate than the above has been given in [36]. However, the error due to a numerical solution of the non-quadratic remainder equation has not been discussed. In particular, there is no analysis for an approximation of the matrix-vector product $\mathbf{W}\mathbf{v}$ (polynomial interpolation, quadrature, index set reduction). We restrict our attention to the local error (6.7) that arises from computing the matrix-vector product $W^{\mathrm{pol}}(\mathbf{X}^{(\alpha;\varepsilon)})\mathbf{v}$ with the help of the fast algorithm adapted to a Hagedorn wavepacket basis.

Following [23], we use the change of variables $x = q+\sqrt{\varepsilon}|Q|y$, where $|Q| = (QQ^*)^{1/2}$, to define the ε-independent functions $\phi_{\mathbf{k}}$ via

$$\phi_{\mathbf{0}}(y) = \pi^{-d/4}\exp\big(-\frac{1}{2}\sum_\alpha y_\alpha^2\big),$$
$$Q\left(\sqrt{k_\alpha+1}\phi_{\mathbf{k}+\mathbf{e}_\alpha}(y)\right)_{\alpha=1}^{d} = \sqrt{2}(y-q)\phi_{\mathbf{k}}(y) - \bar{Q}\left(\sqrt{k_\alpha}\phi_{\mathbf{k}-\mathbf{e}_\alpha}(y)\right)_{\alpha=1}^{d}.$$

Again, these functions are polynomials of degree $\sum_\alpha k_\alpha$ times the Gaussian $\phi_{\mathbf{0}}$. We aim for a formal relation between the fast algorithm and a suitable choice of quadrature. With the help of the ϕ-functions, the proceeding from Section 2.6 can be mimicked. We rewrite and approximate the entries of the matrix representation $\mathbf{W}^{\mathrm{pol}}$ according to

$$\begin{aligned}\mathbf{W}^{\mathrm{quad}}_{\mathbf{jk}} &= \int_{\mathbb{R}^d}\bar{\varphi}^{\varepsilon}_{\mathbf{j}}(x)W^{\mathrm{pol}}(x)\varphi^{\varepsilon}_{\mathbf{k}}(x)\,\mathrm{d}x = \int_{\mathbb{R}^d}\bar{\phi}_{\mathbf{j}}(y)W^{\mathrm{pol}}(q+\sqrt{\varepsilon}|Q|y)\phi_{\mathbf{k}}(y)\,\mathrm{d}y\\ &\approx \sum_{\mathbf{m}\in\mathcal{K}_{\mathrm{full}}}\omega_{\mathbf{m}}\bar{\phi}_{\mathbf{j}}(\xi_{\mathbf{m}})W^{\mathrm{pol}}(q+\sqrt{\varepsilon}|Q|\xi_{\mathbf{m}})\phi_{\mathbf{k}}(\xi_{\mathbf{m}}),\end{aligned}$$

where we use again $(K+1)$-nodes full product Gauß–Hermite quadrature. The ϕ-functions no longer yield a highly oscillatory integrand. Thus, we no longer require an excessive amount of quadrature nodes for an accurate computation of $\mathbf{W}^{\mathrm{pol}}$.

Let us first consider the full index cube. Using orthonormality of the Hagedorn wavepackets together with the exactness properties of Gaussian quadrature, we can define a parameterized counterpart to the above matrix $\mathbf{U}$ built from $\phi_{\mathbf{k}}$, viz.,

$$\mathbf{U}_{\mathbf{jk}} = \sqrt{\omega_{\mathbf{j}}}\phi_{\mathbf{k}}(\xi_{\mathbf{j}}), \qquad \mathbf{j},\mathbf{k}\in\mathcal{K},$$

(cf. (2.16)) and diagonalize the ε-dependent coordinate matrices according to

$$\mathbf{X}^{(\alpha;\varepsilon)} = \mathbf{U}^T\,\mathrm{diag}_{\mathbf{m}\in\mathcal{K}_{\mathrm{full}}}\big(q_\alpha+\sqrt{\varepsilon}(|Q|\xi_{\mathbf{m}})_\alpha\big)\mathbf{U}.$$

Again, this holds true only for the full index index cube. We see that the very same equivalence of formal insertion of coordinate matrices and Gaussian quadrature as already given in Lemma 2 for the Hermite basis is also valid for Hagedorn wavepackets, and the error due to the fast algorithm again turns out to be an error due to quadrature. Since the matrix $\mathbf{W}^{\mathrm{quad}}$ is Hermitian, we can mimic the proof of Lemma 4 in detail.

The case of a reduced index set is partly an unresolved issue, though. The problem is that Hagedorn wavepackets are multivariate functions that do not factorize into a product of univariate functions—unlike their tensorized Hermite counterparts. In the proof of Theorem 5, however, the error due to quadrature is reduced to the 1D case by means of this very factorization. We do not see how to transfer the binary tree technique, that crucially makes use of the 1D Hermite recurrence (1.10), from the Hermite to the Hagedorn case. A Hagedorn analog to Theorem 6 can be proved, though, since the underlying binary tree construction only invokes the 1D Chebyshev recurrence, but not (1.10). As one might imagine from the binary tree devised in Figure 4.3, mimicking the proceeding only works at the cost of highly complicated binary tree expressions.

6.4 Nonlinearities

We consider the dimensionless Schrödinger equation with a cubic nonlinearity,

$$\begin{aligned} i\dot{\psi}(x,t) = -\frac{1}{2}\big(\Delta\psi\big)(x,t) - \sum_{\alpha=1}^{d} x_\alpha^2\big)\psi(x,t) \\ + \big(W(x,t) + g|\psi(x,t)|^2\big)\psi(x,t), \end{aligned} \qquad x = (x_1,\ldots,x_d) \in \mathbb{R}^d, \tag{6.9}$$

for $t \geq 0$, where W is a real-valued multiplicative, possibly time-dependent potential and g is a constant. For the sake of simplicity, we set $g=1$. In case $W \equiv 0$, eq. (6.9) describes the wave function of a Bose–Einstein condensate in a harmonic trap. First, we discretize (6.9) in space using the same Hermite spectral approach as in Sections 1.1 and 1.2, and then propagate it in time using a Strang splitting ansatz: The harmonic oscillator part yields a trivially computable analytic solution. For the remainder part, however, time-stepping involves matrix-vector products we avoid to do explicitly by a modification of the above fast algorithm.

Let all assumptions on ψ and W from Sections 1.4 and Section 2.1 for the fast algorithm to be applicable hold true. As will be seen, the only further assumption on (6.9) we need concerns the regularity of $|\psi|^2$ as compared to ψ.

6.4.1 Spectral discretization in space

We use the same spectral ansatz as in Sections 1.1 and 1.2. In 1D, we write

$$\varphi_k(x) = f_k H_k(x)\exp(-x^2/2), \quad f_k = \left(2^k k!\pi^{1/2}\right)^{-1/2}, \qquad k \geq 0,$$

for the univariate Hermite functions, where H_k is again the k-th order Hermite polynomial. In higher dimension, we use a tensor product Hermite basis as in Section 1.2, where the underlying index set is the full index cube

$$\mathcal{K}(d,K) = \left\{\mathbf{k} \in \mathbb{N}^d \,|\, 0 \leq k_\alpha \leq K\right\}.$$

The unknown coefficients $\mathbf{c}(t) = (c_{\mathbf{k}}(t))_{\mathbf{k}\in\mathcal{K}}$ of the approximation

$$\psi(x,t) \approx \psi_{\mathcal{K}}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x) \in \operatorname{span}\{\varphi_{\mathbf{k}} \,|\, \mathbf{k}\in\mathcal{K}(d,K)\},$$

are determined via

$$\left(\varphi_{\mathbf{j}}, i\dot{\psi}_{\mathcal{K}} + \frac{1}{2}\Delta\psi_{\mathcal{K}} - \frac{1}{2}\sum_{\alpha=1}^{d} x_\alpha^2\psi_{\mathcal{K}} - \big(W + |\psi_{\mathcal{K}}|^2\big)\psi_{\mathcal{K}}\right) = 0 \qquad \forall \mathbf{j}\in\mathcal{K}. \tag{6.10}$$

Using the eigenfunction relation (1.13) for the Hermite functions together with the orthonormality of the basis, eq. (6.10) is seen to be equivalent to the system of ODEs

$$i\dot{\mathbf{c}}(t) = \mathbf{D}\mathbf{c}(t) + (\mathbf{W}(t) + \mathbf{P}(t))\mathbf{c}(t),$$

with the diagonal matrix $\mathbf{D} = \operatorname{diag}_{\mathbf{k}\in\mathcal{K}}\big(\sum_{\alpha=1}^{d}(k_\alpha + \frac{1}{2})\big)$ and $(|\mathcal{K}| \times |\mathcal{K}|)$-matrices

$$\mathbf{W}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, W(\cdot,t)\varphi_{\mathbf{k}}), \qquad \mathbf{P}_{\mathbf{jk}}(t) = (\varphi_{\mathbf{j}}, |\psi_{\mathcal{K}}(\cdot,t)|^2\varphi_{\mathbf{k}}), \quad \mathbf{j},\mathbf{k}\in\mathcal{K}. \tag{6.11}$$

6.4.2 Propagation in time

For an abstract evolution equation of the form

$$\dot{u}(t) = \big(A + B(u(t),t)\big)u(t), \qquad t\geq 0, \quad u(0) \text{ given},$$

where A is a linear and B is a nonlinear operator, a Strang or symmetric Lie-Trotter splitting method is built from the solutions of evolution equations

$$\begin{cases} \dot{v}(t) = Av(t), \\ v(0) = v_0 \text{ given}, \end{cases} \qquad \begin{cases} \dot{w}(t) = B(w(t),t), \\ w(0) = w_0 \text{ given}, \end{cases}$$

with exact solutions and evolution operators

$$v(t) = \Phi_A(t)[v_0], \quad w(t) = \Phi_B(t)[w_0], \qquad t\geq 0,$$

respectively. The Strang splitting scheme reads

$$u_{n+1} = \Phi_A(h/2)\big[\Phi_B(h)\big[\Phi_A(h/2)[u_n]\big]\big], \qquad n\geq 0,$$

and $u_0 = u(0)$. In our case, u is a placeholder for the coefficient vector $\mathbf{c}(t)$, and

$$A = -i\mathbf{D}, \qquad B(\mathbf{c}(t),t) = -i(\mathbf{W}(t)\mathbf{c}(t) + \mathbf{P}(t))\mathbf{c}(t),$$

are the Galerkin representations of the right-hand side operators as occurring in (6.9).

The linear evolution operator Φ_A trivially is given as

$$\Phi_A(t) = \operatorname{diag}_{\mathbf{k}\in\mathcal{K}}\left(\exp\left(-it\sum_{\alpha=1}^{d}(k_\alpha + \frac{1}{2})\right)\right).$$

As for the evolution Φ_B, we have to discretize in time the system of ODEs

$$i\dot{\mathbf{c}}(t) = (\mathbf{W}(t) + \mathbf{P}(t))\mathbf{c}(t). \tag{6.12}$$

This involves evaluations of $B(\mathbf{v},t)$, for some vectors $\mathbf{v}\in\mathbb{C}^{|\mathcal{K}|}$, which in turn necessitates efficient evaluation procedures for $\mathbf{W}(t)\mathbf{v}$ and $\mathbf{P}(t)\mathbf{v}$. The fast algorithm as devised in Chapter 2 applies to approximate $\mathbf{W}(t)\mathbf{v}$. We shall now devise a similar procedure for $\mathbf{P}(t)\mathbf{v}$.

6.4.3 Approximation of the squared modulus

We consider a truncated Hermite expansion of $|\psi_{\mathcal{K}}|^2$,

$$|\psi_{\mathcal{K}}(x,t)|^2 \approx \mathcal{P}_{\mathcal{L}}|\psi_{\mathcal{K}}(x,t)|^2 = \sum_{\mathbf{l}\in\mathcal{L}} d_{\mathbf{l}}(t)\varphi_{\mathbf{l}}(x), \qquad d_{\mathbf{l}}(t) = (\varphi_{\mathbf{l}}, |\psi_{\mathcal{K}}(\cdot,t)|^2),$$

where $\mathcal{P}_{\mathcal{L}}$ is the L^2-orthogonal projection onto span $\{\varphi_{\mathbf{l}} \,|\, \mathbf{l} \in \mathcal{L}(d,L)\}$ with a multi-index set $\mathcal{L} = \mathcal{L}(d,L) \subset \mathbb{N}^d$.

For all $t \geq 0$, by the definition of $d_{\mathbf{l}}(t)$ and by the fact that the Hermite functions are real-valued, we find

$$d_{\mathbf{l}}(t) = (\varphi_{\mathbf{l}}, |\psi_{\mathcal{K}}(\cdot,t)|^2) = \big(\varphi_{\mathbf{l}}, (\sum_{\mathbf{k}} \bar{c}_{\mathbf{k}}(t)\varphi_{\mathbf{k}})(\sum_{\mathbf{j}} c_{\mathbf{j}}(t)\varphi_{\mathbf{j}})\big) = \sum_{\mathbf{k}} \bar{c}_{\mathbf{k}}(t) \sum_{\mathbf{j}} (\varphi_{\mathbf{k}}, \varphi_{\mathbf{l}}\varphi_{\mathbf{j}}) c_{\mathbf{j}}(t),$$

or, equivalently,

$$d_{\mathbf{l}}(t) = \mathbf{c}^*(t)\mathbf{P}^{(\mathbf{l})}\mathbf{c}(t),$$

with triple-product matrices

$$\mathbf{P}^{(\mathbf{l})}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, \varphi_{\mathbf{l}}\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k} \in \mathcal{K}, \quad \mathbf{l} \in \mathcal{L}. \tag{6.13}$$

In particular, if $\mathbf{P}^{(\mathbf{l})}\mathbf{c}(0)$ is known, the set of coefficients $(d_{\mathbf{l}}(0))_{\mathbf{l}\in\mathcal{L}}$ is computable from $\mathbf{c}(0)$ and $\mathbf{P}^{(\mathbf{l})}\mathbf{c}(0)$ in $\mathcal{O}(|\mathcal{L}||\mathcal{K}|)$. If the squared modulus of $\psi_{\mathcal{K}}$ is considerably smoother than $\psi_{\mathcal{K}}$ itself, we can choose $\mathcal{L}$ such that

$$|\mathcal{K}(d,K)| \gg |\mathcal{L}(d,L)|, \qquad K \gg L,$$

which parallels the smoothness assumptions on ψ and the potential W as discussed in Section 1.4. We are thus in need of an efficient procedure for the matrix-vector product of the form $\mathbf{P}^{(\mathbf{l})}\mathbf{v}$ for some $\mathbf{v} \in \mathbb{C}^{|\mathcal{K}|}$. Given such a procedure that also takes only $\mathcal{O}(|\mathcal{L}||\mathcal{K}|)$ operations, we end up with an overall algorithm that scales essentially linearly with $|\mathcal{K}|$.

We write (6.12) equivalently as

$$i\dot{\mathbf{c}}(t) = \big(\mathbf{W}(t) + \sum_{\mathbf{l}\in\mathcal{L}} \mathbf{c}^*(t)\mathbf{P}^{(\mathbf{l})}\mathbf{c}(t)\,\mathbf{P}^{(\mathbf{l})}\big)\mathbf{c}(t). \tag{6.14}$$

This is the spatially discrete system as we actually propagate it in time.

6.4.4 Factorization of triple products

Our aim is to factorize the triple-product matrices $\mathbf{P}^{(\mathbf{l})}$ in a way that allows for an efficient procedure for corresponding matrix-vector products. As shown in Section 2.6,

$$\operatorname{diag}_{\mathbf{m}\in\mathcal{K}}\big(-\frac{1}{2}\sum_{\alpha=1}^{d}\xi_{m_\alpha}^2\big) = \mathbf{U}\big(-\frac{1}{2}\sum_{\alpha=1}^{d}(\mathbf{X}^{(\alpha)})^2\big)\mathbf{U}^T,$$

where $\mathbf{U}$ and $\mathbf{X}^{(\alpha)}$ are defined as in (2.16) and (2.4), respectively. This implies

$$\begin{aligned}\mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(\varphi_{\mathbf{0}}(\xi_{\mathbf{m}}))\mathbf{U} &= \pi^{-d/4}\mathbf{U}^T \exp\big(\operatorname{diag}_{\mathbf{m}\in\mathcal{K}}\big(-\frac{1}{2}\sum_{\alpha=1}^{d}\xi_{m_\alpha}^2\big)\big)\mathbf{U}\\ &= \pi^{-d/4}\exp\big(-\frac{1}{2}\sum_{\alpha=1}^{d}(\mathbf{X}^{(\alpha)})^2\big).\end{aligned}$$

Setting $f_{\mathbf{l}} = f_{l_1}\cdot\ldots\cdot f_{l_d}$, we can thus approximate $\mathbf{P}^{(\mathbf{l})}$ by quadrature and factorize the approximation according to

$$\begin{aligned}\mathbf{P}^{(\mathbf{l})} &= (\varphi_{\mathbf{k}}, \varphi_{\mathbf{l}}\varphi_{\mathbf{k}})_{\mathbf{j},\mathbf{k}\in\mathcal{K}}\\ &\approx ((\varphi_{\mathbf{k}}, \varphi_{\mathbf{l}}\varphi_{\mathbf{k}})_{\mathrm{quad}})_{\mathbf{j},\mathbf{k}\in\mathcal{K}} = f_{\mathbf{l}}((\varphi_{\mathbf{j}}, \varphi_{\mathbf{0}}H_{\mathbf{l}}\varphi_{\mathbf{k}})_{\mathrm{quad}})_{\mathbf{j},\mathbf{k}\in\mathcal{K}}\\ &= f_{\mathbf{l}}\mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}\big(H_{\mathbf{l}}(\xi_{\mathbf{m}})\varphi_{\mathbf{0}}(\xi_{\mathbf{m}})\big)\mathbf{U}\\ &= f_{\mathbf{l}}\mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}\big(H_{\mathbf{l}}(\xi_{\mathbf{m}})\big)\mathbf{U}\mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}\big(\varphi_{\mathbf{0}}(\xi_{\mathbf{m}})\big)\mathbf{U}\\ &= \pi^{-d/4} f_{\mathbf{l}} H_{\mathbf{l}}(\mathbf{X})\exp\big(-\frac{1}{2}\sum_{\alpha=1}^{d}(\mathbf{X}^{(\alpha)})^2\big).\end{aligned}$$

Turning back to the system (6.14), we are thus concerned with the matrix-vector product

$$\sum_{\mathbf{l}\in\mathcal{L}} \mathbf{v}^*\mathbf{P}^{(\mathbf{l})}\mathbf{v}\,\mathbf{P}^{(\mathbf{l})}\mathbf{v} \approx \pi^{-d/2}\sum_{\mathbf{l}\in\mathcal{L}} f_{\mathbf{l}}^2\mathbf{v}^* H_{\mathbf{l}}(\mathbf{X})\mathbf{w}H_{\mathbf{l}}(\mathbf{X})\mathbf{w}, \qquad \mathbf{v}\in\mathbb{C}^{|\mathcal{K}|}, \tag{6.15}$$

where

$$\mathbf{w} = \exp\big(-\frac{1}{2}\sum_{\alpha=1}^{d}(\mathbf{X}^{(\alpha)})^2\big)\mathbf{v}.$$

In Section 6.4.5, we present both a direct evaluation procedure for the computation of $\mathbf{w}$ and a modification of the above fast algorithm for the computation of the product (6.15). Applying these procedures sequentially then yields a linearly scaling procedure for the overall matrix-vector product.

6.4.5 Efficient matrix-vector products

We recall the direct operation procedure with the coordinate matrices given in Section 2.2, viz.,

$$\big(\mathbf{X}^{(\alpha)}\mathbf{v}\big)_{\mathbf{j}} = \sqrt{\frac{j_\alpha}{2}}v_{\mathbf{j}-\mathbf{e}_\alpha} + \sqrt{\frac{j_\alpha+1}{2}}v_{\mathbf{j}+\mathbf{e}_\alpha} \tag{6.16}$$

for all $\mathbf{j}\in\mathcal{K}$, where $\mathbf{e}_\alpha$ is again the αth unit vector, and $v_{\mathbf{j}-\mathbf{e}_\alpha} = 0$ if $j_\alpha = 0$ and $v_{\mathbf{j}+\mathbf{e}_\alpha} = 0$ if $j_\alpha = K$. Repeating this, we find

$$\begin{aligned}\big(-\frac{1}{2}\sum_{\alpha=1}^{d}(\mathbf{X}^{(\alpha)})^2\mathbf{v}\big)_{\mathbf{j}} = \sum_{\alpha=1}^{d}\Big[&-\frac{1}{4}\Big(\sqrt{j_\alpha(j_\alpha-1)}v_{\mathbf{j}-2\mathbf{e}_\alpha} + j_\alpha v_{\mathbf{j}}\Big)(1-\delta_{j_\alpha 0})\\ &-\frac{1}{4}\Big((j_\alpha+1)v_{\mathbf{j}} + \sqrt{(j_\alpha+1)(j_\alpha+2)}v_{\mathbf{j}+2\mathbf{e}_\alpha}\Big)(1-\delta_{j_\alpha K})\Big]\end{aligned} \tag{6.17}$$

Algorithm 9: Recursive procedure for

$$\mathbf{y} = \sum_{\mathbf{l}\in\mathcal{L}} f_{\mathbf{l}}^2 \mathbf{v}^* H_{\mathbf{l}}(\mathbf{X})\mathbf{w} H_{\mathbf{l}}(\mathbf{X})\mathbf{w}$$

for given vectors $\mathbf{v}$, $\mathbf{w} \in \mathbb{C}^{|\mathcal{K}|}$ and given coefficients $f_{\mathbf{l}}$, $\mathbf{l} \in \mathcal{L}$, starting from $\mathbf{y} = \mathbf{0}$, $\alpha = 1$, and $\mathbf{l} = \mathbf{0} \in \mathbb{N}^d$.

$\mathbf{z} =$ fastalgorithm $(\mathbf{v}, \mathbf{w}, \mathbf{y}, \alpha, \mathbf{l})$
for $l = 0$ **to** L **do**
  if $l = 0$ **then**
    $\mathbf{w}_- := \mathbf{w}$
  else if $l = 1$ **then**
    $\mathbf{w}_+ := 2\mathbf{X}^{(\alpha)}\mathbf{w}$ *use* (6.16)*:* $\mathcal{O}(|\mathcal{K}|)$
  else
    temp $:= \mathbf{w}_+$
    $\mathbf{w}_+ := 2\mathbf{X}^{(\alpha)}\mathbf{w}_+ - 2l\mathbf{w}_-$ *use* (6.19) *together with* (6.16)*:* $\mathcal{O}(|\mathcal{K}|)$
    $\mathbf{w}_- :=$ temp
  $\tilde{\mathbf{w}} := \begin{cases} \mathbf{w}_-, & l = 0, \\ \mathbf{w}_+, & \text{else}, \end{cases}$
  $\mathbf{l} := \mathbf{l} \overset{\alpha}{\leftarrow} l$
  if $\alpha < d$ **then**
    fastalgorithm $(\mathbf{v}, \tilde{\mathbf{w}}, \mathbf{y}, \alpha+1, \mathbf{l})$ *recursion: next coordinate*
  else
    $z := \mathbf{v}^*\tilde{\mathbf{w}}$ *scalar product:* $\mathcal{O}(|\mathcal{K}|)$
    $\mathbf{y} := \mathbf{y} + f_{\mathbf{l}}^2 z\tilde{\mathbf{w}}$ *last coordinate: sum up*

for all $\mathbf{j} \in \mathcal{K}$, where $v_{\mathbf{j}-2\mathbf{e}_\alpha} = 0$ if $j_\alpha \in \{0, 1\}$ and $v_{\mathbf{j}+2\mathbf{e}_\alpha} = 0$ if $j_\alpha \in \{K-1, K\}$. Thus, $\mathbf{X}^{(\alpha)}\mathbf{v}$ is obtained via (6.16) in $\mathcal{O}(|\mathcal{K}|)$ operations, and $-\frac{1}{2}\sum_{\alpha=1}^d (\mathbf{X}^{(\alpha)})^2\mathbf{v}$ is obtained via (6.17) in $\mathcal{O}(d|\mathcal{K}|)$ operations.

We approximate the matrix exponential

$$\exp\big(-\frac{1}{2}\sum_{\alpha=1}^d (\mathbf{X}^{(\alpha)})^2\big)\mathbf{v} \approx \mathbf{V}_m \exp(\mathbf{T}_m) e_1$$

by a Galerkin ansatz based on an m-step Hermitian Lanczos process as given in Section 1.5. In each step of the Lanczos process, we evaluate the product of the Hermitian matrix $-\frac{1}{2}\sum_{\alpha=1}^d (\mathbf{X}^{(\alpha)})^2\big)$ with a vector using the efficient procedure (6.17). The overall Lanczos process takes $\mathcal{O}(md|\mathcal{K}|)$ operations.

We now turn to the computation of

$$\pi^{-d/2} \sum_{\mathbf{l}\in\mathcal{L}} f_{\mathbf{l}}^2 \mathbf{v}^* H_{\mathbf{l}}(\mathbf{X})\mathbf{w}\, H_{\mathbf{l}}(\mathbf{X})\mathbf{w}, \qquad \mathbf{w} \text{ given}, \tag{6.18}$$

and propose a simple modification of the procedure presented in Section 2.3. In 1D, (6.18)

can be computed using the Hermite recurrence relation

$$\begin{aligned} H_{l+1}(X)\mathbf{w} &= 2XH_l(X)\mathbf{w} - 2lH_{l-1}(X)\mathbf{w}, \\ H_1(X)\mathbf{w} &= 2X\mathbf{w}, \qquad H_0(X)\mathbf{w} = \mathbf{w}, \end{aligned} \tag{6.19}$$

(instead of its Chebyshev analog as used before) in $\mathcal{O}(LK)$ operations, where X is again the 1D coordinate matrix. The generalization to higher dimensions then fully parallels the proceeding in Section 2.3. As compared to Algorithm 5, in the last lines of the below Algorithms 9, we additionally need to compute a scalar product (line **16**) and multiply with $f_{\mathbf{l}}^2$ instead of the Chebyshev expansion coefficient $\hat{w}_{\mathbf{r}}$ (line **17**).

At the very end of doing the matrix-vector product, we need to multiply with $\pi^{-d/2}$ once.

6.4.6 Algorithmic description

Having chosen spatial discretization parameters K, L and R for the multi-index sets $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{R}$ that underly the Hermite and Chebyshev approximations $\psi \approx \psi_{\mathcal{K}}$, $|\psi_{\mathcal{K}}|^2 \approx \mathcal{P}_{\mathcal{L}}|\psi_{\mathcal{K}}|^2$, and $W(\cdot,t) \approx W^{\mathrm{pol}}(\cdot,t)$, respectively, we propagate the system

$$i\dot{\mathbf{c}}(t) = \mathbf{D}\mathbf{c}(t) + \big(\mathbf{W}(t) + \sum_{\mathbf{l}\in\mathcal{L}} \mathbf{c}^*(t)\mathbf{P}^{(\mathbf{l})}\mathbf{c}(t)\,\mathbf{P}^{(\mathbf{l})}\big)\mathbf{c}(t), \tag{6.20}$$

for some given initial value $\mathbf{c}(0)$. We employ the Strang splitting scheme as given in Section 6.4.2.

The harmonic oscillator part is trivially propagated exactly with linearly scaling costs. To illustrate time-propagation of the remainder part using the efficient procedures devised in Section 6.4.5, we choose, e.g., the classical 4th-order Runge–Kutta method. This yields an approximation B^{discr} to the operator B; see Section 6.4.2. The matrix-vector products of the factorized form (6.15) and with the matrix representation of the polynomial potential W^{pol} as required for the computation of the Runge–Kutta increments is done using the efficient procedures given in Section 6.4.5 and in Chapter 2, respectively. No matrices are assembled.

Given a choice of time step size h and a choice of number of Lanczos steps m for the matrix exponentials as occurring in (6.15), propagation is explicitly done as given in Algorithm 10. The overall computational costs for a single time step scale as

$$\mathcal{O}\big((|\mathcal{L}| + |\mathcal{R}| + md)|\mathcal{K}|\big).$$

If $|\mathcal{L}|$, $|\mathcal{R}| \ll |\mathcal{K}|$, this is essentially linear in $|\mathcal{K}|$.

Alternatively, for the remainder part, we can employ a Lanczos-based Galerkin approximation to the matrix exponential as required, e.g., when using a Magnus integrator.

Algorithm 10: Time propapgation of (6.20) using the fast algorithm.

timepropagation$(h, m, \mathbf{c}(0))$
$\mathbf{c}_0 := \mathbf{c}(0)$
for $n = 0, 1, \ldots$ **do**
  Harmonic oscillator half-step:
  $\mathbf{c}^A_{n+1} := \Phi_A(h/2)\mathbf{c}_n = \mathrm{diag}_{\mathbf{k}\in\mathcal{K}}\left(\exp\left(-i\frac{h}{2}\sum_{\alpha=1}^d (k_\alpha + \frac{1}{2})\right)\right)\mathbf{c}_n$ $\mathcal{O}(|\mathcal{K}|)$
  Remainder step *for* $\Phi_B^{\mathrm{discr}}(h)\mathbf{c}^A_{n+1}$, *where we choose, e.g., the classical 4th-order RK method:*
  $\mathbf{b}_1 := B^{\mathrm{discr}}(\mathbf{c}^A_{n+1})$; i.e.,
    $\mathbf{b}_1^{(0)} := \exp\left(-\frac{1}{2}\sum_{\alpha=1}^d (\mathbf{X}^{(\alpha)})^2\right)\mathbf{c}^A_{n+1}$ *Section 6.4.5::* $\mathcal{O}(md|\mathcal{K}|)$
    $\mathbf{b}_1^{(1)} := \pi^{-d/2}\sum_{\mathbf{l}\in\mathcal{L}} f_{\mathbf{l}}^2 (c^A_{n+1})^* H_{\mathbf{l}}(\mathbf{X})\mathbf{b}_1^{(0)}\, H_{\mathbf{l}}(\mathbf{X})\mathbf{b}_1^{(0)}$ *Alg. 9:* $\mathcal{O}(|\mathcal{L}||\mathcal{K}|)$
    $\mathbf{b}_1 := -i(W^{\mathrm{pol}}(\mathbf{X}, t_n)c^A_{n+1} + \mathbf{b}_1^{(1)})$ *Chapter 2:* $\mathcal{O}(|\mathcal{R}||\mathcal{K}|)$
  and analogously for
  $\mathbf{b}_2 := B^{\mathrm{discr}}(\mathbf{c}^A_{n+1} + \frac{h}{2}\mathbf{b}_1)$ $\mathcal{O}((|\mathcal{L}| + |\mathcal{R}| + md)|\mathcal{K}|)$
  $\mathbf{b}_3 := B^{\mathrm{discr}}(\mathbf{c}^A_{n+1} + \frac{h}{2}\mathbf{b}_2)$ $\mathcal{O}((|\mathcal{L}| + |\mathcal{R}| + md)|\mathcal{K}|)$
  $\mathbf{b}_4 := B^{\mathrm{discr}}(\mathbf{c}^A_{n+1} + h\mathbf{b}_3)$ $\mathcal{O}((|\mathcal{L}| + |\mathcal{R}| + md)|\mathcal{K}|)$
  $\mathbf{c}^B_{n+1} := \mathbf{c}^A_{n+1} + \frac{h}{6}(\mathbf{b}_1 + 2\mathbf{b}_2 + 2\mathbf{b}_3 + \mathbf{b}_4)$
  Harmonic oscillator half-step:
  $\mathbf{c}_{n+1} := \Phi_A(h/2)\mathbf{c}^B_{n+1}$ $\mathcal{O}(|\mathcal{K}|)$

II.

Application to initial-boundary value problems

7 Introduction

When solving, for instance, the linear Schrödinger equation (1.1) using a spectral method, there are essentially two ways to determine the unknown coefficients $(c_{\mathbf{k}})_{\mathbf{k}\in\mathcal{K}}$ in the linear expansion $\psi_{\mathcal{K}}$. The first way is a Galerkin approach as used in Part I of this thesis. The second way is collocation: to require that $\psi_{\mathcal{K}}$ satisfy the equation in a finite number of grid points, where the number of points equals the number of unknown coefficients. Let us begin with a few remarks on the latter approach. Consider, for the sake of simplicity, the Schrödinger equation on the domain $[-\pi,\pi]^d$ with periodic boundary conditions. Using a trigonometric polynomial as an approximate,

$$\psi_{\mathcal{K}}(x,t) = \sum_{k_1=-K/2}^{K/2-1} \dots \sum_{k_d=-K/2}^{K/2-1} c_{\mathbf{k}}(t)\exp(i\mathbf{k}\cdot x),$$

and a full grid of K points per coordinate, we require that

$$i(\psi_{\mathcal{K}})_t(\xi_{\mathbf{j}},t) = -\frac{1}{2}\Delta\psi_{\mathcal{K}}(\xi_{\mathbf{j}},t) + V(\xi_{\mathbf{j}},t)\psi_{\mathcal{K}}(\xi_{\mathbf{j}},t),$$

where

$$\xi_{\mathbf{j}} = (\xi_{j_1},\dots,\xi_{j_d}), \quad \xi_{j_\alpha} = j_\alpha\cdot 2\pi/K, \qquad -K/2 \le j_\alpha \le K/2-1, \quad \alpha = 1,\dots,d.$$

In coefficient space, this can equivalently by written as

$$i\dot{\mathbf{c}}(t) = \operatorname{diag}_{\mathbf{j}\in\mathcal{K}}\left(\frac{1}{2}\sum_\alpha j_\alpha^2\right)\mathbf{c}(t) + \mathcal{F}\operatorname{diag}_{\mathbf{j}\in\mathcal{K}}\left(V(\xi_{\mathbf{j}},t)\right)\mathcal{F}^{-1}\mathbf{c}(t),$$

where $\mathcal{F}$ denotes the d-dimensional discrete Fourier transform of length K; see [61], Chapter III.1.3. The Fast Fourier Transform (FFT) together with its inverse allow to compute the action of $\mathcal{F}$ and $\mathcal{F}^{-1}$ in $\mathcal{O}(dK^d\log(K))$; see, e.g., [72], Chapter 12. Derivatives are thus trivially done in coefficient space, while multiplication with a potential becomes trivial in physical space, and FFT enables us to switch between the two spaces. By virtue of the fast transform, the costs for an evaluation of the right-hand side scale essentially linearly with the size of the full tensor grid. In general, having a basis that admits for a fast transform between expansion coefficients $(c_{\mathbf{k}})_{\mathbf{k}\in\mathcal{K}}$ and grid values $(\psi_{\mathcal{K}}(\xi_{\mathbf{j}}))_{\mathbf{j}\in\mathcal{K}}$ yields an essentially linearly scaling method; for derivative operators other than the Laplacian, or for spectral bases other than trigonometric functions, corresponding diagonal derivative matrices have been devised for a variety of problems throughout the literature on spectral methods; see again, e.g., [10, 15, 26, 38, 47, 76, 88].

The above collocation ansatz is easily seen to be equivalent to a corresponding Galerkin ansatz if each entry

$$\left(\frac{1}{2\pi}\right)^d \int_{[-\pi,\pi]^d} \exp(-i\mathbf{j}\cdot x)V(x,t)\exp(i\mathbf{k}\cdot x)\,\mathrm{d}x \approx \frac{1}{K^d}\sum_{\mathbf{l}} \exp(-i\mathbf{j}\cdot\xi_{\mathbf{l}})V(\xi_{\mathbf{l}},t)\exp(i\mathbf{k}\cdot\xi_{\mathbf{l}})$$

of the matrix representation of V is replaced by a trapezoidal sum approximation. The equivalence "collocation = Galerkin + quadrature" is a generic feature for full index sets and full tensor grids; see, e.g., [10], Chapter 4.4. Analogously, in the setting of Chapter 1, using full-product $(K+1)$-nodes Gauß–Hermite quadrature is the same as collocation with $\xi_{\mathbf{j}}$ being tuples of zeros of H_{K+1}. Thus, fast transforms provide a loophole for the computational challenges posed by both kinds of spectral methods based on full index sets or tensor grids.

As we have already mentioned in the introduction to Part I, fast transforms have also been devised for reduced grids, where the equivalence between the collocation ansatz and the Galerkin ansatz in combination with quadrature is lost, though. We refer to the literature given above. However, in Part II, we restrict our attention to the full index cube. This is for two reasons: First, we consider boundary value problems, where d is typically small (say, $d = 2, 3$), and reducing the spectral basis is therefore less urgent a need—provided one manages to come up with a linearly scaling method. Second, although our method does scale linearly, it turns out not to bear a significant advantage when reducing the basis because the time-error ratio does not improve sufficiently (or even becomes worse). We shall comment on this issue in Section 11.4.

What if, for a particular basis that is favorable for some reason, there is no fast transform? We use, e.g., a basis of Legendre polynomials, because it yields an efficient way to do derivatives, but Legendre polynomials do not come with a fast transform (see [16], though). Or what if the problem exhibits non-trivial boundary conditions such that a basis that allows for a fast transform (e.g., a Fourier or a Chebyshev basis) is inappropriate? To complicate matters, we add non-constant coefficients to the derivatives. Normally, as stated above, one wishes to do derivatives in coefficient space and multiplication with non-constant coefficients in physical space. This road is now blocked, as we can no longer switch efficiently between the spaces. The fast algorithm in its basic form can already be seen as a substitute for a missing fast transform, but it is serviceable also in cases where fast transforms do exist.

The fast algorithm presented in Part I for the Schrödinger equation on an unbounded domain is free of any transformations and still scales linearly. As we will show, even with non-trivial boundary conditions and derivatives with non-constant coefficients, its methodology carries over to initial-boundary value problems discretized in space by a Legendre–Galerkin approach. We briefly summarize the main ideas of the subsequently devised method in an abstract manner. First, we impose boundary conditions weakly. This shall be discussed separately in a subsequent paragraph. Second, we suggest working in the Legendre coefficient space instead of in a nodal basis, and, in higher dimensions, tensor products of univariate Legendre polynomials are employed. The choice of Legendre polynomials as Galerkin basis functions makes evaluations of derivatives feasible in linear time using well-known recurrence relations of the Legendre polynomials. Third, in Legendre space, however, variable coefficients lead to dense matrices. Still, the corresponding matrix-vector products can also be evaluated efficiently in essentially linear time by an appropriate enhancement of the

fast algorithm in its basic form. Again, we have to assume that the coefficients are much smoother than the solution. This is a typical situation: For wave propagation problems, or in the presence of boundary layers, the solution can exhibit rapid oscillations despite the coefficients of the PDE being smooth. These are problems where spectral and high order methods are advantageous.

A related technique for one-dimensional boundary value problems was presented in [68]. It operates in a Chebyshev basis and treats boundary conditions using boundary bordering [69]. Similarly as we do in our matrix-free, recursive procedure, they approximate variable coefficients by polynomials. Thereby, they get banded discretization matrices, and thus a linearly scaling method. The method was extended to boundary value problems in two dimensions in [87], albeit no longer with linear complexity.

We conclude this introduction with some comments on the way we impose boundary conditions. There are two main approaches to enforce boundary conditions: Strongly, where the solution is set to satisfy the boundary conditions exactly, and weakly, where penalty terms force the solution to satisfy the boundary conditions approximately. Enforcing boundary conditions in a strong way may seem more natural and, at least for problems with certain simple boundary conditions and constant coefficients, problem-specific bases built from standard basis functions can be constructed [76]. E.g., so-called recombinations of Chebyshev polynomials yield an easy way to deal with homogeneous Dirichlet boundary conditions on $[-1,1]^d$:

$$\varphi_k(x) = \begin{cases} T_k(x) - T_0(x), & k \text{ even}, \\ T_k(x) + T_0(x), & k \text{ odd}, \end{cases} \qquad \text{such that } \varphi_k(\pm 1) = 0.$$

However, the convenience of using a standard basis is lost, and implementing more general boundary conditions strongly is not straightforward. Stability of the resulting scheme can also be an issue; see [18]. Weak enforcement of boundary conditions using penalties was introduced for spectral methods in [28, 29]. Penalty methods are flexible, and can be applied to a good variety of boundary conditions for PDEs of different type. By construction, the resulting spatial discretizations satisfy similar energy estimates as the continuous problems, which in combination with suitable time-stepping guarantees stability. Penalty methods are not only used for spectral methods. They are essential building blocks for discontinuous Galerkin (DG) methods (see [21, 48, 66]) , and also the dominant way of imposing boundary conditions for finite difference methods on summation by parts (SBP) form; see [18, 63, 75]. Weak boundary conditions are typically implemented analogously for DG, SBP and spectral methods. For spectral penalty methods, one typically uses a Lagrange basis on Gauß–Lobatto quadrature nodes; see [17, 25, 45, 46]. The motivation is that if the corresponding quadrature rule is used to evaluate the integrals defining it, the mass matrix becomes diagonal. This simplifies the implementation and improves the efficiency of the method, not least by facilitating explicit time-stepping. Differentiation matrices are however still dense. Unfortunately, penalty methods cannot be used together with the Chebyshev weight function since it is singular at the boundaries. One can therefore not, as far as we know, use weak boundary conditions and at the same time take advantage of diagonal mass matrices and the fast Fourier transform. The present approach combines the flexibility of weak enforcement of boundary conditions with the convenience of a standard Legendre basis and still yields a linearly scaling scheme.

The second part of this thesis is organized as follows. In Chapter 8, we present our Legendre–Galerkin spectral method. For illustration purposes, we use the example of a wave equation with spatially variable coefficients on the unit hypercube. In Chapter 9, we explain how to compute the different kinds of matrix-vector products in linear time, without assembling any matrices. Chapter 10 contains the error analysis. Numerical experiments for the wave equation in curvilinear coordinates are presented in Chapter 11.

8 Spectral approximation of the wave equation

We illustrate possible generalizations of the fast algorithm as given in Part I according to the program outlined in the above introduction to Part II. Although the method is applicable under much less severe restrictions than the following, since we want to demonstrate its strengths, we impose a few criteria for an interesting problem setting: We seek for an educational example that

- exhibits derivatives with non-constant coefficients that cannot be easily converted into trivial (i.e., linearly scaling) multiplications,
- comes with boundary conditions that do not easily allow for a strong imposition using appropriate recombinations of classical orthogonal polynomials as basis functions,

and combine this with a standard choice of Galerkin basis that

- meets the requirements for the fast algorithm to be applicable as they are given in Section 6.1,
- generalizes to arbitrary dimensions due to its tensor product structure,
- but does not yield a fast transform,
- and still allows for an overall algorithmic treatment that scales only linearly with the size of the basis in each time step.

The methodology is general, though, and can be adapted to well-posed initial-boundary value problems other than the following example.

In Section 8.1, we introduce the wave equation with non-constant coefficients on a d-dimensional hypercube as an illustrative example for our method, and we prove stability of the solution. Stability of the subsequent semidiscrete approximation is later deduced by arguments that parallel the continuous proceeding. Section 8.2 contains the Galerkin approach where weak imposition of boundary conditions is also discussed. In Section 8.3, we present the Legendre polynomial basis together with its key features as they are later used when devising efficient procedures for the corresponding matrix-vector products.

8.1 The wave equation

In the below numerical experiments given in Chapter 11, we consider the acoustic wave equation in Cartesian coordinates and with an isotropic medium on some bounded domain, say $\tilde{\Omega} \subset \mathbb{R}^d$. We assume that there exists a smooth curvilinear coordinate transformation from $\tilde{\Omega}$ to

$$\Omega = [-1, 1]^d.$$

This yields an abstract wave equation with variable coefficients on the d-dimensional hypercube. Now, we start right from such an abstract problem, viz.,

$$\psi_{tt}(x,t) = \nabla \cdot a(x)\nabla\psi(x,t) + f(x), \qquad x \in \Omega,\ t \geq 0, \tag{8.1}$$
$$\psi(x,0) = \psi_1(x), \qquad x \in \Omega, \tag{8.2}$$
$$\psi_t(x,0) = \psi_2(x), \qquad x \in \Omega, \tag{8.3}$$
$$b(x)\psi_t(x,t) + \mathbf{n} \cdot a(x)\nabla\psi(x,t) = g(x,t), \qquad x \in \partial\Omega,\ t \geq 0, \tag{8.4}$$

with coefficient functions $a(x)$ and $b(x)$. We assume that a is a symmetric and positive definite $(d \times d)$-matrix with real entries and that b is a real-valued function such that

$$\det(a(x)) \geq a_0 > 0, \quad b(x) \geq b_0 > 0, \qquad x \in \Omega, \tag{8.5}$$

and that the norms of a and b and all elements of a and are bounded for all x,

$$\|a(x)\| \leq a_{\max} < \infty, \qquad |a_{\alpha\beta}(x)| \leq a_{\max}, \quad 1 \leq \alpha, \beta \leq d, \qquad b(x) \leq b_{\max}.$$

The vector $\mathbf{n}$ denotes the outward unit normal. Again, we denote the standard $L^2(\Omega)$-inner product and norm by $(\cdot,\cdot)$ and $\|\cdot\|$, respectively. Similarly, $(\cdot,\cdot)_{\partial\Omega}$ and $\|\cdot\|_{\partial\Omega}$ denote the L^2-inner product and norm on the boundary.

The positivity assumptions (8.5) allows for an application of the energy method from [40] in order to prove strong well-posedness of the continuous problem (8.1)–(8.4). A problem is strongly well-posed if it has a unique solution and there is an appropriate energy seminorm $\|\cdot\|_E$ such that the solution satisfies an energy estimate of the form

$$\|\psi(\cdot,t)\|_E^2 \leq \kappa(T)\Big(\|\psi(\cdot,0)\|_E^2 + \int_0^t (\|f(\cdot,\tau)\|^2 + \|g(\cdot,\tau)\|_{\partial\Omega}^2)\,\mathrm{d}\tau\Big) \quad \forall\, t \in [0,T], \tag{8.6}$$

where κ is independent of ψ_1, ψ_2, f and g, and is bounded for any finite T; see [40]. We define the seminorm

$$\|\psi\|_E = \Big(\|\psi_t\|^2 + (\nabla\psi, a\nabla\psi)\Big)^{1/2}. \tag{8.7}$$

Although the below proof of well-posedness merely uses standard techniques, it still deserves some attention. The reason is that the subsequent spectral approach (viz., the way we impose boundary conditions) gives a scheme that allows us to closely mimic the following continuous stability analysis when proving stability of the spatial semidiscretization; see Section 10.3.

Lemma 8. (well-posedness)
The problem (8.1)–(8.4) *is strongly well-posed in the seminorm* (8.7).

Proof. We use the energy method from [40]. First, we differentiate $\|\psi(\cdot,t)\|_E^2$ with respect to t, use integration by parts, and invoke the boundary conditions (8.4), which yields

$$\begin{aligned}\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|\psi\|_E^2 &= (\psi_t,\psi_{tt}) + (\nabla\psi_t, a\nabla\psi)\\ &= (\psi_t,\nabla\cdot a\nabla\psi) + (\psi_t,f) + (\nabla\psi_t,a\nabla\psi)\\ &= (\psi_t,f) - (\psi_t,b\psi_t)_{\partial\Omega} + (\psi_t,g)_{\partial\Omega}\\ &\le \|\psi_t\|\|f\| + \int_{\partial\Omega}(-b\psi_t^2+\psi_t g)\,\mathrm{d}x,\end{aligned}$$

where we have omitted time-dependency in $\psi(\cdot,t)$. Next, due to the positivity assumption (8.5) on b, we can complete the square, viz.,

$$-b\psi_t^2+\psi_t g = -\left(\sqrt{b}\psi_t - \frac{g}{2\sqrt{b}}\right)^2 + \frac{g^2}{4b}$$

(omitting all arguments). Together with the definition (8.7) of the energy seminorm, we thus have

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|\psi\|_E^2 \le \|\psi\|_E\|f\| - \int_{\partial\Omega}\left(\sqrt{b}\psi_t - \frac{g}{2\sqrt{b}}\right)^2\mathrm{d}x + \int_{\partial\Omega}\frac{g^2}{4b}\,\mathrm{d}x,$$

Using a scaled version of Young's inequality for the product of $\|\psi\|_E\|f\|$ together with the positivity of the squared integrand and the lower bound for b, we find

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\psi\|_E^2 \le \frac{1}{T}\|\psi\|_E^2 + T\|f\|^2 + \frac{1}{2b_0}\|g\|_{\partial\Omega}^2.$$

Finally, by integrating this differential inequality, we arrive at (8.6) with

$$\kappa(T) = \mathrm{e}\max(1, 1/2b_0, T)$$

for all $t\in[0,T]$. □

8.2 Galerkin approach

This sections basically parallels the proceeding for the Hermite–Galerkin approach to the linear Schrödinger equation as given in Section 1.1. Additionally, we have to discuss how the Galerkin ansatz needs to be extend by suitably chosen boundary terms in order to incorporate the boundary conditions.

As for the Galerkin basis, we consider again a set of $L^2(\Omega)$-orthonormal functions. This normalization yields a diagonal mass matrix, which eases both the presentation of the method and computations. In 1D, we choose $L^2([-1,1])$-normalized Legendre polynomials φ_k; see Section 8.3. In higher dimensions, we consider an expansion of tensor products of these univariate Legendre polynomials,

$$\psi_{\mathcal{K}}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x), \qquad \varphi_{\mathbf{k}}(x) = \prod_{\alpha=1}^{d}\varphi_{k_\alpha}(x_\alpha),$$

where $\mathcal{K}$ is the full index cube

$$\mathcal{K} = \mathcal{K}(d, K) = \big\{\mathbf{k} \in \mathbb{N}^d \,:\, 0 \le k_\alpha \le K\big\}.$$

Again, for the sake of brevity, we shall often omit the arguments d and K. Throughout Part II, we omit the subscript "full". We determine the unknown coefficients $\mathbf{c}(t) = (c_\mathbf{k}(t))_{\mathbf{k}\in\mathcal{K}}$ of the approximation $\psi_\mathcal{K}$ to the solution ψ of (8.1)–(8.4) by the Galerkin condition

$$\begin{aligned}(\varphi_\mathbf{j}, (\psi_\mathcal{K})_{tt}(\cdot,t)) = &-(\nabla\varphi_\mathbf{j}, a\nabla\psi_\mathcal{K}(\cdot,t)) \\ &+ (\varphi_\mathbf{j}, \mathbf{n}\cdot a\nabla\psi_\mathcal{K}(\cdot,t))_{\partial\Omega} + (\varphi_\mathbf{j}, f),\end{aligned} \qquad \mathbf{j}\in\mathcal{K}. \tag{8.8}$$

Eq. (8.8) defines a system of ODEs for $\mathbf{c}(t)$, but it does not respect the boundary conditions and it is not stable. We address this by replacing $\mathbf{n}\cdot a\nabla\psi_\mathcal{K}$ using the boundary condition (8.4), as is commonly done in Galerkin methods. The resulting system of ODEs reads

$$\begin{aligned}(\varphi_\mathbf{j}, (\psi_\mathcal{K})_{tt}(\cdot,t)) = &-(\nabla\varphi_\mathbf{j}, a\nabla\psi_\mathcal{K}(\cdot,t)) \\ &- (\varphi_\mathbf{j}, b(\psi_\mathcal{K})_t(\cdot,t))_{\partial\Omega} + (\varphi_\mathbf{j}, f) + (\varphi_\mathbf{j}, g)_{\partial\Omega},\end{aligned} \qquad \mathbf{j}\in\mathcal{K}, \tag{8.9}$$

or, equivalently,

$$\ddot{\mathbf{c}} = -\mathbf{S}\mathbf{c} - \mathbf{B}\dot{\mathbf{c}} + \mathbf{f} + \mathbf{g}, \tag{8.10}$$

with the stiffness and boundary term matrices

$$\begin{aligned}\mathbf{S}_{\mathbf{jk}} &= \sum_{\alpha,\beta=1}^{d} \mathbf{S}_{\mathbf{jk}}^{(\alpha,\beta)} = \sum_{\alpha,\beta=1}^{d} \Big(\frac{\partial}{\partial x_\alpha}\varphi_\mathbf{j}, a^{(\alpha,\beta)}\frac{\partial}{\partial x_\beta}\varphi_\mathbf{k}\Big), \\ \mathbf{B}_{\mathbf{jk}} &= (\varphi_\mathbf{j}, b\varphi_\mathbf{k})_{\partial\Omega},\end{aligned} \qquad \mathbf{j},\,\mathbf{k}\in\mathcal{K}, \tag{8.11}$$

respectively, and load vectors

$$\mathbf{f}_\mathbf{j} = (\varphi_\mathbf{j}, f), \quad \mathbf{g}_\mathbf{j} = (\varphi_\mathbf{j}, g)_{\partial\Omega}, \qquad \mathbf{j}\in\mathcal{K}.$$

We will prove stability for this system or, more precisely, for a suitable approximation to (8.9), in Section 10.3.

The replacement of the boundary term above can also be seen as adding penalty terms to the scheme in order to enforce the boundary conditions weakly. We did indeed add

$$(\varphi_\mathbf{j}, g)_{\partial\Omega} - (\varphi_\mathbf{j}, \mathbf{n}\cdot a\nabla\psi_\mathcal{K}(\cdot,t))_{\partial\Omega} + (\varphi_\mathbf{j}, b(\psi_\mathcal{K})_t(\cdot,t))_{\partial\Omega}$$

to the right-hand side of (8.8). These terms vanish when the boundary conditions are satisfied and penalize deviation from the boundary conditions. The penalty terms carry a large weight in physical space: As can be seen by a rather tedious exercise, their contribution to $(\psi_\mathcal{K})_{tt}(x,t)$, $x\in\partial\Omega$, is of order K^2. That is, the enforcement of the boundary conditions becomes stronger when the basis is enlarged. In general, for linear problems, if the added penalty terms (i) vanish when $\psi_\mathcal{K}(x,t)$ satisfies the boundary conditions, and (ii) make the semidiscretization stable, we can expect the scheme to converge.

8.3 Legendre basis

The classical orthogonal Legendre polynomials, P_k, are normalized with respect to their boundary values, viz., $P_k(1) = 1$ and $P_k(-1) = (-1)^k$, and the $L^2([-1,1])$-inner product of two Legendre polynomials evaluates to $(P_j, P_k) = 2/(2j+1)\delta_{jk}$. See again, e.g., [1], Section 22, for basic properties of P_k. We aim at a renormalization such that the above Galerkin approach yields an identity mass matrix. The desired relation of a set $\{\varphi_k\}_{k\geq 0}$ of $L^2([-1,1])$-normalized Legendre polynomials to their classical counterparts must then read

$$\varphi_k(x) = \sqrt{\frac{2k+1}{2}} P_k(x),$$

which, by the generic recurrence for the classical Legendre polynomials, yields the three-term recurrence

$$\begin{aligned} \varphi_0(x) &= \frac{1}{\sqrt{2}}, \qquad \varphi_1(x) = \sqrt{\frac{3}{2}}x, \\ \varphi_k(x) &= \frac{\sqrt{(2k-1)(2k+1)}}{k} x\varphi_{k-1}(x) - \frac{k-1}{k}\sqrt{\frac{2k+1}{2k-3}}\varphi_{k-2}(x), \quad k \geq 2, \end{aligned} \tag{8.12}$$

on $[-1,1]$. The values at the boundary are given by

$$\varphi_k(1) = \sqrt{\frac{2k+1}{2}}, \qquad \varphi_k(-1) = \sqrt{\frac{2k+1}{2}}(-1)^k,$$

which is also the maximum over $[-1,1]$. For some choices of k, the polynomials φ_k are visualized in Figure 8.1.

Besides having a recurrence for the Galerkin basis functions themselves, as required by the fast algorithm in its basic form, we additionally need an analogous recurrence for their first derivatives, which the efficient evaluation procedur of derivative matrices will be built upon; see Section 9.3. It is easily seen that the derivatives of φ_k obey the relation

$$\begin{aligned} \varphi_0'(x) &= 0, \qquad \varphi_1'(x) = \sqrt{\frac{3}{2}}, \\ \varphi_k'(x) &= \sqrt{(2k-1)(2k+1)}\varphi_{k-1}(x) + \sqrt{\frac{2k+1}{2k-3}}\varphi_{k-2}'(x), \quad k \geq 2, \end{aligned} \tag{8.13}$$

on $[-1,1]$. Due to the presence of φ_{k-1} without a derivative in the second line of (8.13), the two recurrences (8.12) and (8.13) are intertwined. This latter recurrence will more or less immediately translate into an algorithm for the derivatives.

In higher dimensions, we consider again tensor products

$$\varphi_{\mathbf{k}}(x) = \varphi_{k_1}(x_1) \cdot \ldots \cdot \varphi_{k_d}(x_d), \qquad \mathbf{k} \in \mathbb{N}^d, x = (x_1, \ldots, x_d) \in \Omega,$$

of univariate $L^2([-1,1])$-normalized Legendre polynomials, which then constitute an $L^2(\Omega)$-orthonormal set of functions $\{\varphi_{\mathbf{k}}\}_{\mathbf{k}\in\mathbb{N}^d}$. Let $\mathcal{P}_{\mathcal{K}}$ denotes the $L^2(\Omega)$-orthogonal projection onto the polynomial approximation space $\operatorname{span}\{\varphi_{\mathbf{k}}\,;\, \mathbf{k} \in \mathcal{K}(d,K)\}$, where $\mathcal{K}(d,K)$ is the full index cube; cf. (1.5). If $\|\cdot\|_{H^s(\Omega)}$ and $|\cdot|_{H^s(\Omega)}$ denote the standard Sobolev norm

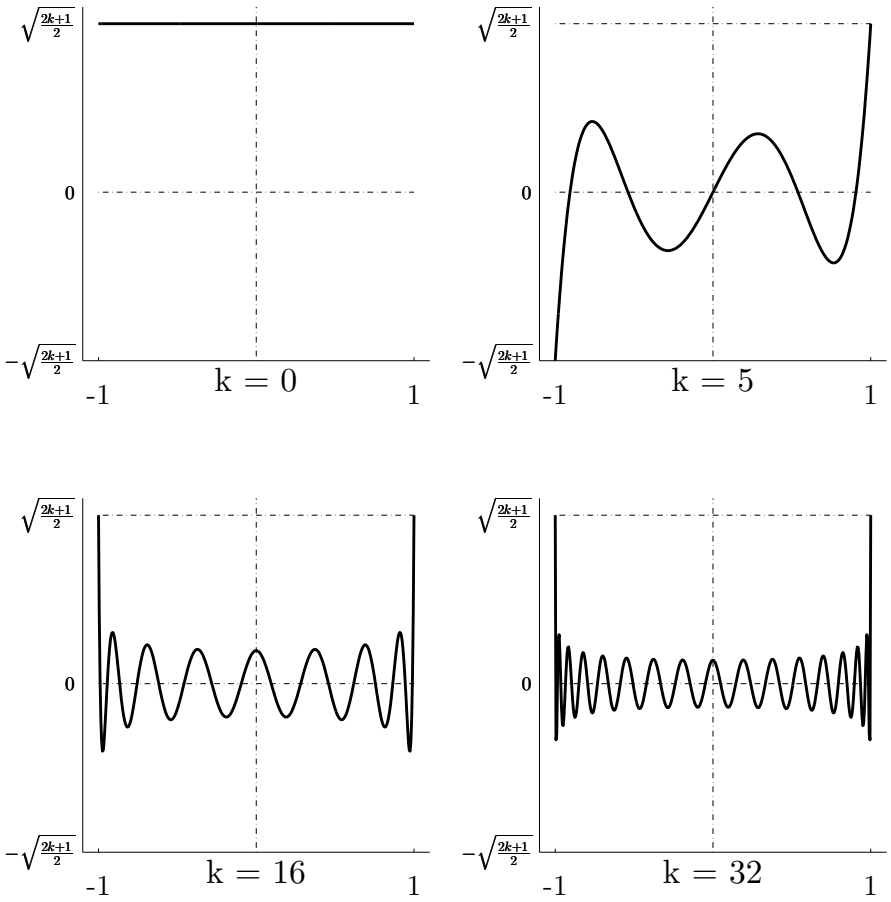


Figure 8.1: $L^2([-1,1])$-normalized Legendre polynomials for some choices of k.

and seminorm, respectively, the truncation error is then given by the standard projection estimate

$$\|\chi - \mathcal{P}_K\chi\|_{H^n(\Omega)} \leq CK^{r-s}|\chi|_{H^s(\Omega)}, \qquad r = \begin{cases} 0, & n = 0, \\ 2n - \frac{1}{2}, & n \geq 1, \end{cases} \tag{8.14}$$

provided χ is sufficiently regular; see [15], Chapter 5.8.2.

9 Efficient procedures for matrix-vector products

In Chapter 11, we propagate (8.10) in time using the classical Runge–Kutta method of order 4. For any explicit Runge–Kutta method or multistep method, time propagation of (8.10) involves computing the actions of the dense matrices $\mathbf{S}$ and $\mathbf{B}$ on vectors in each time step. As in the first part of this thesis, our aim is again to reduce the computational costs for doing these matrix-vector products from a number of operations that scales quadratically with $|\mathcal{K}|$ to a linearly scaling procedure only. We do so by an adaptation of the above fast algorithm as given in Chapter 2 of the first part, which invovles an efficient algorithm for evaluating derivatives, in particular. This is discussed in detail in the present chapter.

As we have seen in Section 2.6, on a full index cube, the fast algorithm for the Schrödinger equation is equivalent to an approximation of the potential matrix representation by suitably chosen Gaussian quadrature. In the context of the wave equation with a differential operator that exhibits non-constant coefficient functions, there also exists a connection between a modified fast algorithm and Gaussian quadrature. In contrast to the presentation chosen in Part I, we now start right from Gaussian quadrature. As a starting point, we approximate $\mathbf{S}$ by Gauss–Legendre quadrature and factorize the approximate matrix $\mathbf{S}_{\text{quad}}$ suitably. Each factor can then be dealt with separately in an efficient way, with computational costs that scale only linearly. The factorization is derived in Section 9.1. In Section 9.2, we give a concise description of the procedure for non-constant coefficients—which is simply the fast algorithm from Part I adapted to a Legendre basis. The routine for the derivative part is discussed in Section 9.3. Section 9.4 contains an efficient routine for the boundary term matrix $\mathbf{B}$, which basically reduces to the original fast algorithm in $d-1$ dimensions. To conclude this chapter, we briefly outline the overall algorithm procedure and comment on its computational complexity in Section 9.5.

9.1 Approximation of matrix-vector products

This time, having a Galerkin basis of Legendre polynomials instead of Hermite functions, we employ Gauß–Legendre quadrature instead of Gauß–Hermite quadrature for an entrywise approximation of the matrix representations. In 1D, this is the instance of Gaussian quadrature that uses the constant unit weight function and approximates integrals of functions over $[-1, 1]$,

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{m=0}^{K} w_m f(\xi_m);$$

see again, e.g., [32], Chapter 3.2. The quadrature nodes ξ_m are the zeros of P_{K+1} with corresponding weights w_m, and the resulting quadrature formula $(w_m, \xi_m)_{m=0}^K$ is exact for polynomials of degree at most $2K+1$. In higher dimensions, we define

$$\xi_{\mathbf{m}} = (\xi_{m_1}, \ldots, \xi_{m_d}), \qquad w_{\mathbf{m}} = \prod_{\alpha=1}^{d} w_{m_\alpha}, \qquad \mathbf{m} \in \mathcal{K}(d, K), \tag{9.1}$$

which constitutes a product of 1D Gauß–Legendre quadrature formulas with $K+1$ nodes in every direction. This d-dimensional quadrature rule is exact for polynomials of degree up to $2K+1$ in each variable. In Part II of this thesis, the subscript or superscript "quad" now refers to this kind of Gaussian quadrature.

We consider again a multivariate polynomial with real coefficients,

$$q : \Omega \to \mathbb{R},$$

which is a placeholder for polynomial approximates to the coefficients functions as occurring in (8.10), and the corresponding matrix representations $\mathbf{Q}$ and $\mathbf{Q}^{(\alpha,\beta)}$ with and without derivatives, respectively,

$$\begin{aligned} \mathbf{Q}_{\mathbf{jk}} &= (\varphi_{\mathbf{j}}, q\varphi_{\mathbf{k}}), \\ \mathbf{Q}^{(\alpha,\beta)}_{\mathbf{jk}} &= \left(\frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{j}}, q \frac{\partial}{\partial x_\beta}\varphi_{\mathbf{k}} \right), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \quad \alpha, \beta = 1, \ldots, d. \end{aligned} \tag{9.2}$$

The approximate counterparts read

$$\begin{aligned} (\mathbf{Q}^{\mathrm{quad}})_{\mathbf{jk}} &= \sum_{\mathbf{m}\in\mathcal{K}} \omega_{\mathbf{m}} \varphi_{\mathbf{j}}(\xi_{\mathbf{m}}) q(\xi_{\mathbf{m}}) \varphi_{\mathbf{k}}(\xi_{\mathbf{m}}), \\ (\mathbf{Q}^{(\alpha,\beta)}_{\mathrm{quad}})_{\mathbf{jk}} &= \sum_{\mathbf{m}\in\mathcal{K}} \omega_{\mathbf{m}} \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{j}}(\xi_{\mathbf{m}}) q(\xi_{\mathbf{m}}) \frac{\partial}{\partial x_\beta}\varphi_{\mathbf{k}}(\xi_{\mathbf{m}}). \end{aligned}$$

Our aim is to devise efficient procedures for a matrix-free computation of $\mathbf{Qv}$ and $\mathbf{Q}^{(\alpha,\beta)}\mathbf{v}$ with a vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$.

In addition to the matrix

$$\mathbf{U}_{\mathbf{jk}} = \sqrt{\omega_{\mathbf{j}}}\varphi_{\mathbf{k}}(\xi_{\mathbf{j}}), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \tag{9.3}$$

(cf. (2.16)), we define the matrices

$$\mathbf{U}^{(\alpha)}_{\mathbf{jk}} = \sqrt{\omega_{\mathbf{j}}} \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}(\xi_{\mathbf{j}}), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \quad \alpha = 1, \ldots, d, \tag{9.4}$$

each now based on the above Gauß–Legendre quadrature. By the same arguments as in Lemma 1, we see that the matrix $\mathbf{U}$ is orthogonal. Finally, we introduce the Galerkin differentiation matrices

$$\mathbf{D}^{(\alpha)}_{\mathbf{jk}} = \left(\varphi_{\mathbf{j}}, \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}} \right), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \quad \alpha = 1, \ldots, d. \tag{9.5}$$

The following lemma shows how these matrices are related.

Lemma 9. (quadrature and derivatives)
The matrices defined in (9.3), (9.4), *and* (9.5) *satisfy*

$$\mathbf{U}^T\mathbf{U}^{(\alpha)} = \mathbf{D}^{(\alpha)}, \qquad \alpha = 1, \dots, d.$$

Proof. Using the exactness of $(K+1)$-nodes full-product Gauß–Legendre quadrature, we can compute

$$\begin{aligned}\big(\mathbf{U}^T\mathbf{U}^{(\alpha)}\big)_{\mathbf{jk}} &= \sum_{\mathbf{m}\in\mathcal{K}} \mathbf{U}_{\mathbf{mj}}\mathbf{U}^{(\alpha)}_{\mathbf{mk}} = \sum_{\mathbf{m}\in\mathcal{K}} w_{\mathbf{m}}\varphi_{\mathbf{j}}(\xi_{\mathbf{m}})\frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}(\xi_{\mathbf{m}}) \\ &= \big(\varphi_{\mathbf{j}}, \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}\big)_{\text{quad}} = \big(\varphi_{\mathbf{j}}, \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}\big) = \mathbf{D}^{(\alpha)}_{\mathbf{jk}}.\end{aligned}$$

□

With the help of Lemma 9, these matrices can be used to reformulate the quadrature approximations of $\mathbf{Q}$ and $\mathbf{Q}^{(\alpha,\beta)}$. In Section 2.6, we have already stated that

$$\mathbf{Q} \approx \mathbf{Q}^{\text{quad}} = \mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(q(\xi_{\mathbf{m}}))\mathbf{U}.$$

As for the derivative matrices $\mathbf{Q}^{(\alpha,\beta)}$, we then find

$$\begin{aligned}\mathbf{Q}^{(\alpha,\beta)} \approx \mathbf{Q}^{(\alpha,\beta)}_{\text{quad}} &= \big(\mathbf{U}^{(\alpha)}\big)^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(q(\xi_{\mathbf{m}}))\mathbf{U}^{(\beta)} \\ &= \big(\mathbf{U}^{(\alpha)}\big)^T\mathbf{U}\mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(q(\xi_{\mathbf{m}}))\mathbf{U}\mathbf{U}^T\mathbf{U}^{(\beta)} \\ &= \big(\mathbf{D}^{(\alpha)}\big)^T\mathbf{Q}^{\text{quad}}\mathbf{D}^{(\beta)},\end{aligned} \tag{9.6}$$

where we use Lemma 9 and the fact $\mathbf{U}\mathbf{U}^T = \mathbf{I}$. The quadrature approximation of $\mathbf{Q}^{(\alpha,\beta)}$ is thus a product of the Galerkin differentiation matrices with the quadrature approximation to $\mathbf{Q}$.

We can now turn to the fast algorithm itself. The coordinate matrices

$$\mathbf{X}^{(\alpha)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, x_\alpha\varphi_{\mathbf{k}}), \qquad \mathbf{j},\mathbf{k}\in\mathcal{K}, \qquad \alpha = 1,\dots,d,$$

are the same as in (2.4), but with Legendre polynomials instead of Hermite functions. The Legendre analog of the equivalence given in Lemma 2 reads

$$\mathbf{Q}^{\text{quad}} = q(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(d)}) = q(\mathbf{X}), \tag{9.7}$$

where the right-hand side denotes again formal insertion of the matrices $\mathbf{X}^{(\alpha)}$ into the polynomial q. We can then use the equivalence (9.7) together with the factorization (9.6) to approximately compute the matrix-vector product

$$\mathbf{Q}^{(\alpha,\beta)}\mathbf{v} \approx \mathbf{Q}^{(\alpha,\beta)}_{\text{quad}}\mathbf{v} = \big(\mathbf{D}^{(\alpha)}\big)^T q(\mathbf{X})\mathbf{D}^{(\beta)}\mathbf{v}$$

for an arbitrary vector $\mathbf{v}\in\mathbb{R}^{|\mathcal{K}|}$. This is done by performing the three matrix-vector products

$$\mathbf{D}^{(\beta)}\mathbf{v}, \qquad q(\mathbf{X})\mathbf{v}, \qquad \big(\mathbf{D}^{(\alpha)}\big)^T\mathbf{v}$$

sequentially. The fast evaluation of these matrix-vector products is discussed in the subsequent sections. Section 9.2 contains the fast evaluation procedure for $q(\mathbf{X})\mathbf{v}$ which is simply the second version of the fast algorithm given in Section 2.3 adapted to a Legendre basis. The products $\mathbf{D}^{(\beta)}\mathbf{v}$ and $\left(\mathbf{D}^{(\alpha)}\right)^T\mathbf{v}$, whose computational costs scale as $|\mathcal{K}|$, are considered in Section 9.3.

Turning back to the system (8.10), we replace all matrices by their quadrature counterparts. If we approximate the coefficient functions $a^{(\alpha,\beta)}$ by polynomial interpolation, say

$$a^{(\alpha,\beta)} \approx a^{(\alpha,\beta)}_{\mathrm{pol}}, \qquad \alpha,\beta = 1,\ldots,d,$$

we can then compute the actions of the corresponding matrices according to

$$\mathbf{S}^{\mathrm{quad}}\mathbf{c} = \sum_{\alpha,\beta=1}^{d} \mathbf{S}^{(\alpha,\beta)}_{\mathrm{quad}}\mathbf{c} = \sum_{\alpha,\beta=1}^{d} \left(\mathbf{D}^{(\alpha)}\right)^T a^{(\alpha,\beta)}_{\mathrm{pol}}(\mathbf{X})\mathbf{D}^{(\beta)}\mathbf{c}. \tag{9.8}$$

The treatment of the boundary term matrix also builts upon the preliminaries given in the present section, but shall be delayed until Section 9.4.

9.2 Fast algorithm for non-constant coefficients

In this section, we briefly describe the efficient algorithm for the matrix-vector product $q(\mathbf{X})\mathbf{v}$, where $\mathbf{X}^{(\alpha)}$ are the coordinate matrices built from the Legendre basis. This section completely mirrors the proceeding given in Chapter 2.

The analog to the direct operation procedure given in Section 2.2 adapted to a Legendre basis reads

$$\left(\mathbf{X}^{(\alpha)}\mathbf{v}\right)_{\mathbf{j}} = \frac{j_\alpha}{\sqrt{(2j_\alpha-1)(2j_\alpha+1)}} v_{\mathbf{j}-\mathbf{e}_\alpha} + \frac{j_\alpha+1}{\sqrt{(2j_\alpha+1)(2j_\alpha+3)}} v_{\mathbf{j}+\mathbf{e}_\alpha} \tag{9.9}$$

for all $\mathbf{j} \in \mathcal{K}$, where $v_{\mathbf{j}-\mathbf{e}_\alpha} = 0$ if $j_\alpha = 0$ and $v_{\mathbf{j}+\mathbf{e}_\alpha} = 0$ if $j_\alpha = K$. We use again orthogonality of the basis together with the recurrence (8.12). This allows to compute $\mathbf{X}^{(\alpha)}\mathbf{v}$ in $\mathcal{O}(|\mathcal{K}|)$ operations. Algorithm 11 shows a brief algorithmic description. Again, we assume q to be in Chebyshev form,

$$q(x) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} T_{\mathbf{r}}(x) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} \prod_{\alpha=1}^{d} T_{r_l}(x_l), \tag{9.10}$$

where the multi-index set $\mathcal{R}(d,R)$ is assumed to satisfy

$$|\mathcal{K}| \gg |\mathcal{R}|, \qquad K \gg R,$$

and we compute the matrix-vector product according to

$$\mathbf{w} = q(\mathbf{X})\mathbf{v} = \sum_{\mathbf{r}\in\mathcal{R}} \hat{q}_{\mathbf{r}} \prod_{\alpha=1}^{d} T_{r_\alpha}(\mathbf{X}^{(\alpha)})\mathbf{v}.$$

Because the dimensionality d of the underlying boundary value problem is normally smaller than for the Schrödinger initial value problem considered in Part I (say, $d = 2,3$), the

fact that the required storage grows linearly in d is not an issue. Hence, the recursive version of the fast algorithm devised in Section 2.3 is clearly preferable over its non-recursive counterpart. Again, we employ the 1D Chebyshev recurrence (2.9), viz.,

$$\begin{aligned} T_0\big(\mathbf{X}^{(\alpha)}\big)\mathbf{v} &= \mathbf{v}, \quad T_1\big(\mathbf{X}^{(\alpha)}\big)\mathbf{v} = \mathbf{X}^{(\alpha)}\mathbf{v}, \\ T_{r+1}\big(\mathbf{X}^{(\alpha)}\big)\mathbf{v} &= 2XT_r\big(\mathbf{X}^{(\alpha)}\big)\mathbf{v} - T_{r-1}\big(\mathbf{X}^{(\alpha)}\big)\mathbf{v}, \quad r \geq 1, \end{aligned}$$

in combination with the above direct operation procedure (9.9) to compute the vector $\mathbf{w}$. The resulting algorithm is identical to the one given as Algorithm 5, but with the procedure directoperation replaced by directoperation-Legendre as given in Algorithm 11.

Algorithm 11: Computation of

$$\mathbf{w} = \mathbf{X}^{(\alpha)}\mathbf{v}$$

in $\mathcal{O}(|\mathcal{K}|)$ using the direct operation given in (9.9).

1 function $\mathbf{w}$ = directoperation-Legendre$(\alpha, \mathbf{v})$
2 **for** $\boldsymbol{j} \in \mathcal{K}$ **do**
3 $$w_{\mathbf{j}} := \begin{cases} \frac{1}{\sqrt{3}} v_{\mathbf{j}+\mathbf{e}_\alpha}, & j_\alpha = 0, \\ \frac{K}{\sqrt{(2K-1)(2K+1)}} v_{\mathbf{j}-\mathbf{e}_\alpha}, & j_\alpha = K, \\ \frac{j_\alpha}{\sqrt{(2j_\alpha-1)(2j_\alpha+1)}} v_{\mathbf{j}-\mathbf{e}_\alpha} + \frac{j_\alpha+1}{\sqrt{(2j_\alpha+1)(2j_\alpha+3)}} v_{\mathbf{j}+\mathbf{e}_\alpha}, & \text{else.} \end{cases}$$

9.3 Derivatives

As follows directly from the recurrence relation (8.13) for the Legendre polynomials, matrix-vector products with the 1D differentiation matrix D,

$$D_{jk} = (\varphi_j, \varphi_k'), \qquad 0 \leq j, k \leq K, \tag{9.11}$$

and a vector $\mathbf{v} \in \mathbb{R}^{K+1}$ can be evaluated recursively as

$$\begin{aligned} (D\mathbf{v})_K &= 0, \qquad (D\mathbf{v})_{K-1} = \sqrt{(2K+1)(2K-1)}\, v_K, \\ (D\mathbf{v})_{j-2} &= \sqrt{\frac{2j-3}{2j+1}} (D\mathbf{v})_j + \sqrt{(2j-1)(2j-3)}\, v_{j-1}, \qquad j = 2, \ldots, K. \end{aligned} \tag{9.12}$$

This is well-known; see, e.g., [15], Chapter 2.3. Clearly, the computational work to compute the resulting vector scales linearly with K. Similarly, the product $D^T\mathbf{v}$ can be evaluated as

$$\begin{aligned} (D^T\mathbf{v})_0 &= 0, \qquad (D^T\mathbf{v})_1 = \sqrt{3}\, v_0, \\ (D^T\mathbf{v})_j &= \sqrt{\frac{2j+1}{2j-3}} (D^T\mathbf{v})_{j-2} + \sqrt{(2j+1)(2j-1)}\, v_{j-1}, \quad j = 2, \ldots, K. \end{aligned} \tag{9.13}$$

As for the order of indices, the recurrence (9.12) is top-down, while the recurrence (9.13) is bottom-up. This is reflected in the subsequent algorithmic descriptions.

In higher dimensions, we wish to evaluate the matrix-vector product

$$\mathbf{w} = \mathbf{D}^{(\alpha)}\mathbf{v} = \Big(\sum_{\mathbf{k}\in\mathcal{K}} (\varphi_{\mathbf{j}}, \frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{k}}) v_{\mathbf{k}} \Big)_{\mathbf{j}\in\mathcal{K}},$$

with a vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$, as well as $(\mathbf{D}^{(\alpha)})^T\mathbf{v}$. Since the basis is orthonormal, we find

$$w_{\mathbf{j}} = \sum_{\mathbf{k}^{(\neg\alpha)}} (\varphi_{\mathbf{j}^{(\neg\alpha)}}, \varphi_{\mathbf{k}^{(\neg\alpha)}}) \sum_{k_\alpha} (\varphi_{j_\alpha}, \varphi'_{k_\alpha}) v_{\mathbf{k}} = \sum_{k_\alpha=0}^{K} (\varphi_{j_\alpha}, \varphi'_{k_\alpha}) v_{\mathbf{j}\xleftarrow{\alpha} k_\alpha}, \qquad \mathbf{j} \in \mathcal{K}.$$

Thus, the computation of $\mathbf{D}^{(\alpha)}\mathbf{v}$ boils down to computing the action of the 1D differentiation matrix D on the 1D vectors

$$\big(v_{\mathbf{j}\xleftarrow{\alpha} k_\alpha}\big)_{k_\alpha=0}^{K}, \qquad \mathbf{j}^{(\neg\alpha)} \in \mathcal{K}(d-1, K),$$

recursively using (9.12). An algorithmic description implementing these ideas is given as Algorithm 12. The matrix-vector product $(\mathbf{D}^{(\alpha)})^T\mathbf{v}$ is computed analogously using (9.13); see Algorithm 13 for an algorithmic description. Obviously, both algorithms scale linearly with $|\mathcal{K}|$.

Algorithm 12: Efficient computation of $\mathbf{w} = \mathbf{D}^{(\alpha)}\mathbf{v}$.

```
1 w = Dalpha (α, v)
2 for j^(¬α) ∈ K(d−1, K) do
3     w_{j ←α K} := 0
4     w_{j ←α K−1} := sqrt((2K+1)(2K−1)) v_{j ←α K}
5     for j_α = K, K−1, ..., 2 do
6         w_{j ←α j_α−2} := sqrt((2j_α−3)/(2j_α+1)) w_{j ←α j_α} + sqrt((2j_α−1)(2j_α−3)) v_{j ←α j_α−1}
```

Algorithm 13: Efficient computation of $\mathbf{w} = (\mathbf{D}^{(\alpha)})^T\mathbf{v}$.

```
1 w = Dalphatransp (α, v)
2 for j^(¬α) ∈ K(d−1, K) do
3     w_{j ←α 0} := 0
4     w_{j ←α 1} := sqrt(3) v_{j ←α 0}
5     for j_α = 2, ..., K do
6         w_{j ←α j_α} := sqrt((2j_α+1)/(2j_α−3)) w_{j ←α j_α−2} + sqrt((2j_α+1)(2j_α−1)) v_{j ←α j_α−1}
```

9.4 Treatment of boundary terms

This section is concerned with an efficient procedure for the matrix-vector product with the boundary term matrix $\mathbf{B}$, cf. (8.10), and a vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$. The work mainly consists in rewriting the matrix-vector product in a suitable way such that the procedure given in Section 9.2 can be applied, but now in $d-1$ dimensions.

First, we introduce some notation which makes the handling of the boundary terms more convenient. We denote the boundary faces by

$$\Omega^{(\pm\alpha)} = \{x \in \Omega \ : \ x_\alpha = \pm 1\}, \qquad \alpha = 1, \dots, d.$$

We also introduce the notation

$$\Omega^{(\neg\alpha)} = \big\{x^{(\neg\alpha)} = (x_1, \dots, x_{\alpha-1}, x_{\alpha+1}, \dots, x_d) \ : \ x \in \Omega\big\},$$

where $x^{(\neg\alpha)}$ denotes the free coordinates on a boundary face and $\Omega^{(\neg\alpha)}$ denotes the dimensional reduction of the domain. Then, if f and g are separable functions, i.e., $f(x) = f(x_\alpha) f_{\neg\alpha}(x^{(\neg\alpha)})$ and similarly for g, their L^2 inner product over a boundary face can be written as

$$(f, g)_{\Omega^{(\pm\alpha)}} = f_\alpha(\pm 1) g_\alpha(\pm 1) (f_{\neg\alpha}, g_{\neg\alpha})_{\Omega^{(\neg\alpha)}}.$$

For an arbitrary polynomial $q : \Omega \to \mathbb{R}$, its restrictions to the boundary faces are denoted by

$$q^{(\pm\alpha)} = q|_{\Omega^{(\pm\alpha)}}, \qquad \alpha = 1, \dots, d. \tag{9.14}$$

Note that $q^{(\pm\alpha)}$ only depends on the coordinates $x^{(\neg\alpha)}$. Using separability, the action of the boundary-restricted matrix

$$\mathbf{Q}^{(\pm\alpha)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, q^{(\pm\alpha)} \varphi_{\mathbf{k}})_{\Omega^{(\pm\alpha)}}, \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}, \tag{9.15}$$

on a vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$ is given by

$$\begin{aligned} \Big(\mathbf{Q}^{(\pm\alpha)} \mathbf{v}\Big)_{\mathbf{j}} &= \sum_{\mathbf{k} \in \mathcal{K}} \Big(\varphi_{\mathbf{j}}, q^{(\pm\alpha)} \varphi_{\mathbf{k}}\Big)_{\Omega^{(\pm\alpha)}} v_{\mathbf{k}} \\ &= \varphi_{j_\alpha}(\pm 1) \sum_{\mathbf{k}^{(\neg\alpha)} \in \mathcal{K}(d-1,K)} \Big(\varphi_{\mathbf{j}^{(\neg\alpha)}}, q^{(\pm\alpha)} \varphi_{\mathbf{k}^{(\neg\alpha)}}\Big)_{\Omega^{(\neg\alpha)}} v^{(\neg\alpha,\pm)}_{\mathbf{k}^{(\neg\alpha)}}, \end{aligned}$$

with

$$\mathbf{v}^{(\neg\alpha,\pm)}_{\mathbf{m}} = \sum_{m=0}^{K} \varphi_m(\pm 1) v_{(m_1,\dots,m_{\alpha-1},m,m_\alpha,\dots,m_{d-1})}, \qquad \mathbf{m} \in \mathcal{K}(d-1, K). \tag{9.16}$$

It takes $\mathcal{O}(|\mathcal{K}(d,K)|)$ operations to compute $\mathbf{v}^{(\neg\alpha,\pm)}$ from $\mathbf{v}$. Using again the equivalence of Gauß–Legendre quadrature and insertion of coordinate matrices into a polynomial, we can thus approximate

$$\Big(\mathbf{Q}^{(\pm\alpha)} \mathbf{v}\Big)_{\mathbf{j}} \approx \Big(\mathbf{Q}^{(\pm\alpha)}_{\text{quad}} \mathbf{v}\Big)_{\mathbf{j}} = \varphi_{j_\alpha}(\pm 1) \Big(q^{(\pm\alpha)}\Big(\mathbf{X}^{(\neg\alpha)}\Big) \mathbf{v}^{(\neg\alpha,\pm)}\Big)_{\mathbf{j}^{(\neg\alpha)}}, \tag{9.17}$$

for all $\mathbf{j} \in \mathcal{K}$, where we set

$$q^{(\pm\alpha)}\left(\mathbf{X}^{(\neg\alpha)}\right) = q^{(\pm\alpha)}\left(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\alpha-1)}, \mathbf{X}^{(\alpha+1)}, \ldots, \mathbf{X}^{(d)}\right),$$

and

$$\mathbf{X}^{(\beta)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, x_\beta \varphi_{\mathbf{k}}), \qquad \mathbf{j}, \mathbf{k} \in \mathcal{K}(d-1, K),$$

Given $\mathbf{v}^{(\neg\alpha,\pm)}$, the action of $q^{(\pm\alpha)}\left(\mathbf{X}^{(\neg\alpha)}\right)$ on a vector can then be computed efficiently using the procedure given in Section 9.2, but now in $d-1$ dimensions.

Turning back to the system (8.10), we define the restrictions of the boundary function b to the boundary faces

$$b^{(\pm\alpha)} = b|_{\Omega^{\pm(\alpha)}}, \qquad \alpha = 1, \ldots, d,$$

and approximate $b^{(\pm\alpha)}$ by polynomial interpolation, say $b^{(\pm\alpha)} \approx b^{(\pm\alpha)}_{\mathrm{pol}}$. The action of $\mathbf{B}$ on $\dot{\mathbf{v}}$ is then computable via

$$\mathbf{B}\dot{\mathbf{v}} = \sum_{\pm\alpha} \mathbf{B}^{(\pm\alpha)}\dot{\mathbf{v}} \approx \sum_{\pm\alpha} \mathbf{B}^{(\pm\alpha)}_{\mathrm{quad}}\dot{\mathbf{v}} = \mathbf{B}_{\mathrm{quad}}\dot{\mathbf{v}}, \tag{9.18}$$

where

$$\mathbf{B}^{(\pm\alpha)}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, b^{(\pm\alpha)}_{\mathrm{pol}}\varphi_{\mathbf{k}})_{\Omega^{(\pm\alpha)}}, \qquad \alpha = 1, \ldots, d,$$

and $\mathbf{B}^{(\pm\alpha)}_{\mathrm{quad}}\dot{\mathbf{v}}$ is computed as in (9.17) with $b^{(\pm\alpha)}_{\mathrm{pol}}$ in place of $q^{(\pm\alpha)}$. A fast evaluation procedure for matrix-vector products with $b^{(\pm\alpha)}_{\mathrm{pol}}\left(\mathbf{X}^{(\neg\alpha)}\right)$ has been given in Section 9.2.

9.5 A brief note on complexity

To sum up, using (9.8) and (9.18) in (8.10), the system of ODEs we actually solve reads

$$\ddot{\mathbf{c}} = -\mathbf{S}^{\mathrm{quad}}\mathbf{c} - \mathbf{B}^{\mathrm{quad}}\dot{\mathbf{c}} + \mathbf{f} + \mathbf{g}.$$

The matrix-vector products (9.8) and (9.18) are computed using the efficient procedures devised in this chapter. If all $a^{(\alpha,\beta)}$ are approximated using a Chebyshev polynomial tensor product expansion with polynomials indexed over a set $\mathcal{R}(d, R)$,

$$a^{(\alpha,\beta)}(x) \approx a^{(\alpha,\beta)}_{\mathrm{pol}}(x) = \sum_{\mathbf{r}\in\mathcal{R}} \hat{a}^{(\alpha,\beta)}_{\mathbf{r}} \prod_{\gamma=1}^{d} T_{r_\gamma}(x_\gamma), \qquad \alpha, \beta = 1, \ldots, d,$$

and if b is approximated analogously using an index set $\mathcal{R}(d\text{–}1, R)$, these procedures require

$$\sim d^2\big(|\mathcal{R}(d, R)| + 2\big)|\mathcal{K}(d, K)|\big) \tag{9.19}$$

and

$$\sim 2d\big(|\mathcal{K}(d, K)| + |\mathcal{R}(d-1, R)||\mathcal{K}(d-1, K)|\big) \tag{9.20}$$

operations, respectively. The number of operations given in (9.19) is due to the fact that there are d^2 coefficient functions $a^{(\alpha,\beta)}$, and each one requires a single application of the linearly scaling coefficient procedure given in Section 9.2, which explains the factor $|\mathcal{R}(d,R)|$, and a single application of the linearly scaling Algorithms 12 and 13, which explains the factor 2. The costs given in (9.20) are due to there being $2d$ boundary faces, where each boundary face requires a single computation of $\mathbf{v}^{(\neg\alpha,\pm)}$, which explains the factor $|\mathcal{K}(d,K)|$, and an application of the fast algorithm given in Section 9.2 in $d-1$ dimensions, which explains the factor $|\mathcal{R}(d{-}1,R)||\mathcal{K}(d{-}1,K)$. If we choose a different number of interpolation nodes for different coefficient functions, the expressions (9.19) and (9.20) become more complicated, but in a straightforward way.

10 Error analysis

The following error analysis parallels the proceeding from Chapter 4, Sections 4.1–4.3. There are three sources of error: First, there is an error due to polynomial approximation of the coefficient functions. Again, we opt for Chebyshev interpolation. Second, we truncate the Galerkin basis. Third, we approximate the stiffness and boundary term matrices by entrywise Gauß–Legendre quadrature, the latter being equivalent to the fast matrix-vector product procedures given in Chapter 9.

In Section 10.1, we give an outline of the subsequent error analysis including the main result. As before, the interpolation error is dealt with separately in Section 10.2, while the truncation plus quadrature errors are treated in a combined analysis in Section 10.4. The intermediate Section 10.3 contains a stability analysis for the spatially discrete system of ODEs as we actually solve it.

As we restrict our attention to the case of a full index cube, there is no error due to index set reduction reduction. We shall comment on our experiences with reduced index sets in Chapter 11.

In this chapter, we omit time-dependency of functions or vectors whenever it is clear from the context. The constants C and κ denote general constants, which may adopt different values at different occurrences. For matrix-valued arguments, the Sobolev norm is understood as the Frobenius norm of the Sobolev norm of the matrix elements.

10.1 Outline and main results

We start with some notational conventions. Let

$$\begin{aligned} a_{\mathrm{pol}}^{(\alpha,\beta)} &: \Omega \to \mathbb{R}, \qquad \alpha,\beta = 1,\dots,d, \\ b^{\mathrm{pol}} &: \partial\Omega \to \mathbb{R} \end{aligned}$$

denote polynomial approximations to the coefficient functions $a^{(\alpha,\beta)}$ and b as they occur in the original problem (8.1)–(8.4). For the sake of simplicity, we assume them all to be given as a sum of separable polynomials where the sum ranges over a common index set $\mathcal{R}(d,R)$ for all $a_{\mathrm{pol}}^{(\alpha,\beta)}$ (and over $\mathcal{R}(d-1,R)$ for b^{pol}) such that

$$|\mathcal{K}(d,K)| \gg |\mathcal{R}(d,R)|, \qquad |\mathcal{K}(d-1,K)| \gg |\mathcal{R}(d-1,R)|, \qquad K \gg R.$$

The index set $\mathcal{K}(d,K) = \mathcal{K}_{\mathrm{full}}(d,K)$ is the full index cube that underlies the Galerkin approximation to the original problem. We set $a^{\mathrm{pol}} = (a_{\mathrm{pol}}^{(\alpha,\beta)})_{\alpha,\beta=1}^{d}$.

To facilitate the error analysis, we introduce the Legendre expansions

$$\psi(x,t) = \sum_{\mathbf{k}\in\mathbb{N}^d} u_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x), \qquad \psi^{\mathrm{pol}}(x,t) = \sum_{\mathbf{k}\in\mathbb{N}^d} u^{\mathrm{pol}}_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x),$$

as well as the truncated Legendre expansion

$$\psi_{\mathcal{K}}(x,t) = \sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}(t)\varphi_{\mathbf{k}}(x).$$

The coefficients $u_{\mathbf{k}}$ are chosen such that ψ solves the original problem (8.1)–(8.4). In particular, ψ satisfies the weakly formulated problem (omitting unnecessary arguments)

$$\begin{aligned}(\varphi,\psi_{tt}) = &-(\nabla\varphi, a\nabla\psi)\\ &-(\varphi, b\psi_t)_{\partial\Omega} + (\varphi,f) + (\varphi,g)_{\partial\Omega}\end{aligned} \qquad \forall\,\varphi\in H^1(\Omega), \tag{10.1}$$

with the initial data $\psi(x,0)=\psi_1(x)$ and $\psi_t(x,0)=\psi_2(x)$.

Analogously, ψ^{pol} is the solution to the same problem, but with $a^{(\alpha,\beta)}$ replaced by $a^{(\alpha,\beta)}_{\mathrm{pol}}$ and with b replaced by b_{pol}. In particular, ψ^{pol} satisfies the weakly formulated problem

$$\begin{aligned}(\varphi,\psi^{\mathrm{pol}}_{tt}) = &-(\nabla\varphi, a^{\mathrm{pol}}\nabla\psi^{\mathrm{pol}})\\ &-(\varphi, b^{\mathrm{pol}}\psi^{\mathrm{pol}}_t)_{\partial\Omega} + (\varphi,f) + (\varphi,g)_{\partial\Omega}\end{aligned} \qquad \forall\,\varphi\in H^1(\Omega), \tag{10.2}$$

with the same initial data $\psi^{\mathrm{pol}}(x,0)=\psi_1(x)$ and $\psi^{\mathrm{pol}}_t(x,0)=\psi_2(x)$.

The function $\psi_{\mathcal{K}}$ is the Galerkin approximation to ψ^{pol}, where $(K{+}1)$-nodes full-product Gauß–Legendre quadrature has been taken into account for the stiffness and boundary term matrices,

$$\begin{aligned}(\varphi_{\mathbf{j}},(\psi_{\mathcal{K}})_{tt}) = &-(\nabla\varphi_{\mathbf{j}}, a^{\mathrm{pol}}\nabla\psi_{\mathcal{K}})_{\mathrm{quad}}\\ &-(\varphi_{\mathbf{j}}, b^{\mathrm{pol}}(\psi_{\mathcal{K}})_t)_{\partial\Omega,\mathrm{quad}} + (\varphi_{\mathbf{j}},f) + (\varphi_{\mathbf{j}},g)_{\partial\Omega}\end{aligned} \qquad \forall\,\mathbf{j}\in\mathcal{K}, \tag{10.3}$$

or, equivalently,

$$\ddot{\mathbf{c}} = -\mathbf{S}^{\mathrm{quad}}\mathbf{c} - \mathbf{B}^{\mathrm{quad}}\dot{\mathbf{c}} + \mathbf{f} + \mathbf{g}, \tag{10.4}$$

with the stiffness and boundary term matrices

$$\begin{aligned}\mathbf{S}^{\mathrm{quad}}_{\mathbf{jk}} &= \sum_{\alpha,\beta=1}^{d} (\mathbf{S}^{(\alpha,\beta)}_{\mathrm{quad}})_{\mathbf{j},\mathbf{k}} = \sum_{\alpha,\beta=1}^{d} \Big(\frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{j}}, a^{(\alpha,\beta)}_{\mathrm{pol}}\frac{\partial}{\partial x_\beta}\varphi_{\mathbf{k}}\Big)_{\mathrm{quad}}, \qquad \mathbf{j},\,\mathbf{k}\in\mathcal{K},\\ \mathbf{B}^{\mathrm{quad}}_{\mathbf{jk}} &= (\varphi_{\mathbf{j}}, b^{\mathrm{pol}}\varphi_{\mathbf{k}})_{\partial\Omega,\mathrm{quad}},\end{aligned}$$

respectively, and vectors

$$\mathbf{f}_{\mathbf{j}} = (\varphi_{\mathbf{j}},f), \quad \mathbf{g}_{\mathbf{j}} = (\varphi_{\mathbf{j}},g)_{\partial\Omega}, \qquad \mathbf{j}\in\mathcal{K}. \tag{10.5}$$

As an initialization, we use the $L^2(\Omega)$-orthogonal projections of ψ_1 and ψ_2, i.e., $\mathcal{P}_{\mathcal{K}}\psi_1$ and $\mathcal{P}_{\mathcal{K}}\psi_2$, respectively.

Equation (10.4) is the semidiscrete problem as we actually propagate it in time. We show that the method devised in this paper is spectrally accurate with respect to the energy seminorm (8.7). Our overall aim is to estimate the spatial error

$$\|\psi(\cdot,t)-\psi_{\mathcal{K}}(\cdot,t)\|_E, \qquad t\in[0,T].$$

We do this using the above error decomposition. The error due to polynomial interpolation of the coefficient functions can be bounded in terms of the interpolation errors and of the exact solution using energy estimates. Following standard theory, Section 10.2 contains an analysis of this error due to replacing the coefficient functions by their polynomial counterparts before discretizing the equation in space. In Section 10.3, we show stability of the spatially discrete system (10.4). In Section 10.4, we give an estimate for the combined error due to Galerkin approximation (i.e., truncation of the basis) and entrywise quadrature for the right-hand side matrices. The error is bounded in terms of the exact solution using energy estimates in combination with the above standard Legendre approximation result (8.14) and an appropriate projection which is related to the exactness of Gaussian quadrature; for the latter technique see also the above proof of Lemma 4 from Section 4.3.

Putting everything together then yields the following overall error result:

Theorem 7. (error due to interpolation and spatial discretization)
For the solution $\psi(\cdot,t)$ of (8.1)–(8.4), assume that $\psi(\cdot,t)\in H^s(\Omega)$ and $\psi_t(\cdot,t)\in H^{s-1}(\Omega)$ for all $t\in[0,T]$, with $s\geq 4$. Assume also that the degree R interpolants a^{pol} and b^{pol} of a and b satisfy

$$\|a-a^{\mathrm{pol}}\|_{H^1(\Omega)}\leq\varepsilon, \qquad \|b-b^{\mathrm{pol}}\|_{L^2(\partial\Omega)}\leq\varepsilon,$$

with $R\leq K/2$. Then, the error of the solution $\psi_{\mathcal{K}}(\cdot,t)$ of the semidiscretized problem (10.3) is bounded by

$$\|\psi(\cdot,t)-\psi_{\mathcal{K}}(\cdot,t)\|_E^2\leq C\big(C_2\varepsilon^2+C_sK^{-2s+7}\big) \qquad \forall\, t\in[0,T],$$

where $C=C(d,a,b,\Omega,T)$ is independent of ψ, K, and R, and depends linearly on T, and where

$$C_r=\int_0^t\big(\|\psi(\cdot,\tau)\|_{H^r(\Omega)}^2+\|\psi_t(\cdot,\tau)\|_{H^{r-1}(\Omega)}^2\big)\,\mathrm{d}\tau,$$

with $r=2,s$.

10.2 Interpolation error

Before dealing with the error due to discretization in space, we consider the error due to replacing the coefficients a and b by their polynomial counterparts a^{pol} and b^{pol} still on the continuous level with respect to the energy norm (8.7). Since the proof employs again the energy method given in [40], we can keep the presentation brief and refer back to the proof of strong well-posedness of the continuous solution as done in Lemma 8.

Lemma 10. (interpolation error)
If $\psi(\cdot,t) \in H^2(\Omega)$ is the solution at time t of (8.1)–(8.4), $\psi_t(\cdot,t) \in H^1(\Omega)$, and if $\psi^{\mathrm{pol}}(\cdot,t) \in H^1(\Omega)$ is the solution at time t of the same problem with the coefficients a and b replaced with their polynomial approximants a^{pol} and b^{pol}, the error due to interpolation is given by

$$\|\psi(\cdot,t) - \psi^{\mathrm{pol}}(\cdot,t)\|_E^2 \le C \int_0^t \big(\|a - a^{\mathrm{pol}}\|_{H^1(\Omega)}^2 \|\psi(\cdot,\tau)\|_{H^2(\Omega)}^2 + \\ + \|b - b^{\mathrm{pol}}\|_{L^2(\partial\Omega)}^2 \|\psi_t(\cdot,\tau)\|_{H^1(\Omega)}^2 \big)\, \mathrm{d}\tau$$

for all $t \in [0,T]$, where $C = C(\Omega,T)$ is independent of ψ and depends linearly on T.

Proof. Subtracting the weak equations (10.1) and (10.2) shows that the error

$$e^{\mathrm{pol}} = u - u^{\mathrm{pol}}$$

satisfies the error equation

$$\begin{aligned}(\varphi, e_{tt}^{\mathrm{pol}}) = &-(\nabla\varphi, a^{\mathrm{pol}} \nabla e^{\mathrm{pol}}) \\ &- (\varphi, b^{\mathrm{pol}} e_t^{\mathrm{pol}})_{\partial\Omega} + (\varphi, f^{\mathrm{pol}}) + (\varphi, g^{\mathrm{pol}})_{\partial\Omega}\end{aligned} \qquad \forall\, \varphi \in H^1(\Omega), \tag{10.6}$$

with $e^{\mathrm{pol}}(x,0) = e_t^{\mathrm{pol}}(x,0) = 0$ due to the choice of initializations, and forcings

$$f^{\mathrm{pol}} = \nabla \cdot (a - a^{\mathrm{pol}}) \nabla\psi, \qquad g^{\mathrm{pol}} = -\mathbf{n} \cdot (a - a^{\mathrm{pol}}) \nabla\psi - (b - b^{\mathrm{pol}}) \psi_t,$$

where we have used integration by parts. Since (10.6) is of the same form as (10.1), it satisfies a similar energy estimate, viz.,

$$\|e^{\mathrm{pol}}(\cdot,t)\|_E^2 \le \kappa(T) \int_0^t (\|f^{\mathrm{pol}}(\cdot,\tau)\|^2 + \|g^{\mathrm{pol}}(\cdot,\tau)\|_{\partial\Omega}^2)\, \mathrm{d}\tau$$

for all $t \in [0,T]$, where $\kappa(T)$ depends linearly on T. We can thereby bound e^{pol} in terms of the interpolation errors $a - a^{\mathrm{pol}}$ and $b - b^{\mathrm{pol}}$, and of the exact solution ψ.

The rest of the proof is manipulations of Sobolev norms in order to bound f^{pol} and g^{pol}. Writing out

$$f^{\mathrm{pol}} = \sum_{j,k=1}^{d} (a - a^{\mathrm{pol}})_{jk} \partial_{j,k}^2 u + \sum_{j,k=1}^{d} (\partial_j (a - a^{\mathrm{pol}})_{jk})(\partial_k u),$$

and using

$$\begin{aligned}\sum_{j,k=1}^{d} (a - a^{\mathrm{pol}})_{jk} \partial_{j,k}^2 u &\le \Big(\sum_{j,k=1}^{d} |a - a^{\mathrm{pol}}|_{jk}^2 \Big)^{1/2} \Big(\sum_{j,k=1}^{d} |\partial_{j,k}^2 u|^2 \Big)^{1/2} \\ &= \|a - a^{\mathrm{pol}}\| |u|_{H^2(\Omega)}, \\ \sum_{j,k=1}^{d} (\partial_j (a - a^{\mathrm{pol}})_{jk})(\partial_k u) &\le \Big\| \sum_{j=1}^{d} \partial_j (a - a^{\mathrm{pol}})_{j,\cdot} \Big\| |u|_{H^1(\Omega)} \\ &\le \Big(2 \sum_{j,k=1}^{d} \|\partial_j (a - a^{\mathrm{pol}})_{jk}\|^2 \Big)^{1/2} |u|_{H^1(\Omega)} \\ &\le \sqrt{2} |a - a^{\mathrm{pol}}|_{H^1(\Omega)} |u|_{H^1(\Omega)},\end{aligned}$$

we find

$$\|f^{\mathrm{pol}}\| \le |a - a^{\mathrm{pol}}|_{H^1(\Omega)}|u|_{H^1(\Omega)} + \|a - a^{\mathrm{pol}}\||u|_{H^2(\Omega)}.$$

By the trace inequality,

$$\|u\|^2_{L^2(\partial\Omega)} \le C\|u\|^2_{H^1(\Omega)},$$

we can bound g^{pol} as

$$\begin{aligned}\|g^{\mathrm{pol}}\|_{\partial\Omega} &\le \|\mathbf{n}\cdot(a - a^{\mathrm{pol}})\nabla u\|_{\partial\Omega} + \|(b - b^{\mathrm{pol}})u_t\|_{\partial\Omega}\\ &\le \|(a - a^{\mathrm{pol}})\|_{\partial\Omega}\|\nabla u\|_{\partial\Omega} + \|b - b^{\mathrm{pol}}\|_{\partial\Omega}\|u_t\|_{\partial\Omega}\\ &\le C(\|a - a^{\mathrm{pol}}\|_{H^1(\Omega)}\|u\|_{H^2(\Omega)} + \|b - b^{\mathrm{pol}}\|_{L^2(\partial\Omega)}\|u_t\|_{H^1(\Omega)}).\end{aligned}$$

Putting the bounds together yields the desired result. □

The interpolation errors $a - a^{\mathrm{pol}}$ and $b - b^{\mathrm{pol}}$ can be bounded in terms of the order R of the interpolants and of the derivatives of a and b using standard theory; see [15]. For smooth coefficients, the error decays faster than any polynomial in R.

10.3 Stability of spatial semidiscretization

The spatial semidiscretization given in (10.4) is constructed such that the resulting system of ODEs satisfies a similar energy estimate as the PDE. A semidiscretization with such a property is called (strongly) stable; see [40]. Strong stability of the semidiscretization (10.4) with respect to a discrete analog of the seminorm (8.7), viz.,

$$\|\mathbf{c}\|^2_{\mathbf{E}} = \|\dot{\mathbf{c}}\|^2 + \mathbf{c}^T\mathbf{S}^{\mathrm{quad}}\mathbf{c}, \tag{10.7}$$

can be established using similar arguments. Before giving a proof, we have to show that (10.7) is actually a seminorm. If $\det(a_{\mathrm{pol}}(x)) \ge a_0 > 0$, the inner matrix in

$$\mathbf{S}^{\mathrm{quad}} = \begin{pmatrix}\mathbf{U}^{(1)}\\ \vdots\\ \mathbf{U}^{(d)}\end{pmatrix}^T \begin{pmatrix}[a^{(1,1)}_{\mathrm{pol}}] & \cdots & [a^{(1,d)}_{\mathrm{pol}}]\\ \vdots & \ddots & \vdots\\ [a^{(d,1)}_{\mathrm{pol}}] & \cdots & [a^{(d,d)}_{\mathrm{pol}}]\end{pmatrix} \begin{pmatrix}\mathbf{U}^{(1)}\\ \vdots\\ \mathbf{U}^{(d)}\end{pmatrix} \tag{10.8}$$

is positive definite, where we use the abbreviation

$$[a^{(\alpha,\beta)}_{\mathrm{pol}}] = \mathrm{diag}_{\mathbf{k}\in\mathcal{K}}(a^{(\alpha,\beta)}_{\mathrm{pol}}(\xi_{\mathbf{k}})), \qquad \alpha,\beta = 1,\dots,d.$$

Then, $\mathbf{S}^{\mathrm{quad}}$ is positive semidefinite, which implies that $\|\cdot\|_{\mathbf{E}}$ is a seminorm.

Lemma 11. (strong stability of spatial discretization)
Provided $\det(a^{\mathrm{pol}}(x)) \ge a_0 > 0$ *and* $b^{\mathrm{pol}}(x) \ge b_0 > 0$ *for all* x*, the semidiscretization* (10.4) *is strongly stable in the discrete seminorm* (10.7)*.*

Proof. If $b^{\mathrm{pol}}(x) \ge b_0 > 0$, $\mathbf{B}$ is positive semidefinite, and positive definite with respect to the boundary norm. We now prove that this property transfers to $\mathbf{B}_{\mathrm{quad}}$. Consider an

arbitrary $\mathbf{v} = (v_\mathbf{k})_{\mathbf{k}\in\mathcal{K}}$ and set $\chi = \sum_{\mathbf{k}\in\mathcal{K}} v_\mathbf{k}\varphi_\mathbf{k}$. Recall the notation from Section 9.4. For any boundary face $\Omega^{(\pm\alpha)}$, we find

$$\begin{aligned}
\mathbf{v}^T\mathbf{B}^{(\pm\alpha)}_{\text{quad}}\mathbf{v} &= \sum_{\mathbf{k}\in\mathcal{K}} v_\mathbf{k}(\mathbf{B}^{(\pm\alpha)}_{\text{quad}}\mathbf{v})_\mathbf{k} \\
&= \sum_{\mathbf{k}\in\mathcal{K}} v_\mathbf{k}\varphi_{k_\alpha}(\pm 1)(b^{(\pm\alpha)}_{\text{pol}}(\mathbf{X}^{(\neg\alpha)})\mathbf{v}^{(\neg\alpha,\pm)})_{\mathbf{k}^{(\neg\alpha)}} \\
&= \sum_{\mathbf{k}^{(\neg\alpha)}\in\mathcal{K}(d\text{–}1,K)} \big(\sum_{k_\alpha} v_\mathbf{k}\varphi_{k_\alpha}(\pm 1)\big)\big(b^{(\pm\alpha)}_{\text{pol}}(\mathbf{X}^{(\neg\alpha)})\mathbf{v}^{(\neg\alpha,\pm)}\big)_{\mathbf{k}^{(\neg\alpha)}\in\mathcal{K}(d\text{–}1,K)} \\
&= (\mathbf{v}^{(\neg\alpha,\pm)})^T b^{(\pm\alpha)}_{\text{pol}}(\mathbf{X}^{(\neg\alpha)})\mathbf{v}^{(\neg\alpha,\pm)} \\
&= \sum_{\mathbf{j}\in\mathcal{K}(d\text{–}1,K)} \mathbf{v}^{(\neg\alpha,\pm)}_\mathbf{j} \sum_{\mathbf{k}\in\mathcal{K}(d\text{–}1,K)} \big(\sum_{\mathbf{m}\in\mathcal{K}(d\text{–}1,K)} w_\mathbf{m}\varphi_\mathbf{j}(\xi_\mathbf{m})b^{(\pm\alpha)}_{\text{pol}}(\xi_\mathbf{m})\varphi_\mathbf{k}(\xi_\mathbf{m})\big)\mathbf{v}^{(\neg\alpha,\pm)}_\mathbf{k} \\
&= \sum_\mathbf{m} \big(\sum_\mathbf{j} \mathbf{v}^{(\neg\alpha,\pm)}_\mathbf{j}\varphi_\mathbf{j}(\xi_\mathbf{m})\big)\big(\sum_\mathbf{k} \mathbf{v}^{(\neg\alpha,\pm)}_\mathbf{k}\varphi_\mathbf{k}(\xi_\mathbf{m})\big)b^{(\pm\alpha)}_{\text{pol}}(\xi_\mathbf{m})w_\mathbf{m} \\
&\geq b_0\|v\|^2_{\Omega^{(\pm\alpha)}},
\end{aligned}$$

where we have used the equivalence of matrix insertion and quadrature, and the fact that $b^{(\pm\alpha)}_{\text{pol}}$ is positive. After multiplication of (10.4) by $\dot{\mathbf{c}}^T$ from the left, the rest is then simply a discrete analog of the proof of stability for the continuous counterpart; cf. Lemma 8. In particular, we can complete the square using the positivity results established above. We eventually get the estimate

$$\|\mathbf{c}(t)\|^2_\mathbf{E} \leq \kappa(T)\Big(\|\mathbf{c}(0)\|^2_\mathbf{E} + \int_0^t (\|f(\cdot,\tau)\|^2 + \|g(\cdot,\tau)\|^2_{\partial\Omega})\,\mathrm{d}\tau\Big) \qquad \forall\, t \in [0,T],$$

with $\kappa = \mathrm{e}\max(1, 1/2b_0, T)$. □

10.4 Spatial discretization

In this section, we study the error when comparing the exact solution to the semidiscretization, approximation of the matrix elements by quadrature taken into account. For the sake of simplicity, we drop all subscripts or superscripts "pol" throughout this section and let a and b refer to the polynomial approximants of the coefficients.

In the error estimates given in Part I of this thesis, as a starting point of our analysis, we could profit from self-adjointness of the Hermitian operators under consideration and the fact that taking the real part in the error equation thus made certain terms vanish. Now, since there are only real quantities involved, energy estimates are the backbone of the analysis. The very ansatz of a suitable error decomposition is the same as in Part I, though; see [86].

Let $\mathcal{P}_\mathcal{K}$ and $\mathcal{P}^\perp_\mathcal{K}$ be the $L^2(\Omega)$-orthogonal projection onto the polynomial approximation space $\operatorname{span}\{\varphi_\mathbf{j}\,;\,\mathbf{j}\in\mathcal{K}\}$ and its orthogonal complement, respectively. The exact solution

$$\psi(x,t) = \sum_{\mathbf{k}\in\mathbb{N}^d} u_\mathbf{k}(t)\varphi_\mathbf{k}(x)$$

of (8.1)–(8.4) satisfies, in particular,

$$\begin{aligned}(\varphi_{\mathbf{j}}, \mathcal{P}_{\mathcal{K}}\psi_{tt}) = &-(\nabla\varphi_{\mathbf{j}}, a\nabla\mathcal{P}_{\mathcal{K}}\psi) \\ &- (\varphi_{\mathbf{j}}, b\mathcal{P}_{\mathcal{K}}\psi_t)_{\partial\Omega} + (\varphi_{\mathbf{j}}, f + f_{\mathcal{K}}) + (\varphi_{\mathbf{j}}, g + g_{\mathcal{K}})_{\partial\Omega},\end{aligned} \qquad \mathbf{j} \in \mathcal{K},$$

with the internal and boundary defects

$$f_{\mathcal{K}} = \nabla \cdot a\nabla\mathcal{P}_{\mathcal{K}}^{\perp}\psi, \qquad g_{\mathcal{K}} = -\mathbf{n} \cdot a\nabla\mathcal{P}_{\mathcal{K}}^{\perp}\psi - b\mathcal{P}_{\mathcal{K}}^{\perp}\psi_t.$$

If we let $\mathbf{u} = (u_{\mathbf{k}})_{\mathbf{k}\in\mathcal{K}}$, $\mathbf{f}_{\mathcal{K}} = (f_{\mathcal{K},\mathbf{j}})_{\mathbf{j}\in\mathcal{K}}$, $\mathbf{g}_{\mathcal{K}} = (g_{\mathcal{K},\mathbf{j}})_{\mathbf{j}\in\mathcal{K}}$, where

$$f_{\mathcal{K},\mathbf{j}} = (\varphi_{\mathbf{j}}, f_{\mathcal{K}}), \qquad g_{\mathcal{K},\mathbf{j}} = (\varphi_{\mathbf{j}}, g_{\mathcal{K}})_{\partial\Omega}, \tag{10.9}$$

this can equivalently be written as

$$\ddot{\mathbf{u}} = -\mathbf{S}\mathbf{u} - \mathbf{B}\dot{\mathbf{u}} + \mathbf{f} + \mathbf{g} + \mathbf{f}_{\mathcal{K}} + \mathbf{g}_{\mathcal{K}}. \tag{10.10}$$

Similarly, let $\mathbf{c} = (c_{\mathbf{k}})_{\mathbf{k}\in\mathcal{K}}$ be the solution of the semidiscrete system (10.4). Repeating the strategy used for the proof of Theorem 3 from Section 4.3, we decompose the error according to

$$\psi - \sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}\varphi_{\mathbf{k}} = \underbrace{\big(\psi - \mathcal{P}_{\mathcal{K}}\psi\big)}_{=\rho} + \underbrace{\big(\mathcal{P}_{\mathcal{K}}\psi - \sum_{\mathbf{k}\in\mathcal{K}} c_{\mathbf{k}}\varphi_{\mathbf{k}}\big)}_{=\theta},$$

where ρ can be bounded using the standard projection estimate (8.14). The error θ or, equivalently, $\mathbf{u} - \mathbf{c}$, is bounded in the below Theorem 8. The proof is based on standard energy estimates in combination with a suitable projection matrix which is related to the exactness properties of Gaussian quadrature. This kind of projection matrix has already been used in the proof of Lemma 4 from Section 4.3.

Theorem 8. (global error of spatial discretization)
Let $\psi(\cdot,t)$ be the solution at time t of (8.1)–(8.4) with the coefficient functions replaced with their degree R polynomial interpolants, and $\mathcal{P}_{\mathcal{K}}\psi(\cdot,t) = \sum_{\boldsymbol{k}\in\mathcal{K}} u_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}$ be the orthogonal projection of $\psi(\cdot,t)$ onto the polynomial approximation space $\mathrm{span}\{\varphi_{\boldsymbol{j}}; \boldsymbol{j} \in \mathcal{K}\}$, with $\mathcal{K} = \mathcal{K}(d,K)$ being the full index cube. Denote $\boldsymbol{u}(t) = (u_{\boldsymbol{k}}(t))_{\boldsymbol{k}\in\mathcal{K}}$, and let $\mathbf{c}(t)$ be the solution at time t of the semidiscrete approximation (10.4). Then,

$$\|\boldsymbol{u}(t) - \mathbf{c}(t)\|_{\boldsymbol{E}}^2 \le C(K-R)^{-2s} \int_0^t \Big(K^7 |u(\cdot,\tau)|_{H^s(\Omega)}^2 + (K-R)^5 |u_t(\cdot,\tau)|_{H^{s-1}(\Omega)}^2 \Big)\, \mathrm{d}\tau$$

for all $t \in [0,T]$, where $C = C(d,a,b,\Omega,T)$ is independent of ψ, K, and R, and depends linearly on T.

Proof. We set $\mathbf{e} = \mathbf{u} - \mathbf{w}$ and subtract (10.4) from (10.10) to obtain

$$\ddot{\mathbf{e}} = -\mathbf{S}^{\mathrm{quad}}\mathbf{e} - \mathbf{B}^{\mathrm{quad}}\dot{\mathbf{e}} + \mathbf{f}^{\mathrm{quad}} + \mathbf{f}_{\mathcal{K}} + \mathbf{g}^{\mathrm{quad}} + \mathbf{g}_{\mathcal{K}},$$

where

$$\mathbf{f}^{\mathrm{quad}} = -(\mathbf{S} - \mathbf{S}^{\mathrm{quad}})\mathbf{u}, \qquad \mathbf{g}^{\mathrm{quad}} = -(\mathbf{B} - \mathbf{B}^{\mathrm{quad}})\dot{\mathbf{u}}.$$

We can bound the error $\mathbf{e}$ using energy estimates of the same form as before. Differentiating the discrete seminorm (10.7) of $\mathbf{e}$ with respect to time yields

$$\begin{aligned}\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\|\mathbf{e}\|_{\mathbf{E}}^2 &= \dot{\mathbf{e}}^T(\mathbf{f}^{\mathrm{quad}}+\mathbf{f}_{\mathcal{K}}) - \dot{\mathbf{e}}^T\mathbf{B}^{\mathrm{quad}}\dot{\mathbf{e}} + \dot{\mathbf{e}}^T(\mathbf{g}^{\mathrm{quad}}+\mathbf{g}_{\mathcal{K}}), \qquad (10.11)\\ \frac{\mathrm{d}}{\mathrm{d}t}\|\mathbf{e}\|_{\mathbf{E}}^2 &\le \frac{1}{T}\|\dot{\mathbf{e}}\|^2 + T\|\mathbf{f}^{\mathrm{quad}}+\mathbf{f}_{\mathcal{K}}\|^2 - 2\dot{\mathbf{e}}^T\mathbf{B}^{\mathrm{quad}}\dot{\mathbf{e}} + 2\dot{\mathbf{e}}^T(\mathbf{g}^{\mathrm{quad}}+\mathbf{g}_{\mathcal{K}})\\ &\le \frac{1}{T}\|\mathbf{e}\|_{\mathbf{E}}^2 + 2T\|\mathbf{f}^{\mathrm{quad}}\|^2 + 2T\|\mathbf{f}_{\mathcal{K}}\|^2 - 2\dot{\mathbf{e}}^T\mathbf{B}^{\mathrm{quad}}\dot{\mathbf{e}} + 2\dot{\mathbf{e}}^T(\mathbf{g}^{\mathrm{quad}}+\mathbf{g}_{\mathcal{K}})\end{aligned}$$

for all $t \in [0,T]$. We first treat the contributions from the internal forcings $\mathbf{f}^{\mathrm{quad}}$ and $\mathbf{f}_{\mathcal{K}}$, and then turn to the boundary forcings $\mathbf{g}^{\mathrm{quad}}$ and $\mathbf{g}_{\mathcal{K}}$.

Controlling the internal forcing $\mathbf{f}^{\mathrm{quad}}$ is done using the same line of arguments as in the proof of Lemma 4 from Section 4.3. We start with a suitable projection matrix that is related to the exactness properties of Gauß–Legendre quadrature. We get a non-vanishing quadrature error only in the elements $(\mathbf{j},\mathbf{k})$ of $\mathbf{S}-\mathbf{S}^{\mathrm{quad}}$ and $\mathbf{B}-\mathbf{B}^{\mathrm{quad}}$ with $|\mathbf{j}+\mathbf{k}|_\infty = \max(\mathbf{j}+\mathbf{k}) > 2K+1-R$. This yields again the diagonal projection matrix $\mathbf{P}$ as $\mathbf{P}_{\mathbf{jj}} = 1$ for $|\mathbf{j}|_\infty > K+1-R$, and $\mathbf{P}_{\mathbf{jj}} = 0$ otherwise. Then,

$$\mathbf{S}-\mathbf{S}^{\mathrm{quad}} = \mathbf{P}(\mathbf{S}-\mathbf{S}^{\mathrm{quad}})\mathbf{P},$$

and

$$\|\mathbf{f}^{\mathrm{quad}}\| = \|(\mathbf{S}-\mathbf{S}^{\mathrm{quad}})\mathbf{u}\| \le (\|\mathbf{S}\| + \|\mathbf{S}^{\mathrm{quad}}\|)\|\mathbf{P}\mathbf{u}\|, \qquad (10.12)$$

where the matrix norm is the spectral matrix norm. Again, by the projection estimate (8.14),

$$\|\mathbf{P}\mathbf{u}\| \le C(K-R)^{-s}|\psi|_{H^s(\Omega)}.$$

This bound is still spectrally accurate with respect to K for $K \gg R$.

We can estimate $\|\mathbf{S}\|$ using Rayleigh quotients,

$$\begin{aligned}\|\mathbf{S}\| &= \sup_{\|\mathbf{v}\|\neq 0}\frac{\mathbf{v}^T\mathbf{S}\mathbf{v}}{\mathbf{v}^T\mathbf{v}} = \sup_{\|\mathbf{v}\|=1}\sum_{\mathbf{j},\mathbf{k}\in\mathcal{K}} v_{\mathbf{j}}\mathbf{S}_{\mathbf{jk}}v_{\mathbf{k}}\\ &= \sup_{\|\mathbf{v}\|=1}\Big(\Big(\sum_{\mathbf{j}\in\mathcal{K}} v_{\mathbf{j}}\nabla\varphi_{\mathbf{j}}\Big), a\Big(\sum_{\mathbf{j}\in\mathcal{K}} v_{\mathbf{j}}\nabla\varphi_{\mathbf{j}}\Big)\Big)\\ &= (\nabla\varphi_{(K,\dots,K)}, a\nabla\varphi_{(K,\dots,K)})\\ &\le a_{\max}\|\nabla\varphi_{(K,\dots,K)}\|^2 \le Cda_{\max}K^3,\end{aligned}$$

since, subject to $\|\mathbf{v}\| = 1$, the integral

$$\int_\Omega \Big(\sum_{\mathbf{j}\in\mathcal{K}} v_{\mathbf{j}}\frac{\partial}{\partial x_\alpha}\varphi_{\mathbf{j}}(x)\Big)^2\,\mathrm{d}x$$

has a (non-unique) maximum for

$$v_{\mathbf{j}} = \begin{cases}1, & \mathbf{j} = (K,\dots,K),\\ 0, & \text{otherwise},\end{cases}$$

and $\|\varphi_k'\| \le Ck^{3/2}$.

Using the factorization (10.8), the quadrature approximation to the stiffness matrix can be bounded by

$$\|\mathbf{S}^{\mathrm{quad}}\| \le a_{\max} \sum_{\alpha,\beta=1}^{d} \|\mathbf{U}^{(\alpha)}\|\|\mathbf{U}^{(\beta)}\|, \qquad a_{\max} = \max_{\alpha,\beta,\mathbf{m}} |a^{(\alpha,\beta)}(\xi_{\mathbf{m}})|.$$

By the exactness of Gaussian quadrature,

$$\|\mathbf{U}^{(\alpha)}\mathbf{v}\|^2 = \sum_{\mathbf{l}\in\mathcal{K}} \omega_{\mathbf{l}} \Big(\sum_{\mathbf{j}\in\mathcal{K}} v_{\mathbf{j}} \frac{\partial}{\partial x_\alpha} \varphi_{\mathbf{j}}(\xi_{\mathbf{l}}) \Big)^2 = \int_\Omega \Big(\sum_{\mathbf{j}\in\mathcal{K}} v_{\mathbf{j}} \frac{\partial}{\partial x_\alpha} \varphi_{\mathbf{j}}(x) \Big)^2 \,\mathrm{d}x.$$

Thereby, using the orthonormality of φ_k,

$$\|\mathbf{U}^{(\alpha)}\|^2 = \sup_{\|\mathbf{v}\|=1} \|\mathbf{U}^{(\alpha)}\mathbf{v}\|^2 = \int_\Omega \Big(\frac{\partial}{\partial x_\alpha} \varphi_{(K,\dots,K)}(x) \Big)^2 \,\mathrm{d}x = \|\varphi'_K\|^2 \le CK^3.$$

Consequently,

$$\|\mathbf{S}^{\mathrm{quad}}\| \le Cd^2 a_{\max} K^3. \tag{10.13}$$

Together with (10.12) and (10.13), this yields

$$\|\mathbf{f}^{\mathrm{quad}}\| \le Cd^2 a_{\max} K^3 (K-R)^{-s} |\psi|_{H^s(\Omega)}.$$

The norm of $\mathbf{f}_\mathcal{K}$ is bounded by

$$\begin{aligned} \|\mathbf{f}_\mathcal{K}\| \le \|f_\mathcal{K}\| &= \|\nabla \cdot a \nabla \mathcal{P}^\perp_\mathcal{K} \psi\| \le |a|_{H^1(\Omega)} |\mathcal{P}^\perp_\mathcal{K}\psi|_{H^1(\Omega)} + \|a\| |\mathcal{P}^\perp_\mathcal{K}\psi|_{H^2(\Omega)} \\ &\le CK^{7/2-s} |\psi|_{H^s(\Omega)}. \end{aligned}$$

Bounding the influence of the boundary forcings $\mathbf{g}^{\mathrm{quad}}$ and $\mathbf{g}_\mathcal{K}$ is done using similar ideas, but we need to work on the boundary faces—it is not enough to bound $\|\mathbf{g}^{\mathrm{quad}} + \mathbf{g}_\mathcal{K}\|$. Eq. (9.16), which extracts the Legendre coefficients on a boundary face, is a linear transformation from $\mathbb{R}^{|\mathcal{K}(d,K)|}$ to $\mathbb{R}^{|\mathcal{K}(d-1,K)|}$. It can therefore be written as

$$\mathbf{v}^{(\neg\alpha,\pm)} = \mathbf{H}^{(\pm\alpha)}\mathbf{u},$$

where the matrix $\mathbf{H}^{(\pm\alpha)}$ is defined implicitly through (9.16). This allows us to write $\mathbf{B}^{(\pm\alpha)}$ and $\mathbf{B}^{(\pm\alpha)}_{\mathrm{quad}}$ as

$$\mathbf{B}^{(\pm\alpha)} = (\mathbf{H}^{(\pm\alpha)})^T \widehat{\mathbf{B}}^{(\pm\alpha)} \mathbf{H}^{(\pm\alpha)}, \qquad \mathbf{B}^{(\pm\alpha)}_{\mathrm{quad}} = (\mathbf{H}^{(\pm\alpha)})^T \widehat{\mathbf{B}}^{(\pm\alpha)}_{\mathrm{quad}} \mathbf{H}^{(\pm\alpha)},$$

with the $(|\mathcal{K}(d-1,K)| \times |\mathcal{K}(d-1,K)|)$-matrices

$$\begin{aligned} \widehat{\mathbf{B}}^{(\pm\alpha)}_{\mathrm{quad}} &= \mathbf{U}^T \operatorname{diag}_{\mathbf{m}\in\mathcal{K}}(b^{(\pm\alpha)}(\xi_{\mathbf{m}}))\mathbf{U}, \\ (\widehat{\mathbf{B}}^{(\pm\alpha)})_{\mathbf{jk}} &= (\varphi_{\mathbf{j}}, b^{(\pm\alpha)}\varphi_{\mathbf{k}})_{\Omega^{(\pm\alpha)}}, \quad \mathbf{j},\mathbf{k} \in \mathcal{K}(d-1,K), \end{aligned}$$

where $\mathbf{U}$ is defined as in (9.3), but in d–1 dimensions. We decompose the boundary forcings $\mathbf{g}^{\mathrm{quad}} = \sum \mathbf{g}^{(\pm\alpha),\mathrm{quad}}$ and $\mathbf{g}_\mathcal{K} = \sum \mathbf{g}^{(\pm\alpha)}_\mathcal{K}$ into their contributions on each boundary face. We also note that for any $\mathbf{v} \in \mathbb{R}^{|\mathcal{K}|}$,

$$\begin{aligned} \mathbf{v}^T \mathbf{g}^{(\pm\alpha)}_\mathcal{K} &= \mathbf{v}^T (\mathbf{H}^{(\pm\alpha)})^T \widehat{\mathbf{g}}^{(\pm\alpha)}_\mathcal{K}, \\ \big(\widehat{g}^{(\pm\alpha)}_{\mathcal{K},\mathbf{j}}\big)_{\mathbf{j}} &= (\varphi_{\mathbf{j}}, g_\mathcal{K}|_{x_\alpha=\pm1})_{\Omega^{(\neg\alpha)}}, \quad \mathbf{j} \in \mathcal{K}(d-1,K). \end{aligned}$$

Then, the boundary terms as appearing in (10.11), on each face, read

$$\begin{aligned}\Upsilon^{(\pm\alpha)} &:= -\dot{\mathbf{e}}^T\mathbf{B}^{(\pm\alpha)}_{\text{quad}}\dot{\mathbf{e}} + \dot{\mathbf{e}}^T\mathbf{g}^{(\pm\alpha),\text{quad}} + \dot{\mathbf{e}}^T\mathbf{g}^{(\pm\alpha)}_{\mathcal{K}} \\ &= -\dot{\mathbf{e}}^T\mathbf{B}^{(\pm\alpha)}_{\text{quad}}\dot{\mathbf{e}} - \dot{\mathbf{e}}^T(\mathbf{B}^{(\pm\alpha)} - \mathbf{B}^{(\pm\alpha)}_{\text{quad}})\dot{\mathbf{u}} + \dot{\mathbf{e}}^T(\mathbf{H}^{(\pm\alpha)})^T\widehat{\mathbf{g}}^{(\pm\alpha)}_{\mathcal{K}} \\ &= -(\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}})^T\widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}}\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}} + \\ &\quad - (\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}})^T(\widehat{\mathbf{B}}^{(\pm\alpha)} - \widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}})\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{u}} + (\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}})^T\widehat{\mathbf{g}}^{(\pm\alpha)}_{\mathcal{K}}.\end{aligned}$$

The positivity condition on $b^{(\pm\alpha)}(x)$ gives $\mathbf{v}^T\widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}}\mathbf{v} \geq b_0\mathbf{v}^T\mathbf{v}$. We are then able to bound the contribution from $\mathbf{g}^{(\pm\alpha),\text{quad}}$ by completing the square,

$$\begin{aligned}\Upsilon^{(\pm\alpha)} &\leq -b_0(\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}})^T\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}} + (\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{e}})^T\big(-(\widehat{\mathbf{B}}^{(\pm\alpha)} - \widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}})\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{u}} + \widehat{\mathbf{g}}^{(\pm\alpha)}_{\mathcal{K}}\big) \\ &\leq \frac{1}{4b_0}\|(\widehat{\mathbf{B}}^{(\pm\alpha)} - \widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}})\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{u}} + \widehat{\mathbf{g}}^{(\pm\alpha)}_{\mathcal{K}}\|^2.\end{aligned}$$

Using a $(d-1)$-dimensional analog of the projection $\mathbf{P}$, we find

$$\widehat{\mathbf{B}}^{(\pm\alpha)} - \widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}} = \mathbf{P}^{(\neg\alpha)}(\widehat{\mathbf{B}}^{(\pm\alpha)} - \widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}})\mathbf{P}^{(\neg\alpha)}.$$

Together with the bound on $b(x)$ and the projection estimate (8.14), this implies

$$\begin{aligned}\|(\widehat{\mathbf{B}}^{(\pm\alpha)} - \widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}})\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{u}}\| &\leq (\|\widehat{\mathbf{B}}^{(\pm\alpha)}\| + \|\widehat{\mathbf{B}}^{(\pm\alpha)}_{\text{quad}}\|)\|\mathbf{P}^{(\neg\alpha)}\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{u}}\| \\ &\leq 2Cb_{\max}(\|\mathcal{P}^{(\neg\alpha)}\psi_t\|_{\Omega^{(\pm\alpha)}} + \|\mathcal{P}^{\perp}_{\mathcal{K}}\psi_t\|_{\Omega^{(\pm\alpha)}}).\end{aligned}$$

We add the contributions from the different boundary faces together and apply the trace inequality. This yields

$$\begin{aligned}\|(\widehat{\mathbf{B}} - \widehat{\mathbf{B}}^{\text{quad}})\sum_{\pm\alpha}\mathbf{H}^{(\pm\alpha)}\dot{\mathbf{u}}\| &\leq Cb_{\max}(\|\sum_{\pm\alpha}\mathcal{P}^{(\neg\alpha)}\psi_t\|_{\partial\Omega} + \|\mathcal{P}^{\perp}_{\mathcal{K}}\psi_t\|_{\partial\Omega}) \\ &\leq Cb_{\max}(\|\sum_{\pm\alpha}\mathcal{P}^{(\neg\alpha)}\psi_t\|_{H^1(\Omega)} + \|\mathcal{P}^{\perp}_{\mathcal{K}}\psi_t\|_{H^1(\Omega)}) \\ &\leq Cb_{\max}(K-R)^{5/2-s}|\psi_t|_{H^{s-1}(\Omega)}.\end{aligned}$$

We also have

$$\begin{aligned}\|\widehat{\mathbf{g}}_{\mathcal{K}}\| &\leq \|g_{\mathcal{K}}\|_{\partial\Omega} \leq \|\mathbf{n}\cdot a\nabla\mathcal{P}^{\perp}_{\mathcal{K}}\psi\|_{\partial\Omega} + \|b\mathcal{P}^{\perp}_{\mathcal{K}}\psi_t\|_{\partial\Omega} \\ &\leq da_{\max}\|\mathcal{P}^{\perp}_{\mathcal{K}}\psi\|_{H^1(\partial\Omega)} + b_{\max}\|\mathcal{P}^{\perp}_{\mathcal{K}}\psi_t\|_{\partial\Omega} \\ &\leq da_{\max}\|\mathcal{P}^{\perp}_{\mathcal{K}}\psi\|_{H^2(\Omega)} + b_{\max}\|\mathcal{P}^{\perp}_{\mathcal{K}}\psi_t\|_{H^1(\Omega)} \\ &\leq CK^{7/2-s}|\psi|_{H^s(\Omega)} + CK^{5/2-s}|\psi_t|_{H^{s-1}(\Omega)}.\end{aligned}$$

Thereby,

$$-\dot{\mathbf{e}}^T\mathbf{B}^{\text{quad}}\dot{\mathbf{e}} + \dot{\mathbf{e}}^T(\mathbf{g}^{\text{quad}} + \mathbf{g}_{\mathcal{K}}) \leq CK^{7-2s}|\psi|^2_{H^s(\Omega)} + C(K-R)^{5-2s}|\psi_t|^2_{H^{s-1}(\Omega)}.$$

Integrating (10.11) and putting everything together proves the lemma. □

11 Numerical experiments

In this chapter, we demonstrate our method using the above example of the wave equation in d dimensions, where $d = 2, 3$. In Section 11.1, we set up the problem by defining the wave equation in Cartesian coordinates and with an isotropic medium on a bounded d-dimensional domain. After transformation to the hypercube $[-1, 1]^d$, this problem turns out to be equivalent to the abstract setting we have considered in the previous chapters. Time propagation of the semidiscrete approximation using the classical 4th-order Runge–Kutta scheme is done in Section 11.2. In each time step, we employ the efficient procedures devised in Chapter 9. Some error results for the choice of a full index sets are given. In Section 11.3, we briefly comment on how long it takes to assemble only the stiffness matrix to illustrate how much a gain in computational costs this actually constitutes. To conclude the second part of this thesis, some comments on the applicability of our method with an additively reduced index set are given in Section 11.4.

All computations have been carried out on a desktop computer with an Intel Core 2 Duo E8400 3.00 GHz processor with 4 GB RAM using an implementation in `C`.

11.1 The acoustic wave equation

In Cartesian coordinates and with an isotropic medium, the acoustic wave equation reads

$$\begin{aligned} \psi_{tt} &= \Delta\psi, && y \in \tilde{\Omega}, \quad t \geq 0, \\ \psi(y,0) &= \psi_t(y,0) = 0, && y \in \tilde{\Omega}, \\ \tilde{b}\psi_t + \mathbf{n} \cdot \nabla\psi &= \tilde{g}(y,t), && y \in \partial\tilde{\Omega}, \end{aligned} \tag{11.1}$$

where $\tilde{\Omega} \subset \mathbb{R}^d$ is a bounded domain. We use homogeneous initial data, and excite the problem through the boundary condition. We assume that there exists a smooth coordinate transformation $x = x(y)$ which maps

$$\tilde{\Omega} \mapsto \Omega = [-1, 1]^d,$$

and that the corresponding Jacobian matrix is positive definite,

$$\mathbf{J}_{jk} = \frac{\partial y_j}{\partial x_k}, \qquad J = \det(\mathbf{J}) \geq \gamma_0 > 0 \quad \forall\, x \in \Omega.$$

We can then reformulate the problem on the hypercube Ω,

$$\begin{aligned} J\psi_{tt} &= \nabla \cdot a\nabla\psi, && x \in \Omega, \ t \geq 0, \\ \psi(x,0) &= \psi_t(x,0) = 0, && x \in \Omega, \\ b\psi_t + \mathbf{n} \cdot a\nabla\psi &= g(x,t), && x \in \partial\Omega, \end{aligned}$$

with

$$a = J\mathbf{J}^{-1}\mathbf{J}^{-T}, \qquad b = \delta\tilde{b}, \qquad g = \delta\tilde{g},$$

and $\delta = J\|\mathbf{J}^{-T}\mathbf{n}\|$. But for the presence of the function J, this problem is of the form (8.1)–(8.4) as we have discussed it throughout this part of the thesis.

When $d = 2$, we use a combination of Neumann and absorbing boundary conditions, viz.,

$$\begin{aligned} \delta\psi_t + \mathbf{n}\cdot a\nabla\psi &= 0, && \text{on } x_1 = 1,\\ \mathbf{n}\cdot a\nabla\psi &= g(x,t), && \text{on } x_1 = -1,\\ \mathbf{n}\cdot a\nabla\psi &= 0, && \text{on } x_2 = \pm 1. \end{aligned}$$

Engquist and Majda [22] describe the absorbing conditions on $x_1 = 1$. The function g is given as

$$\tilde{g}(x,t) = Mx_2 \exp\big(-20x_2^2 - 200(t-t_0)^2\big), \qquad M = 200,\ t_0 = \frac{1}{2}.$$

We solve the problem on the domain

$$\tilde{\Omega} = \{(y_1, y_2)\ :\ -1 + \phi(y_1) \le y_2 \le 1 - \phi(y_1),\ -1 \le y_1 \le 1\},$$

with $\phi(y_1) = 0.2\cos(\pi y_1)$.

For $d = 3$, we extrude the domain to $-1 \le y_3 \le 1$. We then use homogeneous Neumann boundary conditions on $x_3 = \pm 1$ as well as on $x_2 = \pm 1$. The first order Engquist–Majda boundary condition on $x_1 = 1$ has a direct generalization to three dimensions, and the boundary forcing on $x_1 = -1$ is extended as

$$\tilde{g}(x,t) = M^2 x_2 x_3 \exp\big(-20x_2^2 - 20x_3^2 - 200(t-t_0)^2\big), \qquad M = 200,\ t_0 = \frac{1}{2}.$$

With the same positivity assumptions on a and b as previously, this problem satisfies an energy estimate of the form (8.6) in the J-weighted energy norm

$$\|u\|_{E,J} = \Big((u_t, Ju_t) + (\nabla u, a\nabla u)\Big)^{1/2};$$

cf. (8.7). However, the energy estimate blows up when $b_0 \to 0$. As we have chosen $b = 0$ on parts of the boundary, we cannot show strong well-posedness for this problem using energy estimates alone. Strong well-posedness can nevertheless be proved by combining an energy estimate, which exists for $b = 0$ and $g = 0$, with the Laplace–Fourier technique, as described in [40], Chapter 10.

11.2 Time propagation

After spatial discretization, we get a system of ODEs of the form

$$J^{\text{pol}}(\mathbf{X})\ddot{\mathbf{c}} = -\mathbf{S}\mathbf{c} - \mathbf{B}\dot{\mathbf{c}} + \mathbf{g}, \tag{11.2}$$

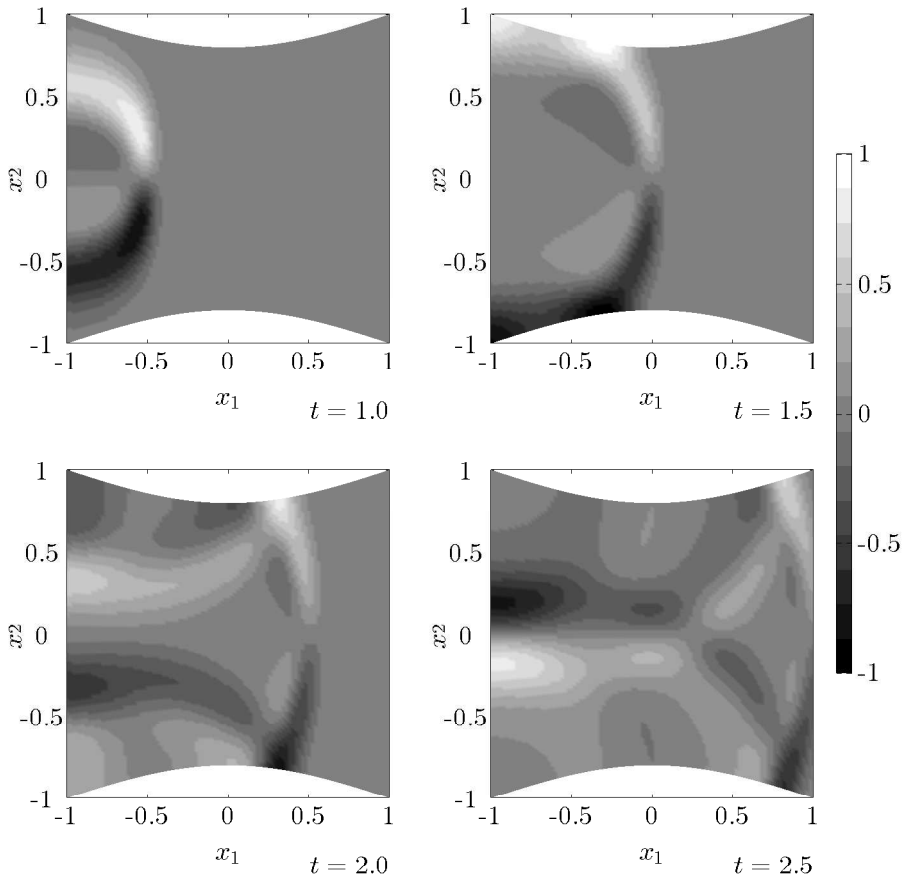


Figure 11.1: Time evolution of (11.1) for $d = 2$. Figure taken [13] from and converted to black and white.

where $J^{\mathrm{pol}}(\mathbf{X})$ denotes again formal insertion of the coordinate matrices into a polynomial approximation of $J(x)$. $(J^{\mathrm{pol}}(\mathbf{X}))^{-1}$ is computed as $(J^{-1})^{\mathrm{pol}}(\mathbf{X})$, rather than as the inverse of $J^{\mathrm{pol}}(\mathbf{X})$. We let $\mathbf{d} = \dot{\mathbf{c}}$ and rewrite (11.2) as the first order system

$$\begin{pmatrix} \dot{\mathbf{c}} \\ \dot{\mathbf{d}} \end{pmatrix} = \begin{pmatrix} 0 & I \\ -(J^{-1})^{\mathrm{pol}}(\mathbf{X})\mathbf{S} & -(J^{-1})^{\mathrm{pol}}(\mathbf{X})\mathbf{B} \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ (J^{-1})^{\mathrm{pol}}(\mathbf{X})\mathbf{g} \end{pmatrix}. \tag{11.3}$$

A first order formulation facilitates explicit time-stepping using standard methods, e.g., the 4th-order Runge–Kutta method. All constituent parts of the right-hand side can be evaluated in essentially linear time using the procedures devised in Chapter 9.

In Figure 11.1, we show what the solution of the two-dimensional problem looks like at four different points in time. Note how the solution is reflected by, as well as glancing along, the curved boundaries. The computation was done with order $K = 48$ Legendre polynomials in each direction, and the coefficients a, b and $(J^{-1})^{\mathrm{pol}}$ were represented by their order $R = 6$ Chebyshev interpolants. Time-stepping was done with the 4th-order Runge–Kutta method with the time step $\Delta t = 2\text{e-}03$.

To test the accuracy of our method, we propagate (11.3) using the 4th-order Runge–Kutta method with the time step $\Delta t = 1\text{e-}03$. We measure the L^2-error of the obtained approximations for various choices of K when compared to a reference solution after transformation of the propagated coefficients to function values on a spatial grid with 100 grid

K	$d=2$		$d=3$	
	error	time (s)	error	time (h)
22	3.529e-02	37.2	3.767e-01	1.93
26	2.103e-02	51.5	2.353e-01	3.15
30	1.431e-02	68.5	1.531e-01	4.80
34	7.159e-03	87.7	7.719e-02	6.94
38	3.808e-03	109.5	4.396e-02	9.90
42	2.238e-03	133.8	2.433e-02	14.18
46	9.802e-04	160.6	1.065e-02	17.60
50	4.888e-04	188.3	5.629e-03	22.72

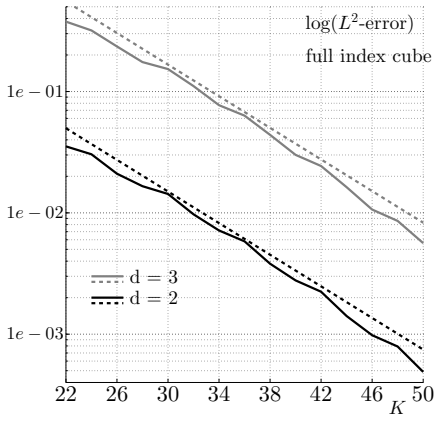


Figure 11.2: [13] L^2-errors and computation times for both $d=2$ (black lines) and $d=3$ (gray lines) for various choices of K. The solid lines represent the L^2-errors, while the dashed lines stand for curves that are proportional to $\exp(-0.15K)$. Semi-logarithmic plot.

points per dimension. The reference solution was computed with $K=64$ and $\Delta t=$ 5e-04. Figure 11.2 shows the observed errors and propagation times in two and three dimensions. As is readily seen, the observed errors decay exponentially with respect to K.

11.3 Assembling the stiffness matrix

To illustrate how much a gain in computation time using the above matrix-free procedures actually is, we briefly compare the observed propagation times to an explicit assembly of the stiffness matrix $\mathbf{S}^{\text{quad}}$ using $(K+1)$-nodes full-product Gauß–Legendre quadrature.

The derivatives of the Legendre basis functions can be computed using the recurrence relations (8.12) and (8.13). To compute $\varphi_k'(\xi_j)$ at the Gaussian quadrature nodes, for all $0 \le j,k \le K$, it takes $\mathcal{O}(K^2)$ operations. Given these values and interpolation coefficients $\hat{a}_{\mathbf{r}}^{(\alpha,\beta)}$, $\mathbf{r} \in \mathcal{R}(d,R)$, for $a_{\text{pol}}^{(\alpha,\beta)}$, $\alpha,\beta = 1,\dots,d$, the entries of $\mathbf{S}^{\text{quad}}$ are computed by the quadrature rule

$$\mathbf{S}_{\mathbf{jk}}^{\text{quad}} = \sum_{\alpha,\beta=1}^{d} \sum_{\mathbf{r}\in\mathcal{R}} \hat{a}_{\mathbf{r}}^{(\alpha,\beta)} \prod_{\gamma=1}^{d} \sum_{m_\gamma=0}^{K} \omega_{m_\gamma} \varphi_{j_\gamma}^{(\alpha,\gamma)}(\xi_{m_\gamma}) T_{\mathbf{r}_\gamma}(\xi_{m_\gamma}) \varphi_{k_\gamma}^{(\beta,\gamma)}(\xi_{m_\gamma}),$$

where $\varphi_j^{(\alpha,\gamma)} = \frac{\partial}{\partial x_\alpha}\varphi_j$ in case $\alpha=\gamma$, and $\varphi_j^{(\alpha,\gamma)} = \varphi_j$ otherwise. Thus, the overall assembly costs for the full stiffness matrix amount to

$$\sim d^3|\mathcal{R}|(K+1)|\mathcal{K}|^2 \sim d^3 R^d K^{2d+1}$$

operations.

In Table 11.1, we show assembly times for $\mathbf{S}^{\text{quad}}$ corresponding to the above example. The number of non-vanishing coefficients $\hat{a}_{\mathbf{r}}^{(\alpha,\beta)}$ is only 18 and 22 for $d=2$ and $d=3$, respectively, which has been taken into account. As is seen, for $d=2$, it does not take an excessive K for the assembly of the stiffness matrix to require a similar amount of time as the full time propagation with our matrix-free method. For $d=3$, assembling the matrix is

prohibitively expensive: For $K = 50$, there are 132651 basis functions. Computing a single row of the corresponding stiffness matrix takes about 11.6 s. By extrapolation, the overall assembly time would amount to 427.2 h or 17.8 days—memory requirements left aside.

K	$d = 2$		$d = 3$	
	time (s)	ratio	time	ratio
30	10.0	0.15	41.7 h	8.69
50	104.5	0.55	≈ 17.8 days	18.80

Table 11.1: [13] Computation times for the assembly of $\mathbf{S}^{\mathrm{quad}}$ and time ratios when compared to the overall propagation times using our fast algorithm as given in Figure 11.2.

11.4 Comment on reduced index sets

Throughout Part II of this thesis, we have not yet taken into account the possibility as well as the effects of reducing the underlying index set $\mathcal{K}(d, K)$. As in the case of the fast algorithm for the linear Schrödinger equation, the procedures devised in Chapter 9 are applicable with any index set that is closed under componentwise decrements (cf. (1.15)) provided that the regularity relations

$$|\mathcal{K}(d, K)| \gg |\mathcal{R}(d, R)|, \qquad K \gg R$$

hold true, where $\mathcal{R}(d, R)$ stands for any index that is used for the polynomial approximation of an arbitrary coefficient function $a^{(\alpha,\beta)}$, b.

Algorithmically, reducing the index set yields some changes to the above procedures. We recall the definition (2.11) of $v_{\mathcal{K}}^{\max}(\mathbf{k})$, which is the maximal integer such that, when inserted into $\mathcal{K}(d-1, K)$ at an arbitrary position, the resulting index is in $\mathcal{K}(d, K)$. We consider an additive reduction $\mathcal{K}(d, K) = \mathcal{K}_{\mathrm{add}}(d, K)$; see (1.21). It holds

$$v_{\mathcal{K}_{\mathrm{add}}}^{\max}(\mathbf{k}) = K - \sum_{\alpha=1}^{d-1} k_\alpha \qquad \forall\, \mathbf{k} \in \mathcal{K}_{\mathrm{add}}(d-1, K).$$

The algorithmic changes can then be stated as follows: First, upper bounds as occurring in a number of loops need to be adjusted. In Algorithms 12 and 13, the upper bounds for j_α in Lines **5** have to be changed from K to $v_{\mathcal{K}_{\mathrm{add}}}^{\max}(\mathbf{j}^{(\neg\alpha)})$. In Algorithm 12, this also applies to Lines **3** and **4**. Using the notation of Chapter 9, if the index set $\mathcal{R}$ for the polynomial interpolation of q as defined in (9.10) is also reduced, the upper bound R for r in the recursive fast algorithm for the action of the coefficients has to be replaced by $v_{\mathcal{R}}^{\max}(\mathbf{r}^{(\neg\alpha)})$; see the end of Section 9.2. Next, the second term in the coordinate matrix application (9.9) vanishes if $\mathbf{j} + \mathbf{e}_\alpha$ happens to fall outside $\mathcal{K}_{\mathrm{add}}$.

Again, we opt for using index manuals in order to compute missing linear addresses of manipulated indices; see the discussion in Section 2.5.

For any reduced index set, the identities given in Section 9.1, where the exactness of Gaussian quadrature has been used, do no longer hold. Using again the notation of Chapter 9, this yields additional errors due to index set reduction

$$\mathbf{Q}^{\mathrm{quad}} \approx q(\mathbf{X}), \qquad \mathbf{Q}_{\mathrm{quad}}^{(\alpha,\beta)} \approx \big(\mathbf{D}^{(\alpha)}\big)^T \mathbf{Q}^{\mathrm{quad}} \mathbf{D}^{(\beta)} \approx \big(\mathbf{D}^{(\alpha)}\big)^T q(\mathbf{X}) \mathbf{D}^{(\beta)}. \tag{11.4}$$

It is unclear how the right-hand side error in (11.4) can be dealt with using the binary tree technique developed in Sections 4.4–4.7. The procedures have proven to work, though, experimentally.

K	$d=2$		$d=3$	
	error	time (s)	error	time (h)
30	3.158e-02	44.0	1.407e+00	1.14
34	1.896e-02	56.4	1.078e+00	1.72
38	1.197e-02	69.5	8.704e-01	2.30
42	7.248e-03	85.0	6.867e-01	3.08
46	4.304e-03	100.3	5.264e-01	4.02
50	2.832e-03	118.2	3.577e-01	5.19

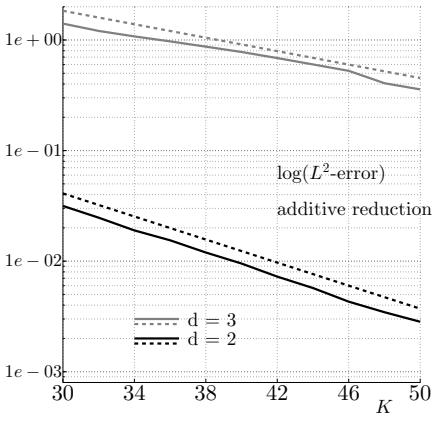


Figure 11.3: L^2-errors and computation times as in Figure 11.2, but with an additively reduced index set. The dashed lines stand for curves that are proportional to $\exp(-0.12K)$ and $\exp(-0.07K)$ in case $d=2$ and $d=3$, respectively. Semi-logarithmic plot.

In Figure 11.3, we show exactly the same experiment as in Figure 11.2 of Section 11.2, but with an additively reduced index set in place of a full index cube. As it is seen, the method still decays exponentially, but the decay becomes slower with increasing d.

However, when applying our method with a reduced index set, it is flawed by the approximations (11.4). In Figure 11.4, we plot computation times vs. errors both in 2D and 3D, both for the full and the reduced index set. As can be seen, there is almost no gain in efficiency in the 2D case, while the time-error ratio actually gets worse in 3D. This is due to the errors (11.4) not behaving sufficiently well. We have done further tests with a hyperbolic reduction, where the time-error behavior happens to be even more unfavorable.

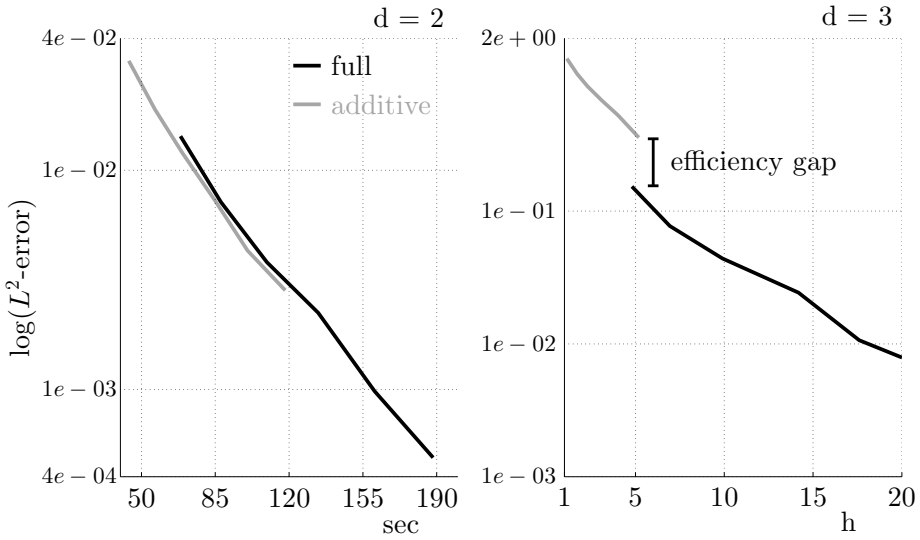


Figure 11.4: Time vs. error for the full index set (black) and the additive reduction (gray). The left picture shows the case $d=2$; the right picture shows $d=3$. The data is taken from Figures 11.2 and 11.3.

Afterword

"All that indeed your poem lacked
Was logic, modesty, and tact"
– Alec Derwent Hope, *His Coy Mistress to Mr. Marvell*

In Australian poet A.D. Hope's answer to Marvell (published in 1978 with a respectable delay of about 325 years), the resolute mistress turns down the courtship outright and picks Marvell's art of persuasion to pieces; see [51]. So the poor fool's life has been spent in vain with all his efforts being ridiculed. We shall therefore say that our own endeavor has turned out to be less of a failure than Mr. Marvell's: One might still spent a lifetime trying to win a reluctant heart, but with the help of the fast algorithm, solving high-dimensional PDEs is no longer an eternal task.

We started from a Galerkin discretization in space of the time-dependent Schrödinger equation, using tensor products of Hermite functions as the underlying basis. Without reducing the number of basis functions, the number of equations in the resulting system of ODEs depends exponentially on the dimension of the problem. Since time-stepping typically requires matrix-vector products with the corresponding matrix representation of the Hamiltonian operator, this is a hopeless situation in dimensions larger than three. Based on the orthogonality and three-term recurrence for the Hermite functions, we have devised and analyzed a fast algorithm that nevertheless allows to compute such matrix-vector products without assembling the matrix itself in essentially linear time. The approach is compatible with an arbitrary reduction of the basis set. It is serviceable also in cases where the chosen Galerkin basis does not allow for a fast transform.

Applying the fast algorithm is restricted to situations where the potential is significantly more regular than the wave function, both due to complexity reasons and to ensure convergence. For an unreduced basis, the approach turns out to be equivalent to a suitable choice of Gaussian quadrature. We have analyzed the local errors due to quadrature and due to index set reduction, and we have presented a global error analysis.

The fast algorithm is not narrowly taylored for the Schrödinger equation only, but rather constitutes a general tool to deal with the intractabilities of spectral approximations to high-dimensional PDEs. In particular, we have shown further applications of the strategy to nonlinearities and, most notably, to initial-boundary value problems with non-constant coefficients and non-trivial boundary conditions.

The efficiency of the fast algorithm becomes clear when comparing it to an existing approach from the chemical literature. We hope that people might find it beneficial for their purposes to use the new methodology.

Notations

We give a brief overview over the notational conventions used in this thesis. The first column of the following table contains the symbols; the second column gives a concise explanation in prose; in the last column, if applicable, we refer to the equation number or section number where the symbol is defined. Some minor notational conventions that have just been used locally, e.g., within a proof, have not been added to the list.
Explanations of symbols that appear in Part II only are written in italics.

symbol	**explanation**	**no.**
	integer and real parameters	
$d \in \mathbb{N}$	dimension of the underlying problem	(1.1)
$\alpha, \beta, \gamma \in \{1, \ldots, d\}$	variables for dimension	–
$K, L, M, R \in \mathbb{N}$	threshold for index sets	1.3
$r, s \in \mathbb{N}$	regularity indices	–
$S \in \mathbb{R}$	width of support of exact solution and polynomial approximation to potential	1.4.1
$\tilde{S} \in \mathbb{R}$	stretching parameter for Galerkin basis	1.4.1
$h \in \mathbb{R}$ $n, N \in \mathbb{N}$	time step size counter and total number of time steps	1.5.1
$m \in \mathbb{N}$	number of Lanczos iterations	1.5.2
	spatial domain	
Ω	$= \begin{cases} [-S,S]^d, & \text{Part I} \\ [-1,1]^d, & \text{Part II} \end{cases}$	1.4.1 8.1
$\Omega^{(\pm\alpha)}$	*boundary face*	9.4
$\Omega^{(\neg\alpha)}$	*dimensional reduction of* Ω	9.4
$x = (x_1, \ldots, x_d)$	spatial variable	(1.1)
	spaces and norms	
$(\cdot,\cdot)$, $\|\cdot\|$	$\begin{cases} L^2(\mathbb{R}^d)\text{-scalar prod. \& norm,} & \text{Part I} \\ L^2([-1,1]^d)\textit{-scalar prod. \& norm,} & \text{Part II} \end{cases}$	(1.4)
$\|\cdot\|_{L^2_\omega(\Omega)}$	weighted L^2-norm	1.4.2
$(\cdot,\cdot)_{\partial\Omega}$, $\|\cdot\|_{\partial\Omega}$	*restriction to boundary*	8.1

$(\cdot,\cdot)_{\Omega(\pm\alpha)}$, $\|\cdot\|_{\Omega(\pm\alpha)}$	*restriction to boundary face*	9.4
$(\cdot,\cdot)_{\Omega(\neg\alpha)}$	*dimensional reduction*	9.4
$\|\mathbf{v}\|$	for vector arguments: Euclidean norm	–
$\|\cdot\|_{H^s(\Omega)}$, $\|\cdot\|_{H^s(\Omega)}$	standard Sobolev (semi)norm	8.3
$\|\cdot\|_{H^{s;R}_\omega(\Omega)}$	weighted Sobolev seminorm	1.4.2
$\|\cdot\|_s$	Korobov-type (semi)norms	(1.19)
$\|\cdot\|_{s;\infty}$		(1.20)
$\|\cdot\|_E$	*continuous energy norm*	(8.7)
$\|\cdot\|_{\mathbf{E}}$	*discrete energy norm*	(10.7)
$\mathcal{S}(\mathbb{R})$, $\mathcal{S}(\mathbb{R}^d)$	Schwartz space	1.2
multi-indices		
$\mathbf{j},\mathbf{k},\mathbf{l},\mathbf{m} \in \mathbb{N}^d$	d-dimensional multi-indices, $\mathbb{N} = \{0,1,2,\dots\}$	–
$\mathbf{k}^{(\neg\alpha)} \in \mathbb{N}^{d-1}$	delete αth component from $\mathbf{k} \in \mathbb{N}^d$	(2.6)
$\mathbf{k} \overset{\alpha}{\leftarrow} k \in \mathbb{N}^d$	replace k_α with k in $\mathbf{k} \in \mathbb{N}^d$	(2.7)
$\mathbf{e}_\alpha \in \mathbb{N}^d$	αth unit tuple	(1.15)
$\|\mathbf{k}\|$	1-norm	(1.22)
$\|\mathbf{k}\|_\infty$	maximum norm	(1.17)
multi-index sets		
$\mathcal{K} = \mathcal{K}(d,K)$ $\subseteq \mathcal{K}_{\text{full}}(d,K)$,	set of d-dim. multi-indices with threshold K, subset of the corresponding full index cube, analogously with $\mathcal{L}(d,L)$, $\mathcal{M}(d,M)$, $\mathcal{R}(d,R)$	1.3
$\mathcal{K}_{\text{hyp}}(d,K)$	hyperbolically reduced set	(1.16)
$\mathcal{K}_{\text{add}}(d,K)$	additively reduced set	(1.21)
$\|\mathcal{K}\|$	size of index set = number of indices	–
$\text{lin}(\mathbf{k})$	linear address of $\mathbf{k} \in \mathcal{K}$ w.r.t. linear order of $\mathcal{K}$	2.5.1
$v^{\max}_{\mathcal{K}}(\mathbf{k})$	maximal insertable value without leaving $\mathcal{K}$	(2.11)
operators		
H	Hamiltonian operator	(1.1)
D	differential operator in SOP form, in particular: harmonic oscillator	(6.2) (1.7)
$T = \sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}} T_{\mathbf{r}} =$ $\sum_{\mathbf{r}\in\mathcal{R}} \hat{t}_{\mathbf{r}} T^{(1)}_{r_1} \dots T^{(d)}_{r_d}$	any operator in SOP form; only in 3.2	3.2
x, x_α	1D and position operator w.r.t. αth coordinate	(1.14)
A, A_α	ladder operators (w.r.t. αth coordinate),	(1.12)
$A^-, A^\dagger, A^-_\alpha, A^\dagger_\alpha$	$A^{\mathbf{s}} = A_1^{s_1} \dots A_d^{s_d}$, and analogously for A^-, $A^\dagger$	(4.22)

σ	cutting operator	(4.17)
σ_+	blow-up operators	(4.18)
functions and their polynomial approximants		
η, ϕ, χ, ψ	L^2-functions over $\mathbb{R}^d$ or $[-1,1]^d$	–
q	arbitrary d-variate polynomial ...	2.1
$q^{(\pm\alpha)} = q\|_{\Omega^{(\pm\alpha)}}$	... and their restrictions to boundary face	(9.14)
V, W	multiplicative, non-polynomial potentials ...	1.1
W^{pol}	... and their polynomial approximants	(1.24)
a, $a^{(\alpha,\beta)}$, b, $b^{(\pm\alpha)}$	*non-polynomial coefficient functions ...*	8.1
a^{pol}, $a^{(\alpha,\beta)}_{\mathrm{pol}}$, b^{pol}, $b^{(\pm\alpha)}_{\mathrm{pol}}$	*... and their polynomial approximants*	9.5
$\hat{q}_{\mathbf{r}}$, $\hat{w}_{\mathbf{r}}$, $\hat{a}^{(\alpha,\beta)}_{\mathbf{r}}$	expansion coefficients of q, W^{pol}, $a^{(\alpha,\beta)}_{\mathrm{pol}}$	–
f, g	*right-hand side of wave equation*	8.1
$\mathbf{n}$	*outer normal vector*	
Galerkin basis, orthogonal polynomials		
$\varphi_k(x)$, $0 \le k \le K$	Hermite function, $x \in \mathbb{R}$, Part I *L^2-norm. Legendre pol.s*, $x \in [-1,1]$, Part II	1.2.1 8.3
$\varphi_{\mathbf{k}}(x)$, $\mathbf{k} \in \mathcal{K}$	$= \prod_{\alpha=1}^d \varphi_{k_\alpha}(x_\alpha)$, tensor product basis	1.2.2 8.2
$\mathcal{P}_{\mathcal{K}}, \mathcal{P}^{\perp}_{\mathcal{K}}$	L^2-orth. projection onto $\mathrm{span}\{\varphi_{\mathbf{k}} \,\|\, \mathbf{k} \in \mathcal{K}\}$	(1.5)
$H_k, H_{\mathbf{k}}$	(tensor products of) Hermite polynomials	1.2
P_k	*1D boundary-norm. Legendre polynomials*	8.3
$T_r, T_{\mathbf{r}}$	(tensor products of) Chebyshev polynomials	(1.24)
ω	1D weight function for class. orth. polynomials	–
quadrature		
$(w_m, \xi_m)_{m=0}^M$	1D Gaussian quadrature: ξ_m are zeros of φ_{M+1}	2.6 9.1
$\omega_m = \exp(\xi_m^2) w_m$	Gauß–Hermite weights times exponentials	(2.14)
$\xi_{\mathbf{m}} = (\xi_{m_\alpha})_{\alpha=1}^d$, $w_{\mathbf{m}}$, $\omega_{\mathbf{m}}$, $\mathbf{m} \in \mathcal{M}_{\mathrm{full}}$	full-product Gaussian quadrature	(2.14) (9.1)
exact solutions and their coefficients		
ψ, $u_{\mathbf{k}}$	Part I: original weak formulation	(4.1)
ψ^{pol}, $u^{\mathrm{pol}}_{\mathbf{k}}$	Part I: $W \approx W^{\mathrm{pol}}$	(4.2)
$\psi^{\mathrm{quad}}_{\mathcal{K}}$, $c^{\mathrm{quad}}_{\mathbf{k}}$	Part I: Galerkin + quadrature	(4.4)
$\psi^{\mathrm{fast}}_{\mathcal{K}}$, $c^{\mathrm{fast}}_{\mathbf{k}}$	Part I: Galerin + fast alg.	(4.7)
ψ, $u_{\mathbf{k}}$	*Part II: original weak formulation*	(10.1)

ψ^{pol}, $u_{\mathbf{k}}^{\mathrm{pol}}$	*Part II: approximation of coefficients*	(10.2)
$\psi_{\mathcal{K}}$, $c_{\mathbf{k}}$	*Part II: Galerkin + fast alg.*	(10.3)
vectors		
$\mathbf{c} = (c_{\mathbf{k}})_{\mathbf{k}\in\mathcal{K}}$, $\mathbf{u}, \mathbf{v}, \mathbf{w}$	real or complex vectors of size $\lvert\mathcal{K}\rvert$	–
$\mathbf{v}^{(\alpha)}$	block vectors for sequential summations	3.2.1
$\mathbf{e}^{\mathrm{quad}}(\mathbf{v}, t)$	error due to quadrature	(4.19)
$\mathbf{e}^{\mathrm{red}}(\mathbf{v}, t)$	error due to index set reduction	(4.20)
$(d_{\mathbf{l}})_{\mathbf{l}\in\mathcal{L}}$	expansion coefficients of nonlinearity	6.4.3
$\mathbf{f}$, $\mathbf{g}$	*internal and boundary vectors*	(10.5)
$\mathbf{f}_{\mathcal{K}}$, $\mathbf{g}_{\mathcal{K}}$	*internal and boundary defects*	(10.9)
$\mathbf{v}^{(\neg\alpha,\pm)}$	*dimensional reduction of* $\mathbf{v}$	(9.16)
matrices		
$\mathbf{H}_{\mathbf{jk}} = (\varphi_{\mathbf{j}}, H\varphi_{\mathbf{k}})$,	$(\lvert\mathcal{K}\rvert \times \lvert\mathcal{K}\rvert)$-basis representation of H	(1.6)
$\mathbf{D}$	basis representation of D, in particular: diagonal eigenvalue matrix	– (1.8)
$\mathbf{Q}$	basis representations of q	(2.1)
$\mathbf{Q}^{(\pm\alpha)}$	… *of* $q^{(\pm\alpha)}$	(9.15)
$\mathbf{W}$	… of W	(1.9)
$\mathbf{W}^{\mathrm{pol}}$	… of W^{pol}	(1.29)
$\mathbf{T}$, $\mathbf{T}^{\mathbf{r}}$	… of T, $T_{\mathbf{r}}$; only in 3.2, where $(T_r^{(\alpha)})_{jk} = (\varphi_j, T_r^{(\alpha)}\varphi_k)$	3.2
$\mathbf{S}$, $\mathbf{S}^{(\alpha,\beta)}$	*stiffness matrix*	(8.11)
$\mathbf{B}$, $\mathbf{B}^{(\pm\alpha)}$	*boundary term matrix*	(8.11)
$\mathbf{Q}^{(\alpha,\beta)}$	*basis representation of q with derivatives*	(9.2)
$\mathbf{Q}^{\mathrm{quad}}$	quadrature approximation to $\mathbf{Q}$	(2.15)
$\mathbf{Q}_{\mathrm{quad}}^{(\pm\alpha)}$	… *to* $\mathbf{Q}^{(\pm\alpha)}$	(9.17)
$\mathbf{W}^{\mathrm{quad}}$	… to $\mathbf{W}^{\mathrm{pol}}$	(3.1)
$\mathbf{S}^{\mathrm{quad}}$, $\mathbf{S}_{\mathrm{quad}}^{(\alpha,\beta)}$	… *to* $\mathbf{S}$, $\mathbf{S}^{(\alpha,\beta)}$	(9.8)
$\mathbf{B}^{\mathrm{quad}}$, $\mathbf{B}_{\mathrm{quad}}^{(\pm\alpha)}$	… *to* $\mathbf{B}$, $\mathbf{B}^{(\pm\alpha)}$	(9.18)
$\mathbf{Q}_{\mathrm{quad}}^{(\alpha,\beta)}$	… *to* $\mathbf{Q}^{(\alpha,\beta)}$	(9.6)
X	1D coordinate matrix	(2.3)
$\mathbf{X}^{(\alpha)}$	coordinate matrix w.r.t αth coordinate	(2.4)
$q(\mathbf{X})$, $W^{\mathrm{pol}}(\mathbf{X})$, $a^{(\alpha,\beta)}(\mathbf{X})$, $b^{(\pm\alpha)}(\mathbf{X})$	formal insertion of $\mathbf{X}^{(\alpha)}$ in place of x_α	2.3.1

$\mathbf{U}$ $\mathbf{U}^{(\alpha)}$	orthogonal diagonalization matrices	(2.16) (9.4)
D $\mathbf{D}^{(\alpha)}$	*1D differentiation matrix* *differentiation matrix w.r.t. αth coordinate*	(9.11) (9.5)
$\mathbf{P}$ $\mathbf{P}^{(\mathbf{l})}$, $\mathbf{l} \in \mathcal{L}$	basis representation of nonlinearity triple-product matrix	(6.11) (6.13)
$\mathbf{\Omega}^{(n)}$	Magnus integrator matrix	1.5.1
$\mathbf{V}_m$, $\mathbf{V}_m^{(n)}$ $\mathbf{T}_m$, $\mathbf{T}_m^{(n)}$	$(\lvert\mathcal{K}\rvert \times m)$-Lanczos basis matrices $(m \times m)$-Lanczos coefficient matrices	1.5.2

Algorithms, figures, and tables

The following table gives an overview over all algorithms, figures, and tables contained in this thesis. The columns give the number, a concise explanation, and the page number. Algorithms etc. taken from Part II are indicated by italics.

Lemmas and theorems

The following table gives an overview over all lemmas and theorems. The columns give the number, a concise explanation, and the page number. Again, lemmas and theorems given in Part II are indicated by italics.

#	**explanation**	**page**
	Lemmas	
1	orthogonality of diagonalization matrix **U**	43
2	equivalence lemma: insertion of coordinate matrices = quadrature	44
3	interpolation error (Schrödinger equation)	68
4	local error due to quadrature (full index cube)	70
5	cutting and blowing up	73
6	decay of Hermite expansion coefficients	75
7	perturbation of Lanczos	94
8	*well-posedness of continuous problem*	129
9	*quadrature and derivatives*	135
10	*interpolation error (wave equation)*	146
11	*strong stability of spatial discretization (wave equation)*	147
	Theorems	
1	global error of spatial discretization, full index cube	66
2	global error of spatial discretization, hyperbolic reduction	66
3	(repetition of Theorem 1)	69
4	global error of spatial discretization, hyperbolic reduction; local error in potential not taken into account	72
5	local error due to quadrature, hyperbolic reduction	76
6	local error due to index set reduction, hyperbolic reduction	82
7	*overall error (wave equation)*	145
8	*global error of spatial discretization (wave equation)*	149

Bibliography

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions*. Dover Publications, New York, NY, 10th printing, 1972. Available at `http://people.math.sfu.ca/~cbm/aands/`.

[2] G. Ávila and T. Carrington. Nonproduct quadrature grids for solving the vibrational Schrödinger equation. *J. Chem. Phys.*, 131:174103, 2009. `doi:10.1063/1.3246593`.

[3] G. Ávila and T. Carrington. Using nonproduct quadrature grids to solve the vibrational Schrödinger equation in 12D. *J. Chem. Phys.*, 134:054126, 2011. `doi:10.1063/1.3549817`.

[4] G. Ávila and T. Carrington. Using a pruned basis, a non-product quadrature grid, and the exact Watson normal-coordinate kinetic energy operator to solve the vibrational Schrödinger equation for C_2H_4. *J. Chem. Phys.*, 135:064101, 2011. `doi:10.1063/1.3617249`.

[5] G. Ávila and T. Carrington. Solving the vibrational Schrödinger equation using bases pruned to include strongly coupled functions and compatible quadratures. *J. Chem. Phys.*, 137:174108, 2012. `doi:10.1063/1.4764099`.

[6] G. Ávila and T. Carrington. Solving the Schrödinger equation using Smolyak interpolants. *J. Chem. Phys.*, 139:134114, 2013. `doi:10.1063/1.4821348`.

[7] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, NJ, 1961.

[8] C. Bernardi and Y. Maday. Spectral Methods. In: P. Ciarlet and J. Lions (editors), *Handbook of Numerical Analysis*, Elsevier, Amsterdam, 1997, 209–486. `doi:10.1016/S1570-8659(97)80003-8`.

[9] S. Blanes, F. Casas, J. Oteo, and J. Ros. The Magnus expansion and some of its applications. *Phys. Rep.*, 470:151–238, 2009. `doi:10.1016/j.physrep.2008.11.001`.

[10] J. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover, Mineola, NY, 2nd edition, 2001.

[11] M. Bramley and T. Carrington. A general discrete variable method to calculate vibrational energy levels of three- and four-atom molecules. *J. Chem. Phys.*, 99:8519, 1993. `doi:10.1063/1.465576`.

[12] B. Brumm. A Fast Matrix-free Algorithm for Spectral Approximations to the Schrödinger Equation. *SIAM J. Sci. Comput.*, 37:A2003–A2025, 2015. doi:10.1137/140981022.

[13] B. Brumm and E. Kieri. A matrix-free Legendre spectral method for initial-boundary value problems. *Preprint*, 2015. Available at https://na.uni-tuebingen.de/~brumm/.

[14] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numer.*, 13:147–269, 2004. doi:10.1017/S0962492904000182.

[15] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang. *Spectral Methods: Fundamentals in Single Domains*. Springer, Berlin, 2006. doi:10.1007/978-3-540-30726-6.

[16] Y. Cao, Y. Jiang, and Y. Xu. Orthogonal polynomial expansions on sparse grids. *J. Complexity*, 30:683–715, 2014. doi:10.1016/j.jco.2014.04.001.

[17] M. Carpenter and D. Gottlieb. Spectral methods on arbitrary grids. *J. Comput. Phys.*, 129:74–86, 1996. doi:10.1006/jcph.1996.0234.

[18] M. Carpenter, D. Gottlieb, and S. Abarbanel. Time-Stable Boundary Conditions for Finite-Difference Schemes Solving Hyperbolic Systems: Methodology and Application to High-Order Compact Schemes. *J. Comput. Phys.*, 111:220–236, 1994. doi:10.1006/jcph.1994.1057.

[19] C. Corey, J. Tromp, and D. Lemoine. Fast Pseudospectral Algorithm in Curvilinear Coordinates. In: C. Cerjan (editor), *Numerical Grid Methods and their Application to Schrödinger's Equation*, NATO ASI Series. Series C: Mathematical and Physical Sciences 412. Kluwer Academic, Boston, MA, 1993, 1–23. doi:10.1007/978-94-015-8240-7_1.

[20] E. Davidson. Monster Matrices: Their Eigenvalues and Eigenvectors. *Comput. Phys.*, 7:519–522, 1993. doi:10.1063/1.4823212.

[21] D. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*. Springer, Berlin, 2012. doi:10.1007/978-3-642-22980-0.

[22] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical simulation of waves. *Math. Comp.*, 31:629–651, 1977. doi:10.1090/S0025-5718-1977-0436612-4.

[23] E. Faou, V. Gradinaru, and Ch. Lubich. Computing Semiclassical Quantum Dynamics with Hagedorn Wavepackets. *SIAM J. Sci. Comput.*, 31:3027–3041, 2009. doi:10.1137/080729724.

[24] E. Feldheim. Quelques Nouvelles Relations Pour les Polynômes D'Hermite. *J. London Math. Soc.*, 13:22–29, 1938. doi:10.1112/jlms/s1-13.1.22.

[25] K.-A. Feng, C.-H. Teng, and M.-H. Chen. A Pseudospectral Penalty Scheme for 2D Isotropic Elastic Wave Computations. *J. Sci. Comput.*, 33:313–348, 2007. doi:10.1007/s10915-007-9154-8.

[26] B. Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, Cambridge, 1996. doi:10.1017/CBO9780511626357.

[27] R. Friesner, R. Wyatt, C. Hempel, and B. Criner. A generalized version of the recursive residue generation method for vector computers. *J. Comput. Phys.*, 64:220–229, 1986. doi:10.1016/0021-9991(86)90026-4.

[28] D. Funaro. Domain decomposition methods for pseudo spectral approximations. *Numer. Math.*, 52:329–344, 1987. doi:10.1007/BF01398883.

[29] D. Funaro and D. Gottlieb. A new method of imposing boundary conditions in pseudospectral approximations of hyperbolic equations. *Math. Comp.*, 51:599–613, 1988. doi:10.2307/2008765.

[30] L. Gauckler. Convergence of a split-step Hermite method for the Gross–Pitaevskii equation. *IMA J. Numer. Anal.*, 31:396–415, 2011. doi:10.1093/imanum/drp041.

[31] W. Gautschi. *Orthogonal Polynomials: Computation and Approximation*. Oxford University Press, New York, NY, 2004.

[32] W. Gautschi. *Numerical Analysis: An Introduction*. Birkhäuser, Boston, MA, 2nd edition, 2012. doi:10.1007/978-0-8176-8259-0.

[33] T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numer. Algor.*, 18:209–232, 1998. doi:10.1023/A:1019129717644.

[34] V. Gradinaru. Fourier transform on sparse grids: Code design and the time dependent Schrödinger equation. *Computing*, 80:1–22, 2007. doi:10.1007/s00607-007-0225-3.

[35] V. Gradinaru. Strang Splitting for the Time-Dependent Schrödinger Equation on Sparse Grids. *SIAM J. Numer. Anal.*, 46:103–123, 2008. doi:10.1137/050629823.

[36] V. Gradinaru and G. Hagedorn. Convergence of a semiclassical wavepacket based time-splitting for the Schrödinger equation. *Numerische Mathematik*, 126:53–73, 2014. doi:10.1007/s00211-013-0560-6.

[37] M. Griebel and J. Hamaekers. Fast Discrete Fourier Transform on Generalized Sparse Grids. In: J. Garcke and D. Pflüger (editors), *Sparse Grids and Applications – Munich 2012*, Lecture Notes in Computational Science and Engineering 97. Springer, Berlin, 2014, 75–107. doi:10.1007/978-3-319-04537-5_4.

[38] B.-Y. Guo. *Spectral Methods and Their Applications*. World Scientific Publishing Co. Inc., River Edge, NJ, 1998. doi:10.1142/9789812816641_fmatter.

[39] S. Gustafson and I. Sigal. *Mathematical Concepts of Quantum Mechanics*. Springer, Berlin, 2003. doi:10.1007/978-3-642-21866-8.

[40] B. Gustafsson, H.-O. Kreiss, and J. Oliger. *Time Dependent Problems and Difference Methods*. Wiley, Hoboken, NJ, 2nd edition, 2013. doi:10.1002/9781118548448.

[41] G. Hagedorn. Semiclassical quantum mechanics. III: The large order asymptotics and more general states. *Ann. Phys*, 135:58–70, 1981. doi:10.1016/0003-4916(81)90143-3.

[42] G. Hagedorn. Semiclassical quantum mechanics, IV: large order asymptotics and more general states in more than one dimension. *Ann. Inst. Henri Poincaré Sect. A*, 42:363–374, 1985. Available at http://www.numdam.org/item?id=AIHPA_1985__42_4_363_0.

[43] G. Hagedorn. Raising and Lowering Operators for Semiclassical Wave Packets. *Ann. Phys.*, 269:77–104, 1998. doi:10.1006/aphy.1998.5843.

[44] K. Hallatschek. Fouriertransformationen auf dünnen Gittern mit hierarchischen Basen. *Numer. Math.*, 63:83–97, 1992. doi:10.1007/BF01385849.

[45] J. Hesthaven. Spectral penalty methods. *Appl. Numer. Math.*, 33:23–41, 2000. doi:10.1016/S0168-9274(99)00068-9.

[46] J. Hesthaven and D. Gottlieb. A Stable Penalty Method for the Compressible Navier–Stokes Equations: I. Open Boundary Conditions. *SIAM J. Sci. Comput.*, 17:579–612, 1996. doi:10.1137/S1064827594268488.

[47] J. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge University Press, Cambridge, 2007. doi:10.1017/CBO9780511618352.

[48] J. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*. Springer, Berlin, 2008. doi:10.1007/978-0-387-72067-8.

[49] M. Hochbruck and Ch. Lubich. On Krylov Subspace Approximations to the Matrix Exponential Operator. *SIAM J. Numer. Anal.*, 34:1911–1925, 1997. doi:10.1137/S0036142995280572.

[50] M. Hochbruck and Ch. Lubich. On Magnus Integrators for Time-Dependent Schrödinger equations. *SIAM J. Numer. Anal.*, 41:945–963, 2003. doi:10.1137/S0036142902403875.

[51] A.D. Hope. *A Book of Answers*. Angus & Robertson, Sydney, 1978.

[52] Y. Jiang and Y. Xu. Fast discrete algorithms for sparse Fourier expansions of high dimensional functions. *J. Complexity*, 26:51–81, 2010. doi:doi:10.1016/j.jco.2009.10.001.

[53] Y. Jiang and Y. Xu. Fast computation of the multidimensional discrete Fourier transform and discrete backward Fourier transform on sparse grids. *Math. Comp.*, 83:2347–2384, 2014. doi:10.1090/S0025-5718-2014-02785-3.

[54] L. Kämmerer and S. Kunis. On the stability of the hyperbolic cross discrete Fourier transform. *Numer. Math.*, 117:581–600, 2011. doi:10.1007/s00211-010-0322-7.

[55] T. Kato. *Perturbation Theory for Linear Operators*. Springer, Berlin, 2nd edition, 1980. doi:10.1007/978-3-642-66282-9.

[56] A. Klimke. Efficient Construction of Hierarchical Polynomial Sparse Grid Interpolants using the Fast Discrete Cosine Transform. Technical report, IANS preprint 2006/007, University of Stuttgart, 2006. Available at http://preprints.ians.uni-stuttgart.de.

[57] H.-O. Kreiss and J. Oliger. Stability of the Fourier Method. *SIAM J. Numer. Anal.*, 16:421–433, 1979. doi:10.1137/0716035.

[58] C. Lasser and S. Troppmann. Hagedorn Wavepackets in Time-Frequency and Phase Space. *J. Fourier An. Appl.*, 20:679–714, 2014. doi:10.1007/s00041-014-9330-9.

[59] J. Light and T. Carrington. Discrete Variable Representations and their Utilization. *Adv. Chem. Phys.*, 114:263–310, 2000. doi:10.1002/9780470141731.ch4.

[60] Ch. Lubich. Integrators for quantum dynamics: a numerical analyst's brief review. In: J. Grotendorst, D. Marx, and A. Muramatsu (editors), *Quantum Simulations of Many-Body Systems: From Theory to Algorithms*, NIC Series Vol. 10. John von Neumann Institute for Computing, Jülich, 2002, 459–466. Available at https://juser.fz-juelich.de/record/24560/files/NIC-Band-10.pdf.

[61] Ch. Lubich. *From Quantum to Classical Molecular Dynamics: Reduced Models and Numerical Analysis*. European Math. Soc., Zürich, 2008. doi:10.4171/067.

[62] U. Manthe and H. Köppel. New method for calculating wave packet dynamics: Strongly coupled surfaces and the adiabatic basis. *J. Chem. Phys.*, 93:345, 1990. doi:10.1063/1.459606.

[63] K. Mattsson, F. Ham, and G. Iaccarino. Stable Boundary Treatment for the Wave Equation on Second-Order Form. *J. Sci. Comput.*, 41:366–383, 2009. doi:10.1007/s10915-009-9305-1.

[64] R. McLachlan and G. Quispel. Splitting methods. *Acta Numer.*, 11:341–434, 2002. doi:10.1017/S0962492902000053.

[65] C. Moler and Ch. Van Loan. Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later. *SIAM Rev.*, 45:3–49, 2003. doi:10.1137/S00361445024180.

[66] J. Oden, I. Babuška, and C. Baumann. A Discontinuous *hp* Finite Element Method for Diffusion Problems. *J. Comput. Phys.*, 146:491–519, 1998. doi:10.1006/jcph.1998.6032.

[67] F. Olver, D. Lozier, R. Boisvert, and C. Clark. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010. Online companion *NIST Digital Library of Mathematical Functions* available at http://dlmf.nist.gov/ (release 1.0.10 of 2015-08-07).

[68] S. Olver and A. Townsend. A Fast and Well-Conditioned Spectral Method. *SIAM Rev.*, 55:462–489, 2013. doi:10.1137/120865458.

[69] E. Ortiz. The Tau Method. *SIAM J. Numer. Anal.*, 6:480–492, 1969. doi:10.1137/0706044.

[70] T. Park and J. Light. Unitary quantum time evolution by iterative Lanczos reduction. *J. Chem. Phys.*, 85:5870, 1986. doi:10.1063/1.451548.

[71] U. Peskin, R. Kosloff, and N. Moiseyev. The solution of the time-dependent Schrödinger equation by the (t, t') method: The use of global polynomial propagators for time-dependent Hamiltonians. *J. Chem. Phys.*, 100:8849, 1994. doi:10.1063/1.466739.

[72] W. Press, S. Teukolsky, W. Vetterlin, and B. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, New York, NY, 3rd edition, 2007. Limited access at http://apps.nrbook.com/empanel/index.html#.

[73] M. Reed and B. Simon. *Methods of Modern Mathematical Physics. I: Functional Analysis*. Academic Press, New York, NY, 2nd edition, 1980.

[74] M. Reed and B. Simon. *Methods of Modern Mathematical Physics. II: Fourier-Analysis, Self-Adjointness*. Academic Press, New York, NY, 1975.

[75] G. Scherer. *On energy estimates for difference approximations to hyperbolic partial differential equations*. PhD thesis, Department of Computer Sciences, Uppsala University, 1977.

[76] J. Shen, T. Tang, and L.-L. Wang. *Spectral Methods: Algorithms, Analysis and Applications*. Springer, Berlin, 2011. doi:10.1007/978-3-540-71041-7.

[77] J. Shen and L.-L. Wang. Sparse Spectral Approximations of High-Dimensional Problems Based on Hyperbolic Cross. *SIAM J. Numer. Anal.*, 48:1087–1109, 2010. doi:10.1137/090765547.

[78] J. Shen and H. Yu. Efficient Spectral Sparse Grid Methods and Applications to High-Dimensional Elliptic Problems. *SIAM J. Sci. Comput.*, 32:3228–3250, 2010. doi:10.1137/100787842.

[79] J. Shen and H. Yu. Efficient Spectral Sparse Grid Methods and Applications to High-Dimensional Elliptic Equations II. Unbounded Domains. *SIAM J. Sci. Comput.*, 34:A1141–A1164, 2012. doi:10.1137/110834950.

[80] N. Smith (editor). *The Poems of Andrew Marvell*. Pearson Education Ltd., Harlow, 2007.

[81] S. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963. Russian original in *Dokl. Akad. Nauk SSSR*, 148:1042–1045, 1963.

[82] G. Szegő. *Orthogonal Polynomials*. American Mathematical Society, Providence, RI, 4th edition, 1975.

[83] E. Tadmor. The Exponential Accuracy of Fourier and Chebyshev Differencing Methods. *SIAM J. Numer. Anal.*, 23:1–10, 1986. doi:10.1137/0723001.

[84] V. Temlyakov. *Approximation of Periodic Functions*. Nova Science, New York, NY, 1993.

[85] B. Thaller. *Visual Quantum Mechanics*. Springer, New York, corrected 2nd printing, 2002. doi:10.1007/b98962.

[86] V. Thomée. *Galerkin Finite Element Methods for Parabolic Problems*. Springer, Berlin, 2nd edition, 1997. doi:10.1007/3-540-33122-0.

[87] A. Townsend and S. Olver. The automatic solution of partial differential equations using a global spectral method. *J. Comput. Phys.*, 299:106–123, 2015. doi:10.1016/j.jcp.2015.06.031.

[88] L. Trefethen. *Spectral Methods in MATLAB*. SIAM, Philadelphia, PA, 2000. doi:10.1137/1.9780898719598.

[89] Ch. Van Loan. The Sensitivity of the Matrix Exponential. *SIAM J. Numer. Anal.*, 14:971–981, 1977. doi:10.1137/0714065.

[90] X. Wang and T. Carrington. The Utility of Constraining Basis Function Indices when using the Lanczos Algorithm to Calculate Vibrational Energy Levels. *J. Phys. Chem. A*, 105:2575–2581, 2001. doi:10.1021/jp003792s.

[91] Ch. Zenger. Sparse grids. In: W. Hackbusch (editor), *Parallel Algorithms for Partial Differential Equations*, Notes on Numerical Fluid Mechanics 31. Vieweg, Braunschweig, 1991, 241–251.