UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 1199

Adaptive Solvers for High-Dimensional PDE Problems on Clusters of Multicore Processors

MAGNUS GRANDIN

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2014

ISSN 1651-6214
ISBN 978-91-554-9095-9
urn:nbn:se:uu:diva-234984

Dissertation presented at Uppsala University to be publicly examined in Room 2446, Polacksbacken, Lägerhyddsvägen 2, Uppsala, Friday, 12 December 2014 at 10:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Michael Bader (Institut für Informatik, TU München, Germany).

Abstract
Grandin, M. 2014. Adaptive Solvers for High-Dimensional PDE Problems on Clusters of Multicore Processors. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 1199. 34 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-554-9095-9.

Accurate numerical solution of time-dependent, high-dimensional partial differential equations (PDEs) usually requires efficient numerical techniques and massive-scale parallel computing. In this thesis, we implement and evaluate discretization schemes suited for PDEs of higher dimensionality, focusing on high order of accuracy and low computational cost.

Spatial discretization is particularly challenging in higher dimensions. The memory requirements for uniform grids quickly grow out of reach even on large-scale parallel computers. We utilize high-order discretization schemes and implement adaptive mesh refinement on structured hyperrectangular domains in order to reduce the required number of grid points and computational work. We allow for anisotropic (non-uniform) refinement by recursive bisection and show how to construct, manage and load balance such grids efficiently. In our numerical examples, we use finite difference schemes to discretize the PDEs. In the adaptive case we show how a stable discretization can be constructed using SBP-SAT operators. However, our adaptive mesh framework is general and other methods of discretization are viable.

For integration in time, we implement exponential integrators based on the Lanczos/Arnoldi iterative schemes for eigenvalue approximations. Using adaptive time stepping and a truncated Magnus expansion, we attain high levels of accuracy in the solution at low computational cost. We further investigate alternative implementations of the Lanczos algorithm with reduced communication costs.

As an example application problem, we have considered the time-dependent Schrödinger equation (TDSE). We present solvers and results for the solution of the TDSE on equidistant as well as adaptively refined Cartesian grids.

Keywords: adaptive mesh refinement, anisotropic refinement, exponential integrators, Lanczos' algorithm, hybrid parallelization, time-dependent Schrödinger equation

Magnus Grandin, Department of Information Technology, Division of Scientific Computing, Box 337, Uppsala University, SE-751 05 Uppsala, Sweden.

ISSN 1651-6214
ISBN 978-91-554-9095-9
urn:nbn:se:uu:diva-234984 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-234984)

To my family

List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I M. Gustafsson, S. Holmgren. An implementation framework for solving high-dimensional PDEs on massively parallel computers. In G. Kreiss, P. Lötstedt, A. Målqvist, M. Neytcheva (editors). *Numerical Mathematics and Advanced Applications 2009*, pp. 417–424, Springer, Berlin, 2010. [1]
Contributions: The author of this thesis did the implementation work and wrote the manuscript in discussion with the second author of the paper. The ideas were developed in collaboration between the authors.

II M. Gustafsson, K. Kormann, S. Holmgren. Communication-efficient algorithms for numerical quantum dynamics. In K. Jónasson (editor). *Applied Parallel and Scientific Computing*, Lecture Notes in Computer Science, Vol. 7134, pp. 368–378, Springer, Berlin, 2012. [2]
Contributions: The ideas were developed in close collaboration between the authors of the paper. The author of this thesis did most of the implementation work and prepared the manuscript in close collaboration with the second author, in discussion with the third author.

III M. Gustafsson, J. Demmel, S. Holmgren. Numerical evaluation of the Communication-Avoiding Lanczos algorithm. Technical report ISSN 1404-3203/2012–001, Department of Information Technology, Uppsala University, 2012.
Contributions: The ideas were developed in collaboration with the second author. All implementation was done by the author of this thesis. The manuscript was prepared by the author of this thesis in discussion with the other authors of the paper.

[1] With kind permission from Springer Science and Business Media.
[2] With kind permission from Springer Science and Business Media.

IV A. Nissen, K. Kormann, M. Grandin, K. Virta. Stable difference methods for block-oriented adaptive grids. (Submitted to Journal of Scientific Computing, under revision.)
Contributions: The ideas were developed in close collaboration between the authors of the paper. The author of this thesis developed the code infrastructure and did the bulk part of the implementation. The manuscript was written in close collaboration between the authors.

V M. Grandin. Data structures and algorithms for high-dimensional structured adaptive mesh refinement. Technical report ISSN 1404-3203/2014–019, Department of Information Technology, Uppsala University, 2014. (Submitted to Advances in Engineering Software)
Contributions: The author of this thesis is the sole author of this paper and the implementation work.

VI M. Grandin, S. Holmgren. Parallel data structures and algorithms for high-dimensional structured adaptive mesh refinement. Technical report ISSN 1404-3203/2014–020, Department of Information Technology, Uppsala University, 2014.
Contributions: The author of this thesis did the implementation and developed the ideas. The manuscript was prepared in collaboration with the second author.

Related work

Although not explicitly discussed in the comprehensive summary, the following papers are related to the contents of this thesis.

- M. Gustafsson, S. Holmgren. Efficient implementation of a high-dimensional PDE-solver on multicore processors. In *Proceedings of the 2nd Swedish Workshop on Multi-Core Computing*, pp 64-66, Uppsala, Sweden, 2009.

Contents

1. Introduction

Accurate numerical solution of time-dependent partial differential equations (PDEs) can be a demanding task. Even if high-order accurate numerical techniques and adaptive schemes in space and time are used, the number of spatial degrees of freedom often becomes very large. Massive-scale computing resources and parallel software are required in order to solve such problems efficiently. As high-performance computers increase in number of processors, cores and memory capacity, the attainable problem size is constantly growing. Current high-performance computers are built as clusters of multi-core nodes and have introduced challenges due to multi-level parallelism and complex memory hierarchies. Aiming at scalability on massive-scale computers, it is important that the numerical methods are implemented with the underlying parallel architectures in mind.

In this thesis, we focus on numerical solution of high-dimensional PDE problems, i.e. problems that need to be discretized in more than three spatial dimensions. This type of computational problems appear in a wide range of research applications, such as quantum dynamics [70], systems biology [26], plasma physics [35] and financial mathematics [43]. The general class of problems considered are d-dimensional, time-dependent, linear PDEs

$$\frac{\partial u}{\partial t} = \hat{P}(\mathbf{x},t)\,u, \tag{1.1}$$

where $\hat{P}$ is the linear operator describing the dynamics in the model. Discretizing (1.1) in space transforms the PDE into a large semi-discrete system of ordinary differential equations (ODEs), an approach generally referred to as the method of lines. We then have

$$\frac{du}{dt} = P(\mathbf{x},t)\,u, \tag{1.2}$$

where P is the discretized linear operator matrix.

As an example of a high-dimensional PDE-problem, consider the time-dependent Schrödinger equation (TDSE), governing the quantum dynamics of molecular processes,

$$\mathrm{i}\hbar\frac{\partial \psi(\mathbf{x},t)}{\partial t} = H\psi(\mathbf{x},t), \tag{1.3}$$

$$\psi(\mathbf{x},0) = \psi_0, \tag{1.4}$$

for $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and time $t \geq 0$. Here, $\hbar$ is Planck's reduced constant and H is the Hamiltonian operator, describing the kinetic and potential energy of the system,

$$H = -\sum_{i=1}^{d} \frac{\hbar^2}{2m_i} \frac{\partial^2}{\partial x_i^2} + V(\mathbf{x}, t), \tag{1.5}$$

where m_i is the mass corresponding to dimension i, and V the potential modeling interactions within the system. The solution to equation (1.3), ψ, is referred to as the wave function and its L_2 norm is finite and preserved. Here, the problem rapidly becomes high-dimensional when the number of particles N in the system studied is increased.

If a standard grid-based discretization technique is used for solving a high-dimensional PDE problem the number of grid points will grow exponentionally with the number of spatial dimensions. As a consequence, the total grid size quickly grows prohibitively large, already for a moderate number of dimensions. On today's medium sized clusters (on the order of 5000 cores), accurate simulation for problems of up to six or seven dimensions may be possible. Using parallel computing and adaptive schemes, as described in this thesis, a few more dimensions can potentially be added and the computational time can be reduced. In order to approach problems with even larger number of spatial dimensions, modelling approaches and approximations which explicitly reduce the dimensionality must be used. These approximations can be considered as application-specific pre-processing transformations, which we do not consider any further in this thesis.

2. Spatial discretization methods

When discretizing a PDE in the spatial domain using mesh based techniques, there are two principal classes of methods to choose from; structured and unstructured methods. Unstructured methods can easily adapt to complex geometries, whereas structured methods usually are limited to simpler domains; however, structured methods are often more efficient than unstructured methods when they can be used [3]. In this thesis, we focus on problems that can be posed on (or mapped to) Cartesian hyperrectangular domains and structured meshes. We demonstrate examples using finite difference methods, but our framework is general enough to allow other methods of discretization to be used. In this chapter we discuss the discretization techniques that we consider within our framework, with adaptive mesh refinement in mind.

The flexibility of unstructured meshes comes with a cost; there are significant overheads associated with storage and memory access indirection due to the inherent lack of structure in the way grid data is stored [3, 50]. The coordinates of each mesh point (degree of freedom) must be stored explicitly and mesh connectivity meta information is required in order to navigate the mesh. On the contrary, structured grids feature implicit information regarding the positions of grid points and their relations to one another. Structured rectangular domains naturally map to tensor-product grids; i.e. grids where any grid point is uniquely identified by a d-vector of integer indices $(j_1, j_2, \ldots, j_d)$ which are used for direct adressing. Accessing a grid point in relation to another (e.g. a direct neighbor of a grid point) is straightforward since we just need to manipulate the corresponding indices in order to obtain the grid point of interest. Furthermore, unstructured methods usually require additional computational steps, such as matrix assembly in finite elements, that are not needed with direct structured methods. Thus, for performance reasons, structured grids are to be preferred over unstructured grids when they can be used.

Nevertheless, some methods originally developed for unstructured meshes have properties that make them numerically attractive also on structured (adaptive) meshes. If the assumption is made that the mesh is structured, computations can be arranged such that much of the overhead associated with the unstructured representation is removed. Kronbichler and Kormann [41] demonstrate how this can be done efficiently for high-order finite element methods.

2.1 Finite difference methods

Finite difference approximations are conceptually straightforward and simple to implement, yet they have competitive numerical properties. Commonly,

finite difference operators are interpreted as stencils, where a weighted sum of close-by grid function values is computed at each grid point, approximating the value of the derivative in that point. A particularly attractive feature of finite difference methods is their ability to be extended to arbitrarily high orders of accuracy [27, 31]. An operator that arises from a finite difference discretization can be viewed either as a sparse band-diagonal matrix or as a stencil operator applied in each individual grid point. In the case of a matrix representation, the bandwidth of the matrix increases with the order of accuracy in the approximations[1]. For a stencil representation, the width of the stencil will grow accordingly. In either case, computing the differential approximation corresponds to multiplying a sparse matrix with a dense vector (sparse matrix-vector multiplication, or SpMV). In this work we focus on the use of finite difference operators on stencil form.

Due to the locality of finite difference approximations it is rather straightforward to parallelize the stencil operation. A standard way of doing this is to partition the computational grid in a number of slices or grid blocks and let each process compute its contribution on its partition [25]. This strategy can be applied recursively to parallelize the computations within each process over a number of threads [10]. Communication among processes is overlapped with computation by letting each process work on the inner parts of its subdomain while interprocess data is being transmitted. However, the surface-to-volume ratio of each hyperrectangular subdomain increases with the number of dimensions and the corresponding communication volume grows correspondingly [19]. The situation is worsened due to the increasing width of the stencils for higher orders of accuracy. Thus, at some point the communication volume will grow too large to be overlapped with communication, and performance will diminish.

The data access pattern of a stencil-like operator becomes problematic with higher-dimensional grids, due to the way data are stored. Computers exhibit a linear view of its memory locations. Thus, in order to store a d-dimensional array in memory, it must be mapped to a one-dimensional index base. For instance (assuming row-major order), when the elements of a two-dimensional array are accessed in order, there will be a stride of unit length between the elements in each row and a stride equal to the row length between each element in a column. Extending this to higher dimensions, the stride between consecutive elements in a particular dimension would be the product of the grid length in all the lower dimensions. Modern cache-based computers exploit temporal locality of data accesses based on the idea that if some data is used at some point, it is expected that it will be used again in a near future. In order to avoid having to fetch the data repeatedly from memory, wasting many clock cycles of latency for each access, the data is stored temporarily in a much faster cache

[1]For general sparse matrices, transformations such as the Cuthill-McKee algorithm can be used to reduce the matrix to a more condensed band matrix [17, 44].

memory. However, due to the small size of the cache memory, it is not possible to store a large number of data values. If the span of memory locations that need to be accessed repeatedly is larger than the size of the cache, the data that has been stored in the cache will constantly be overwritten by new accesses. In the end, this will lead to poor overall performance since a lot of data has to be constantly reloaded from the slow memory. In order to improve cache-reuse for stencil computations, optimizations such as cache-blocking, cache-tiling and time-skewing are often used [18, 19, 20, 60, 61]. However, in our attempts at optimizing this operation for cache performance, we have not been able to achieve any significant speedup for $d > 4$ using these approaches to stencil optimization. This is in part due to the large strides in consecutive data accesses as discussed above, but also due to the impact of the surface-to-volume ratio [19].

2.2 Summation-by-parts operators

Summation-by-parts (SBP) operators are finite difference schemes with central stencils in the interior and one-sided stencils on the boundaries of a (sub) domain, constructed such that they mimic the properties of the continuous operators [40, 49, 64]. Combining the SBP discretization with simultaneous approximating term (SAT) weak boundary conditions leads to provably time-stable numerical approximations [12, 48]. The combined scheme, referred to as SBP-SAT, can be used in single- as well as in multi-block configurations [48]. In all essential detail, SBP-SAT operators can be implemented as stencils in accordance with how standard finite differences are implemented. However, the one-sided differential operators close to block boundaries as well as the penalty terms and interpolation operators are much more complex than the central differences. Furthermore, the width of the region that the stencils extend outside of the blocks (the "ghost zones") is considerably higher with SBP-SAT than with standard finite differences. This will likely become a scalability hurdle when approaching larger problems in higher dimensions.

2.3 Finite element methods

Although primarily used on unstructured meshes, finite element methods can successfully and efficiently be implemented on structured meshes (c.f. [41] and references therein). Special types of high-order finite elements, spectral elements [36, 54], have properties that make them particularly appealing for use on Cartesian grids. An example of such elements are Lagrange elements with support points according to the Gauss-Lobatto quadrature rule [38]. This choice leads to well-conditioned elements of high order, and due to the nodal placement the mass matrix becomes diagonal and trivial to invert. Kron-

bichler and Kormann [41] combined this with a local quadrature approach for a matrix-free finite element implementation using a sum-factorization approach [51, 54]. The local quadrature has a tensor-product structure almost as narrow as finite difference stencils [38]. Implemented in this way, the major obstacles that are associated with finite element methods on unstructured meshes, matrix assembly and unstructured storage of mesh nodes, can be overcome and the final scheme is performance-wise competitive compared to a finite difference implementation. A benefit with this approach compared to high-order finite difference methods (in particular compared to SBP-SAT) is that the width of the ghost-zone around each element is considerably smaller. This will lead to lower communication costs in a parallel implementation which makes this type of discretization appealing for use in our high-dimensional solver (although we have not implemented it yet).

3. Structured adaptive mesh refinement

The main goal of adaptive mesh refinement is to increase the computational efficiency by adjusting the mesh dynamically according to the propagation of the solution. By increasing the mesh resolution locally in areas where the solution exhibits certain features (e.g. steep gradients, shocks or fluctuations) we are able to achieve high accuracy while keeping the overall number of mesh elements at a tractable level. The challenge is to construct a mesh that meets the desired accuracy in the solution using as few mesh points as possible, and to manage the mesh with low overhead in terms of memory and runtime costs. Furthermore, for the scheme to be of practical use in simulations over long time intervalls, the mesh should be updated automatically as the solution is propagated in time by the solver, introducing new grid points in regions where finer resolution is needed and removing grid points in regions where coarser resolution would suffice.

Adaptive mesh refinement is a general concept that can be implemented in numerous ways. Unstructured meshes can straightforwardly be tailored for any geometry, increasing the resolution by introducing more mesh nodes as is required. There is no implicit order among the mesh elements, so a large amount of meta data must be stored for keeping track of the mesh points and their relationships. A structured mesh, on the other hand, implies a specific order among the mesh elements. Given a point in space, we can find any of its surrounding points based on its index. Due to the performance advantage of exploiting the structure in the domain decomposition, *structured adaptive mesh refinement* (SAMR) has become a separate branch of research. In this chapter, we will briefly review the history of SAMR and describe the specific contributions we have made to the field concerning SAMR for PDE problems of higher dimensionality.

3.1 SAMR overview

Structured adaptive mesh refinement was pioneered in a series of papers by Berger and Oliger [7, 8]. In their original method, adaptive meshes are constructed by superimposing rectangular patches with finer resolution onto underlying coarser grids. Patches are allowed to be oriented arbitrarily with respect to rotation and size, which allows for flexibility and accuracy in the generated grids; for instance, a grid patch can be rotated locally in order to be perfectly aligned with a wave front or shock. However, this flexibility also

infers significant overhead due to frequent grid rotations and transformations. In a subsequent paper, Berger and Colella [5] simplified the Berger-Oliger scheme such that all grid patches were restricted to be aligned with respect to a common Cartesian grid. This restriction significantly reduces the grid management overheads in composite grids, without severely impacting grid flexibility aspects. The Berger-Colella approach has been widely adopted and inspired many attempts to implement it (c.f. [6, 16, 22, 69]).

Block-oriented SAMR [24, 45, 46, 57] was proposed as a further simplification of the Berger-Colella approach. In this class of methods, grid patches are required to conform to a block-decomposition of the grid and refinement is performed block-wise rather than point-wise. Due to this, grid management overheads are further reduced since the grid-fitting and clustering steps are no longer needed [59]. Instead of marking individual grid points for refinement, a whole block is marked if one or more points within it needs refinement. Two alternatives have been proposed as for how refinement should take place; more gridpoints can be introduced either by increasing the number of grid points within each block (keeping the number of grid blocks constant) or by generating more grid blocks (keeping the number of grid points per block constant). In the latter case, by introducing more blocks to the mesh were refinement is needed, the effective resolution in those parts of the grid is increased. The subdomain that is represented by the conjunction of the superimposed patches then corresponds exactly to the subdomain represented by the coarser patch.

In this thesis, we focus on the case of block-oriented SAMR where all blocks are of equal size in terms of grid points. In the remainder of this text we will simply refer to this type of mesh representation as block-oriented SAMR. Moreover, we implement refinement by recursive bisection on hyperrectangular grids. This potentially leads to grids with anisotropic mesh elements, i.e. blocks with different levels of refinement in different dimensions. As opposed to block-oriented refinement on isotropic elements (e.g. [11, 24, 46, 65]), the anisotropic strategy is expected to generate far fewer elements when the number of dimensions grows large, since we avoid refinement in dimensions where higher resolution is not needed. The way we store and propagate our grids is different from how SAMR is traditionally done, although the foundation is similar to other block-oriented SAMR attempts. In particular, no data is stored for the internal nodes in the mesh hierarchy, only leaf-level blocks carry mesh data.

3.2 Mesh management

Block-oriented adaptive mesh refinement further simplifies matters when it comes to storing and distributing the mesh compared to the Berger-Colella approach. Since no grid fitting or clustering of grid points is needed, the mesh decomposition can be represented as a hierarchy of exactly overlapping mesh

blocks. This hierarchy of blocks is commonly stored as a regular refinement tree. With isotropic refinement and a refinement factor of two in each refinement, quadtrees and octrees (and their higher-dimensional counterparts, 2^d-trees in general) are commonly used. Other attempts use different refinement factors (e.g. [67] with 3^d-trees as a result) but the overall strategies and tree structures are similar. We refer to the number of branched nodes at each level in a refinement tree as the *fan-out* of that particular tree structure. Another approach of grid representation is to store the refinements dimension-by-dimension in a binary/ternary tree. This way, the fan-out of each refinement is 2 or 3 respectively, and anisotropic refinement is possible to achieve. We use a binary *kd*-tree [4] representation of the refinement throughout this thesis.

Hierarchical data structures such as a refinement tree are often conceptualized using pointers to connect nodes between different tree levels. However, scaling such a data structure towards thousands of processors is not easily accomplished. A more promising approach is to use a linear representation and distribute the leaf nodes in a decoupled manner based on their locality information and a linear ordering [11, 15, 30, 65]. If encoded properly, the locality information of a given node in the mesh can be used to find locality information of mesh elements and their relations to other mesh entities. Furthermore, process ownership of mesh nodes can be discovered efficiently for retrieval of remote mesh data (c.f. [11] and Paper VI for details). In this linear storage of mesh data, tree traversal is replaced with binary search in a linear array which is beneficial from a perspective of algorithmic complexity; in an adaptively refined mesh, tree search is worst-case $\mathcal{O}(n)$, whereas binary search in linear storage is always $\mathcal{O}(\log_2 n)$.

3.3 Dynamic load balancing

As the mesh is adapted to the solution, data dependencies are introduced within and between the patches of the mesh. Since these data-dependencies are not known a priori, they must be accounted for during the adaptation step. Furthermore, the data dependencies change dynamically as the mesh is propagated in time with the solution. Therefore, an important aspect of any dynamically adaptive implementation is to exploit locality of data and balance the workload evenly among the processors at runtime. The load balancing aspect becomes particularly important on large scale parallel computers but as parallelism increases everywhere (even in desktop computers and laptops), workload distribution needs to be considered at all levels.

In block-oriented SAMR, space-filling curves are an indespensable tool in guiding the partitioning. Space-filling curves [2, 66] are recursive locality-preserving mappings from a multidimensional order to a linear order. There are several types of space-filling curves with varying complexity and characteristics. For load-balancing purposes of (hyper)rectangular domains, the

Hilbert, Morton and Peano orderings are commonly used [2, 23, 56]. On decompositions based on 2^d trees, the Hilbert and Morton space filling curves are viable choices. The Morton ordering (often referred to as z-order due to its characteristic z form) is simpler to derive than the Hilbert ordering but Hilbert ordering preserves locality better. The Peano ordering is exclusively used on 3^d trees; since we only consider binary trees in this thesis, we do not discuss the Peano order any further in this thesis.

Once the multidimensional decomposition has been linearized with a space filling curve, we can balance the workload by distributing portions of the linear order amongst the processors according to some partitioning strategy. If different mesh elements are associated with varying amount of computational work, the partitioning can be complemented with assigning weights to the mesh elements [11, 46]. Often, the simplest strategy where each process is assigned an equal chunk of the linear order is considered to generate a good enough partitioning [11, 24]. In other cases, it is desirable to generate partitionings that minimize the inter-processor communication volumes by modifying the partitioning boundaries [56, 65]. E.g. Sundar *et al.* [65] build upon the idea that the inter-processor communication volumes are affected by the "geometrical shape" of each respective processor's partition. In their scheme, the coarser levels of the refinement dictate the distribution such that the inter-processor boundaries are simplified to some degree. This strategy decreases the inter-processor communication costs at the expense of a suboptimal work load distribution.

4. Exponential integrators for time propagation

If P is independent of time, the general solution to (1.2) is given by

$$u(t) = e^{Pt}u(0). \tag{4.1}$$

On the other hand, if P is time-dependent the system of ODEs cannot be solved exactly using (4.1). However, the exponential formulation can be used successively on small time intervals if the time-dependent operator is expanded using the Magnus expansion [47]. At time step k we have

$$u_{k+1} = e^{\Omega_k}u_k, \tag{4.2}$$

where Ω_k is a sum of nested commutators of P, evaluated at different temporal locations. Truncating after the first term yields

$$\Omega_k = h_t P\left(t_k + \frac{h_t}{2}\right), \tag{4.3}$$

corresponding to the second-order accurate exponential midpoint rule. Here, h_t is the time step size and t_k is the time at time interval k. For the Schrödinger equation, Hochbruck and Lubich [33] developed efficient time-integration techniques using the Magnus expansion and exponential integrators. Furthermore, Kormann *et al.* [39] demonstrated how a truncated Magnus expansion was used to develop an efficient and accurate h,p-adaptive time-integrator with global error control.

The complexity of typical methods for explicitly computing the exponential of a general $n \times n$ matrix A is $O(n^3)$ [52]. Exploiting sparsity and structure of the matrix helps reducing the computational cost, but in general the exponential e^A is dense even if the matrix A is sparse and it is not feasible to compute and store the full exponential for large problems [52]. Instead, for large sparse problems, a common approach is to compute an approximate matrix exponential using Krylov subspace methods for eigendecomposition [52, 62]. The Lanczos algorithm [42] (Hermitian matrices) or the Arnoldi algorithm [1] (general matrices) are commonly used for this purpose, computing an approximate eigenvalue decomposition by projecting the original problem onto a Krylov subspace of much lower dimension.

Let A be an $n \times n$ matrix and v_1 a normalized vector of length n. Then, after m iterations of Lanczos/Arnoldi with v_1 as initial vector, an orthonormal basis

V_m^A spanning the Krylov subspace $\mathscr{K}_m(A, v_1)$ has been generated and along with it a corresponding projection matrix H_m^A, satisfying the relation

$$AV_m^A = V_m^A H_m^A + h_{m+1,m} v_{m+1} e_m^T. \tag{4.4}$$

The eigenvalues of H_m^A are approximate eigenvalues of A (Ritz values) and the corresponding eigenvectors of A (Ritz vectors) can be generated from the basis V_m^A and each respective eigenvector of H_m^A (c.f. Paper IV for more details on Ritz values and Ritz vectors).

Rewriting (4.2) in terms of the resulting Krylov basis, we obtain the the following update for u_{k+1} (c.f. [52, 62])

$$u_{k+1} = V_m^{\Omega_k} \exp\left(H_m^{\Omega_k}\right) e_1 ||u_k||_2, \tag{4.5}$$

where $V_m^{\Omega_k}$ is the orthonormal basis spanning the Krylov subspace $\mathscr{K}_m(\Omega_k, u_k)$, $H_m^{\Omega_k}$ the corresponding projection matrix of Ω_k onto $\mathscr{K}_m$ and e_1 is the first column of the unit matrix I_m.

5. Implementation

The work within the scope of this thesis has focused on prototyping different solution methods and implementations for solving high-dimensional PDE problems. In this chapter we give a summarizing overview of the contributions we have made in the different sub-projects of this thesis.

5.1 The HAParaNDA project

HAParaNDA (*High-dimensional Adaptive Parallel Numerical Dynamics Applicaions framework*) is a long term project for developing scalable numerical software suited for the solution of high-dimensional PDEs on massive-scale clusters. The goal is to provide application experts with tools that enable flexible, simple and intuitive implementation of solvers for complex problems, relaxing them from low level implementation details and performance issues.

5.2 An example application solver

In Papers I and II, the implementation of a large scale parallel solver for the time-dependent Schrödinger equation is presented. We discretize the d-dimensional space onto an equidistant Cartesian grid which is distributed in equally sized blocks among the processes. High-order finite difference methods are used for approximation of the spatial derivatives. For temporal discretization, we use exponential integrators based on the Lanczos algorithm, as described in Chapter 4.

Here, we parallelize at two levels; between and within nodes. Within each multicore node we use OpenMP for multi-threaded work sharing of compute-intensive loops. Between compute nodes we use MPI (Message Passing Interface) for distribution of data and synchronization of computations among the processors. This type of hybrid parallelization has advantages compared to an MPI-only implementation, such as better memory efficiency, better mapping to the multicore architecture in the nodes and better scalability [58].

We use a block-structured decomposition of the grid, subdividing the computational domain into equally-sized blocks and distributing them to the compute nodes. The stencil operation (as described in Section 2.1) is parallelized locally in each node by slicing the corresponding block in the least unit-stride

dimension. Along block boundaries, the data dependencies of the finite difference stencil will require neighboring blocks to exchange the data values that are closest to the corresponding boundary. The number of data values that need to be sent depends on the order of the finite difference approximations; in general p layers of data are affected, where the order of accuracy is $2p$. In Paper I we describe an algorithm for exchanging data one dimension at a time in order to reduce the memory usage overhead due to data duplication. This turned out *not* to be a succesful strategy when approaching larger scale clusters. This was particularly noticable in combination with the s-step matrix powers kernel that we used in Paper II. Therefore, we abandoned this idea and instead store explicit buffers for each spatial direction and initiate communication in all dimensions at once. The communication latency in the send/recv operation is overlapped with the local computations that are not affected by the data on block boundaries.

5.3 Communication-efficient time propagation

Analyzing the iterative steps of the Lanczos algorithm (see e.g. Algorithm 1 in paper III), we notice that in each iteration a sparse matrix-vector (SpMV) multiplication is computed, followed by orthogonalization by means of scalar products and vector updates. In a large scale implementation with distributed vectors of a large number of unknowns, the vector operations will require significant data traffic and the scalar products will impose global synchronization points that may hamper parallel scalability. This problem was noticed by Kim and Chronopoulos [37] and they proposed two variations of the Lanczos algorithm that aim at improving the parallelism of the iterative scheme; the few-synch Lanczos algorithm and the s-step Lanczos algorithm. In Paper II, we implement both their algorithms within our framework for time propagation by exponential integration and compare their performance to the classical Lanczos scheme. We further extend the application of the few-synch algorithm and implement a hybrid scheme that avoids the extra iteration that is added compared to the classical Lanczos scheme. The s-step algorithm has particularly attractive features, as it computes s consecutive SpMVs in a matrix-powers kernel $\left[v, Av, A^2v, \ldots, A^sv\right]$ and orthogonalizes the resulting basis vectors using matrix operations rather than vector operations. Our findings in Paper II show that the overall difference between the implementations is not very significant. Indeed, communication is avoided in the s-step algorithm; however, as it computes an extra SpMV in each outer iteration, its potential performance improvements diminish in practice. Furthermore, as was noted in their paper [37] and later by others [13, 14, 34], the size of the inner Krylov kernel, s, is restricted for stability reasons and in most practical computations the s-step Lanczos algorithm is limited to $s < 5$.

Other, more recent attempts at remedying the parallel scalability problems with the Lanzos algorithm have been proposed; Hoemmen [34] formulates the Communication-Avoiding Lanczos algorithm (CA-Lanczos) and Carson and Demmel [13, 14] propose a novel s-step Lanczos formulation with improved numerical properties. These build upon the idea of computing s SpMVs in a matrix kernel, but orthogonalize the basis vectors using more stable methods.

In Paper III, we implement a MATLAB-code for evaluation of the numerical behaviour of the CA-Lanczos algorithm on a number of test problems. We implement several techniques that are often used to control the sometimes peculiar convergence behaviour of the Lanczos iterations; namely periodic orthogonalization [63], selective orthogonalization [55] and explicit restart [32]. An important improvement of the CA-Lanczos algorithm compared to the s-step algorithm of Kim and Chronopoulos is that CA-Lanczos is not restricted to be used with a monomial basis in the matrix-powers kernel. We include the Newton basis in our numerical experiments in Paper III and conclude that with proper reorthogonalization and choice of basis, the CA-Lanczos algorithm performs well in practice, even with rather large step sizes in the inner Krylov kernel ($s = 8$ and $s = 12$ are often viable choices). We have not yet implemented the CA-Lanczos algorithm in our exponential integration framework, but its numerical performance makes it a promising alternative.

5.4 Adaptive mesh refinement

We argue that the fan-out factors of $2^d/3^d$-trees (as discussed in Chapter 3) will lead to unmanageable tree structures and problem sizes as dimensionality is increased. To alleviate this, we support anisotropic refinement and implement the refinement tree as a kd-tree, i.e., a binary tree representation of a hierarchical subdivision of a d-dimensional hyperrectangle by recursive bisection [4, 28, 29, 68]. In this framework, mesh blocks are allowed to be refined any number of times in a single dimension before refinement is done in any of the other dimensions; thus, mesh blocks may become arbitrarily anisotropic with no limitations on the aspect ratio of a block.

In Paper IV, we describe a prototype MATLAB implementation of a block-adaptive solver using the SBP-SAT technique for coupling non-conforming grid blocks. In this paper, we show how to fit grids to given functions based on a dimension-wise error estimate and how to impose necessary restrictions on the resulting grid. We use binary kd-trees for organizing the grid and a 2:1 refinement ratio between face-neighboring blocks. Between blocks that are conforming, we use central finite differences. This paper is an extension of the work by Nissen *et al.* [53] where stability was proven for multi-block grids. We prove stability also for more elaborate grid structures and show that our method works for longer time integrations.

In order to approach larger problems in higher dimensions, we have implemented a C++ framework for handling anisotropic meshes. We describe the fundamental algorithmic details of our refinement scheme in Paper V, whereas a distributed parallel extension of the framework is described in Paper VI. In this framework, we use a linear representation of leaf nodes to store the *kd*-tree. The nodes are stored in *hierarchical Morton order*, a modified Morton order where the bit-interleaving is altered in order to correspond to the anisotropic mesh structure. In order to represent the hierarchical information and guide the ordering of leaf nodes, we store a node index map in addition to the leaf nodes. This is a lightweight representation of the internal structure of the *kd*-tree, implemented as a linear depth-first search tree, requiring only two integers per internal node for storage. By assigning each leaf node an integer index based on the node index map, we create a linear order of leaf nodes. Since the node index map is a small, sequential data structure, retrieving node index information is a fast operation.

In order to parallelize the *kd*-tree structure onto multiple processes, we partition the nodes according to the linear order in equal chunks, and assign each chunk to a process. Load imbalance estimation and repartitioning are based on a prefix-sum of the node counts in each process [9, 21]. Repartioning the nodes is a process-local operation, requiring each process to exchange data only with its left/right neighbor (i.e., process rank r exchanges data only with ranks $r \pm 1$).

Each process stores its own local sub tree. A sub tree is a complete independent entity, storing a set of leaf nodes and a corresponding index map. The index map of a subtree is truncated to only account for the leaf nodes it represents. Upon communication of nodes from a process to another, the parts of the local tree to be sent are pruned off and sent to the corresponding recipient. When a sub tree is received, it is merged into the local tree of the recipient process.

6. Svensk sammanfattning

Många problemställningar inom beräkningsvetenskap och dess tillämpningar leder till partiella differentialekvationer som måste diskretiseras i högre dimensioner. Exempel på sådana problemställningar är Schrödingerekvationen inom kvantdynamik, Black-Scholes ekvation inom finansiell matematik och plasmafysikaliska beräkningar. Med högre dimensioner avses här fler än tre rumsdimensioner. Denna typ av applikationer är problematiska främst på grund av de problemstorlekar som måste hanteras; exponentiell tillväxt i antalet obekanta då dimensionsantalet ökas gör att även problem som är små i varje enskild dimension totalt sett kan bli väldigt krävande eller rent av ohanterliga. För att möjliggöra lösningen av större problem använder vi oss av numeriska metoder med hög noggrannhetsordning och adaptiv steglängd i tiden och rummet. I denna avhandling behandlar vi lösningsmetoder för en generell klass av tidsberoende, linjära partiella differentialekvationer i d dimensioner. Vi använder oss av tidsberoende Schrödingerekvationen som ett exempel på problem vi kan lösa, men de tekniker vi beskriver kan användas även för andra linjära problem.

Genom att parallellisera problemlösningen och sprida ut den på flera beräkningsnoder i ett kluster kan vi räkna på större problem eftersom vi då får totalt sett mer processorkraft och större minnesmängd att arbeta med. Mjukvarubibilioteket MPI (Message Passing Interface) är ett verktyg som används för att få beräkningsnoderna att kommunicera med varandra, vilket är nödvändigt då de inte delar gemensamt minnesutrymme och inte direkt kan utbyta data med varandra. Vidare kan vi parallelisera arbetet inom varje beräkningsnod med hjälp av trådar (vi använder OpenMP) för att utnyttja alla tillgängliga processorkärnor. På detta sätt kan vi undvika duplicering av data och den explicita kommunikation som det innebär att skicka data mellan MPI-processer.

Adaptivitet är ett gemensamt begrepp för numeriska metoder som automatiskt anpassar sig efter lösningen i tiden eller rummet för att maximera utnyttjandet av tillgängliga beräkningsresurser. Adaptiv nätförfining strävar efter att anpassa rumsnätet efter artefakter i lösningen, så att de delar av rummet där lösningen är mer krävande får en finare diskretisering än övriga delar av rummet. I denna avhandling fokuserar vi på problem över hyperrektangulära kartesiska nät och använder strukturerad nätförfining för att anpassa nätet.

Metoder för strukturerad nätförfining skiljs åt beroende på hur näthierarkierna hanteras. I ursprungsformen av strukturerad nätförfining består en näthierarki av flera nivåer av överlappande nät av olika förfining, där lösningen sparas och propageras separat på varje nivå. Vi använder oss av en förenklad variant av strukturerad nätförfining, block-orienterad nätförfining, som begränsar

hur de överlappande näten kan placeras och orienteras. Nätet representeras med hjälp av block. Om ett block förfinas ersätts det av ett antal nya block som tillsammans motsvarar exakt det område som det ursprungliga blocket täcker. Eftersom alla block representerar samma antal nätpunkter kommer de nya blocken att svara mot en finare diskretisering än det ursprungliga blocket.

De metoder som traditionellt används för block-orienterad nätförfining i två och tre dimensioner är svåra att utöka till högre dimensioner. Detta beror främst på att nätförfiningen i dessa metoder normalt sker likformigt i all dimensioner (isotrop förfining). Om förfiningsfaktor 2 antas kommer således 2^d nya block att genereras vid varje förfining. I högre dimensioner riskerar utväxlingen av sådan förfining att leda till ohanterliga problemstorlekar. Vi beskriver hur förfining kan göras anisotropt (olikformigt) med hjälp av rekursiv binär uppdelning. På så sätt kan vi minimera antalet genererade block, vilket i sin tur leder till ett minskat antal nätpunkter jämfört med isotrop förfining. Vi redogör i detalj för hur sådana nät kan konstrueras, hanteras och fördelas mellan processerna i en parallell miljö.

För integration i tiden använder vi en explicit metod där den successiva lösningen formuleras som en exponential av den tidsberoende diskretiseringsmatrisen över korta tidsintervall. Då exponentialen av en stor matris är väldig kostsam att räkna ut (om det ens är möjligt) väljer vi att approximera matrisexponentialen med hjälp av Krylovmetoder. Detta är en övergripande klass av iterativa metoder som successivt räknar fram en (approximativ) projektion av lösningen på ett vektorrum av betydligt lägre dimension än den fulla lösningen. För exponentialintegrering behöver vi ett antal av de mest dominerande egenvärdena och motsvarande egenvektorer av diskretiseringsmatrisen. Beroende på problemets struktur kan då antingen Lanczos metod (symmetriska problem) eller Arnoldis metod (generella problem) användas.

Krylovmetoder är ofta effektiva såtillvida att de approximerar lösningen över förhållandevis få iterationer och den totala beräkningsmängden är ofta låg. Ett problem med dessa metoder är dock att det förfarande med vilket basvektorerna tas fram leder till stora mängder kommunikation och synkronisering i en parallell implementation. Detta problem har studerats i stor utsträckning och flera studier föreslår lösningar till hur iterationerna kan struktureras om för att beräkningarna ska göras effektivare. Vi studerar och jämför flera alternativa formuleringar av Lanczos algoritm och ser hur de beter sig numeriskt och prestandamässigt.

Vi redovisar numeriska resultat för beräkningar av tidsberoende Schrödingerekvationen och ser att våra metoder ger önskvärd noggrannhet och beräkningsmässig effektivitet, med parallell skalning över 4000 beräkningskärnor. Numeriska resultat redovisas för problem i upp till 5 dimensioner, och vi visar att vi kan hantera högdimensionella adaptiva nät. Då våra metoder och implementationer är generella i antalet dimensioner är det beräkningsresurserna som sätter gränserna för de problem vi kan hantera.

Acknowledgements

I am very grateful to my advisor, Sverker Holmgren, for all the support and advice you have given me during this time. I would also like to thank my co-advisor, Michael Thuné, for your involvement and guidance. Thanks to Elisabeth Larsson for your support and travel companionship.

Further, I would like to give a big reach of thanks to my co-authors; Katharina Kormann, Anna Nissen, Kristoffer Virta and James Demmel. It has been a pleasure working with you! Thanks to Malin Källén for bringing my project on in the future.

To all of you at TDB, thank you for creating a good work environment. You will be missed. Extra huge hugs to Martin Tillenius, Sofia Eriksson, Jens Berg, Erik Lehto, Katharina Kormann, Martin Kronbichler, Stefan Hellander, Pavol Bauer and Sven-Erik Ekström.

Martina and Vanja, your endless love and support has helped me all the way. Mamma, Pappa, Mia, Sofia and Niclas, thank you for always being there.

Part of the work in Paper III was accomplished during a 10 week visit at University of California, Berkeley. Many thanks to James Demmel and the other members of the Berkeley Benchmark and Optimization group for hosting me during these weeks.

Travel support from Anna Maria Lundins stipendiefond, Ångpanneföreningen and Stiftelsen Lars Hiertas minne has made my travels possible.

References

[1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29, 1951.

[2] M. Bader. *Space-Filling Curves: An Introduction with Applications in Scientific Computing*, volume 9. Springer Berlin Heidelberg, 2013.

[3] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws. *SIAM Journal on Scientific Computing*, 15(1):127–138, 1994.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.

[6] M. Berger and R. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

[7] M. J. Berger. Data structures for adaptive grid generation. *SIAM Journal on Scientific and Statistical Computing*, 7(3):904–916, 1986.

[8] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[9] G. E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11):1526–1538, 1989.

[10] R. Blikberg and T. Sørevik. Load balancing and OpenMP implementation of nested parallelism. *Parallel Computing*, 31(10-12):984–998, 2005.

[11] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[12] M. H. 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. *Journal of Computational Physics*, 111(2):220–236, 1994.

[13] E. Carson and J. Demmel. Accuracy of the *s*-step Lanczos method for the symmetric eigenproblem. Technical Report UCB/EECS-2014-165, EECS Department, University of California, Berkeley, Sept. 2014.

[14] E. Carson and J. Demmel. Error analysis of the *s*-step Lanczos method in finite precision. Technical Report UCB/EECS-2014-55, EECS Department, University of California, Berkeley, May 2014.

[15] C. Chuanbo and Z. Haiming. Linear binary tree. In *Proceedings of the 9th International Conference on Pattern Recognition*, pages 576–578. IEEE Computer Society Press, 1988.

[16] P. Colella, D. T. Graves, J. N. Johnson, H. S. Johansen, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo software package for AMR applications: Design document. *Lawrence Berkeley National Laboratory*, 2013.

[17] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172. ACM, 1969.

[18] K. Datta. *Auto-tuning stencil codes for cache-based multicore platforms*. PhD thesis, EECS Department, University of California, Berkeley, Dec. 2009.

[19] K. Datta, S. Kamil, S. Williams, L. Oliker, J. Shalf, and K. Yelick. Optimization and performance modeling of stencil computations on modern microprocessors. *SIAM Review*, 51(1):129–159, 2009.

[20] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 4:1–4:12, 2008.

[21] H. DeCougny, K. Devine, J. Flaherty, R. Loy, C. Özturan, and M. Shephard. Load balancing for the parallel adaptive solution of partial differential equations. *Applied Numerical Mathematics*, 16(1-2):157–182, 1994.

[22] R. Deiterding. Detonation structure simulation with AMROC. In L. Yang, O. Rana, B. Di Martino, and J. Dongarra, editors, *High Performance Computing and Communications*, volume 3726 of *Lecture Notes in Computer Science*, pages 916–927. Springer Berlin Heidelberg, 2005.

[23] K. D. Devine, E. G. Boman, R. T. Heaphy, B. A. Hendrickson, J. D. Teresco, J. Faik, J. E. Flaherty, and L. G. Gervasio. New challenges in dynamic load balancing. *Applied Numerical Mathematics*, 52(2-3):133–152, 2005.

[24] J. Dreher and R. Grauer. Racoon: A parallel mesh-adaptive framework for hyperbolic conservation laws. *Parallel Computing*, 31(8-9):913–932, 2005.

[25] H. Dursun, K.-I. Nomura, L. Peng, R. Seymour, W. Wang, R. K. Kalia, A. Nakano, and P. Vashishta. A multilevel parallelization framework for high-order stencil computations. In H. Sips, D. Epema, and H.-X. Lin, editors, *Euro-Par 2009 Parallel Processing*, volume 5704 of *Lecture Notes in Computer Science*, pages 642–653. Springer Berlin Heidelberg, 2009.

[26] L. Ferm, A. Hellander, and P. Lötstedt. An adaptive algorithm for simulation of stochastic reaction-diffusion processes. *Journal of Computational Physics*, 229(2):343–360, 2010.

[27] B. Fornberg. Calculation of weights in finite difference formulas. *SIAM Review*, 40(3):685–691, 1998.

[28] J. H. Freidman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[29] G. Gan, C. Ma, and J. Wu. *Data clustering: Theory, algorithms and applications*. Society for Industrial and Applied Mathematics, 2007.

[30] I. Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[31] B. Gustafsson. *High order difference methods for time dependent PDE*, volume 38 of *Springer Series in Computational Mathematics*. Springer, 2008.

[32] V. Hernandez, J. Roman, and A. Tomas. Evaluation of several variants of explicitly restarted Lanczos eigensolvers and their parallel implementations. In M. Daydé, J. Palma, A. Coutinho, E. Pacitti, and J. Lopes, editors, *High Performance Computing for Computational Science - VECPAR 2006*, volume 4395 of *LNCS*, pages 403–416. Springer Berlin Heidelberg, 2007.

[33] M. Hochbruck and C. Lubich. On Magnus integrators for time-dependent Schrödinger equations. *SIAM Journal on Numerical Analysis*, 41(3):945–963, 2004.

[34] M. F. Hoemmen. *Communication-Avoiding Krylov subspace methods*. PhD thesis, EECS Department, University of California, Berkeley, Apr. 2010.

[35] M. Holmström. Hybrid modeling of plasmas. In G. Kreiss, P. Lötstedt, A. Målqvist, and M. Neytcheva, editors, *Numerical Mathematics and Advanced Applications 2009*, pages 451–458. Springer Berlin Heidelberg, 2010.

[36] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

[37] S. K. Kim and A. T. Chronopoulos. A class of Lanczos-like algorithms implemented on parallel computers. *Parallel Computing*, 17(6-7):763–778, 1991.

[38] K. Kormann. *Efficient and reliable simulation of quantum molecular dynamics*. PhD thesis, Department of Information Technology, Uppsala University, 2012.

[39] K. Kormann, S. Holmgren, and H. O. Karlsson. Global error control of the time-propagation for the Schrödinger equation with a time-dependent Hamiltonian. *Journal of Computational Science*, 2:178–187, 2011.

[40] H.-O. Kreiss and G. Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. In *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 95–211. Academic Press, New York, 1974.

[41] M. Kronbichler and K. Kormann. A generic interface for parallel cell-based finite element operator application. *Computers & Fluids*, 63:135–147, 2012.

[42] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[43] G. Linde, J. Persson, and L. von Sydow. A highly accurate adaptive finite difference solver for the Black–Scholes equation. *International Journal of Computer Mathematics*, 86:2104–2121, 2009.

[44] W. Liu and A. Sherman. Comparative analysis of the Cuthill–McKee and the Reverse Cuthill–McKee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

[45] P. Lötstedt and S. Söderberg. Parallel solution of hyperbolic PDEs with space-time adaptivity. In *Finite Volumes for Complex Applications II : Problems and Perspectives*, pages 769–776. Hermes Science Publications, 1999.

[46] P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126(3):330–354, 2000.

[47] W. Magnus. On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, 7(4):649–673, 1954.

[48] K. Mattsson and M. Carpenter. Stable and accurate interpolation operators for high-order multiblock finite difference methods. *SIAM Journal on Scientific Computing*, 32(4):2298–2320, 2010.

[49] K. Mattsson and J. Nordström. Summation by parts operators for finite difference approximations of second derivatives. *Journal of Computational Physics*, 199(2):503–540, 2004.

[50] D. J. Mavriplis. Three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes. *AIAA Journal*, 33(3):445–453, 1995.

[51] J. Melenk, K. Gerdes, and C. Schwab. Fully discrete hp-finite elements: fast quadrature. *Computer Methods in Applied Mechanics and Engineering*, 190(32-33):4339–4364, 2001.

[52] C. Moler and C. van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[53] A. Nissen, G. Kreiss, and M. Gerritsen. Stability at nonconforming grid interfaces for a high order discretization of the Schrödinger equation. Technical Report ISSN 1404–3203/2011-017, Department of Information Technology, Uppsala University, 2011.

[54] S. A. Orszag. Spectral methods for problems in complex geometries. *Journal of Computational Physics*, 37(1):70–92, 1980.

[55] B. N. Parlett and D. S. Scott. The Lanczos algorithm with selective orthogonalization. *Mathematics of Computation*, 33(145):217–238, 1979.

[56] J. R. Pilkington and S. B. Baden. Dynamic partitioning of non-uniform structured workloads with spacefilling curves. *IEEE Transactions on Parallel and Distributed Systems*, 7(3):288–300, 1996.

[57] K. G. Powell, P. L. Roe, T. J. Linde, T. I. Gombosi, and D. L. De Zeeuw. A solution-adaptive upwind scheme for ideal magnetohydrodynamics. *Journal of Computational Physics*, 154(2):284–309, 1999.

[58] R. Rabenseifner, G. Hager, and G. Jost. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 427–436, 2009.

[59] J. Rantakokko and M. Thuné. Parallel structured adaptive mesh refinement. In R. Trobec, M. Vajteršic, and P. Zinterhof, editors, *Parallel Computing*, pages 147–173. Springer London, 2009.

[60] G. Rivera and C.-W. Tseng. A comparison of compiler tiling algorithms. In S. Jähnichen, editor, *Compiler Construction*, volume 1575 of *Lecture Notes in Computer Science*, pages 168–183. Springer Berlin Heidelberg, 1999.

[61] G. Rivera and C.-W. Tseng. Tiling optimizations for 3D scientific computations. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, SC '00, pages 32:1–32:23, 2000.

[62] R. B. Sidje. Expokit: A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software*, 24(1):130–156, 1998.

[63] H. D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization rethods. *Linear Algebra and its Applications*, 61:101–131, 1984.

[64] B. Strand. Summation by parts for finite difference approximations for d/dx. *Journal of Computational Physics*, 110(1):47–67, 1994.

[65] H. Sundar, R. S. Sampath, and G. Biros. Bottom-up construction and 2:1 balance refinement of linear octrees in parallel. *SIAM Journal on Scientific Computing*, 30(5):2675–2708, 2008.

[66] J. D. Teresco, K. D. Devine, and J. E. Flaherty. Partitioning and dynamic load balancing for the numerical solution of partial differential equations. In A. M. Bruaset and A. Tveito, editors, *Numerical Solution of Partial Differential Equations on Parallel Computers*, volume 51 of *Lecture Notes in Computational Science and Engineering*, pages 55–88. Springer Berlin Heidelberg, 2006.

[67] T. Weinzierl and M. Mehl. Peano: A traversal and storage scheme for octree-like adaptive cartesian multiscale grids. *SIAM Journal on Scientific Computing*, 33(5):2732–2760, 2011.

[68] K. Weiss and L. De Floriani. Bisection-based triangulations of nested hypercubic meshes. In S. Shontz, editor, *Proceedings of the 19th International Meshing Roundtable*, pages 315–333. Springer Berlin Heidelberg, 2010.

[69] A. M. Wissink, R. D. Hornung, S. R. Kohn, S. S. Smith, and N. Elliott. Large scale parallel structured AMR calculations using the SAMRAI framework. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, SC '01, pages 22:1–22:11, 2001.

[70] A. Zewail. Laser femtochemistry. *Science*, 242:1645–1653, 1988.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 1199

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-234984

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2014