
New Approaches for Efficient on-the-fly FE Operator Assembly in a High-performance Mantle Convection Framework

Simon Bauer



München 2018

New Approaches for Efficient on-the-fly FE Operator Assembly in a High-performance Mantle Convection Framework

Simon Bauer

Dissertation
an der Fakultät für Geowissenschaften
der Ludwig-Maximilians-Universität
München

vorgelegt von
Simon Bauer
aus Landshut (Deutschland)

München, den 11 September 2018

Erstgutachter: Prof. Dr. Hans-Peter Bunge

Zweitgutachter: Prof. Dr. Ulrich Rude

Tag der mündlichen Prüfung: 18.02.2019

Contents

Summary	xiii
1 Introduction	1
1.1 Geophysical model	2
1.2 State of the art convection frameworks	4
1.2.1 Model complexities and numerical challenges	4
1.2.2 State of the art numerical solvers	6
1.3 Challenges in high-performance computing for exascale	7
1.3.1 High-performance computing systems	7
1.3.2 Node-level performance	9
1.4 Matrix-free finite element concepts	12
1.5 TerraNeo - A prototype mantle convection framework	15
1.6 Objective of the thesis	18
1.7 References	21
2 Stencil scaling approach for variable coefficient operators	29
2.1 Motivation of scaling approach	29
2.2 Problem setting and definition of the scaling approach	31
2.2.1 Definition of our scaling approach	31
2.2.2 Stencil structure	33
2.3 Variational crime framework and a priori analysis	34
2.3.1 Abstract framework for \mathbf{L}^2 -norm estimates	34
2.3.2 Verification of the assumptions	35
2.4 Guaranteed uniform coercivity	36
2.4.1 Pre-asymptotic modification in 2D based on (A2)	37
2.4.2 Numerical counter example in the pre-asymptotic regime	38
2.4.3 The sign of the stencil entries in 3D	39
2.5 Reproduction property and primitive concept	40
2.6 Cost of a work unit	42
2.6.1 Cost for stencil scaling	42
2.6.2 Cost of the on-the-fly computation for the classical FEM	43
2.6.3 Cost of stencil application	44
2.6.4 Memory accesses	45
2.7 Numerical accuracy study and run-time comparison	46
2.7.1 A quantitative comparison in 2D for a scalar permeability	47
2.7.2 A quantitative comparison in 3D for a scalar permeability	47

2.7.3	A quantitative comparison in 3D for a permeability tensor	48
2.7.4	Application to a blending setting and large scale results	49
2.8	References	53
3	Two-scale approach for efficient on-the-fly operator assembly	55
3.1	Introduction	57
3.2	Model problem and hybrid hierarchical meshes	57
3.2.1	Model problem	57
3.2.2	Domain and finite element approximations	58
3.2.3	Hybrid hierarchical mesh stencils	59
3.3	Two-scale approach	60
3.3.1	Interpolation of stencils (IPOLY)	61
3.3.2	Least-squares approximation (LSQP)	61
3.3.3	IPOLY versus LSQP for quadratic surrogate polynomials	63
3.3.4	Definition of the surrogate operator	63
3.3.5	Some remarks on a priori estimates	66
3.4	Evaluation of accuracy and cost	66
3.4.1	Accuracy of IPOLY and LSQP	67
3.4.2	Cost considerations	69
3.5	Kernel optimization	71
3.5.1	Single core performance	72
3.5.2	Socket performance	74
3.6	Weak and strong scaling	75
3.7	Alternative curved geometries	76
3.8	Conclusion and outlook	77
3.9	Acknowledgements	78
3.10	Appendix A. Blending functions	78
3.10.1	Appendix A.1. Spherical Shell	78
3.10.2	Appendix A.2. Half cylinder mantle	78
3.10.3	Appendix A.3. Curved pipe	78
3.11	References	79
4	Matrix-free approach for large-scale geodynamic simulations	81
4.1	Introduction	83
4.2	Software framework and discretization	85
4.3	Efficient on-the-fly stencil assembly	86
4.4	Towards a rigorous performance analysis	89
4.5	Accuracy and weak scaling results	90
4.6	Outlook	95
4.7	Acknowledgements	95
4.8	References	95
5	Dynamic topography	97
5.1	Benchmark Setup	98
5.2	Benchmark Implementation and Results	99
5.3	Scalability	100
5.4	Dynamic topography with lateral viscosity variations	102

CONTENTS

vii

5.5 Appendix: Supplementary Figures	106
5.6 References	109
6 Outlook	111
Acknowledgments	113

List of Figures

1.1	Different meshes for thick spherical shell	4
1.2	Cache-aware roofline model	11
1.3	Haswell topology	12
1.4	Three types of sub-tetrahedra and 15-point stencil	17
2.1	Stencil with local indices and six sub-tetrahedra	33
2.2	Elements in $S_h(T)$ and S_h	35
2.3	Application of (MA2)	40
2.4	Uniform refinement of one macro element T	40
2.5	Stencil entries colored by their sign	41
2.6	Neighborhood of node i for sub-expression elimination	44
2.7	Macro mesh of the reference domain Ω	50
3.1	Approximation of spherical shell	58
3.2	Sequence of mappings	59
3.3	15-point stencil and three classes of sub-tetrahedra	59
3.4	Layout of 15-point stencil and geometric primitives	60
3.5	Reference tetrahedron with sampling points	61
3.6	Uniform and projected stencil values	62
3.7	Difference between IFEM and IPOLY or LSQP	64
3.8	$e_{\mathcal{W}}$ for IPOLY and LSQP	65
3.9	Exact solution and discretization error	67
3.10	Development of error and residual	68
3.11	Discretization error for a sequence of refinement levels	68
3.12	ECM model	74
3.13	Performance compared with ECM model prediction	75
3.14	Weak and strong scaling results	75
3.15	Half cylinder mantle and curved pipe	76
4.1	Memory usage and computation vs. communication	92
5.1	Benchmark results	100
5.2	Relative errors with respect to semi-analytical solution	101
5.3	Radial viscosity profiles for Cases A, B, and C	103
5.4	Surface dynamic topography	104
5.5	CMB dynamic topography	106
5.6	Lithospheric thicknesses	107

List of Tables

2.1	Minimal and maximal eigenvalues of the global stiffness matrix A	39
2.2	Reproduction property	42
2.3	Operation count for stencil scaling or on-the-fly assembly	44
2.4	Total bytes loaded for computation of stencil	46
2.5	Results for a regular triangular mesh	48
2.6	3D results in the case of a scalar coefficient function	49
2.7	3D results in the case of a tensorial permeability	49
2.8	Results for large scale 3D application	50
3.1	Discretization error for different combinations of h_ℓ and H	69
3.2	Setup times	70
3.3	Cycle numbers reported by IACA	73
3.4	Discretization error for half cylinder mantle	77
3.5	Discretization error for curved pipe	77
4.1	Performance on Intel SNB core	90
4.2	Velocities for IFEM vs. LSQP	92
4.3	MPI_DAPL settings	93
4.4	Configurations used in experiments	93
4.5	Weak scaling results	94
5.1	Weak scaling results on Hazel Hen	102

Summary

The ground under our feet appears to be rock-solid. But on geophysical time-scales, ranging over millions of years, the material of the Earth's mantle acts like a highly viscous fluid that is driven by temperature contrasts. Due to the time-scales, this convection process can only be captured through computer simulations. Starting from simplified models decades ago, to ever more sophisticated models, these simulations provide a lot of insight in the behavior of the Earth. But still, there is an ongoing demand for more complex models with higher resolutions. In view of the up-coming exascale supercomputer generation there seems to be an almost infinite amount of compute power available that facilitates these simulations. However, the utilization of such tremendous compute capacities requires the re-thinking and re-design of existing algorithms and software.

For instance, traditional finite element implementations that store the system matrix have limitations due to their high memory traffic and storage requirements. This is especially true for large scale geophysical simulations where Earth's length scales dictate resolutions that yield systems of up to ten trillion $\mathcal{O}(10^{13})$ unknowns. This can easily exceed the storage capacities of even modern supercomputers. Furthermore, on modern high-performance architectures memory throughput and latency is more critical than the number of floating-point operations. Therefore matrix-free concepts provide an attractive alternative. These approaches perform matrix-vector multiplication locally where the contributions of the matrix are computed on-the-fly and directly multiplied with the local vector. Compared to a stored-matrix implementation this reduces the memory footprint drastically. Such matrix-free techniques are especially well suited for block-structured uniform grids. In this thesis we will build on the hierarchical hybrid grids framework and develop novel methods for an efficient on-the-fly operator assembly. In particular they are designed to tackle two challenges that arise in geophysical simulations: 1) variable viscosity and 2) curved domains that cannot be accurately represented by hybrid grids.

Our first approach shows an efficient method to include variable material parameters, like viscosity. This is based on an appropriate scaling of a constant reference operator with the variable coefficients. We will present an extensive mathematical discussion and demonstrate the efficiency for a large scale application.

The second approach deals with curved geometries, like the spherical shell. Here, the exact finite element operator is approximated by low order polynomials. This significantly improves the performance of the hierarchical hybrid grids concept on non-polyhedral domains. Its accuracy is demonstrated through numerical experiments. Moreover, a modification for complex systems of PDEs is shown. This enables us to run complex geophysical simulations at unprecedented global resolutions of the Earth's mantle close to 1 km. We will use this new method to study the effects of lateral viscosity variations on Earth's dynamic topography.

The proposed methods are incorporated in the prototype of our mantle convection frame-

work and allow researchers to run their models at unprecedented global resolution. This will foster a deeper understanding of our planet.

Chapter 1

Introduction

While to us as human beings the ground on which we walk may appear 'rock-solid' the surface of our planet is actually in constant albeit very slow motion. The continental plates move at a rate of centimeters per year. The reason for this movement are enormous forces acting deep below our feet. Convective processes in the Earth's mantle help the planet to rid itself of excess energy that is either left from the time of its formation or generated by continued radioactive decay. The mantle is a layer of Earth starting from below the crust at roughly 60 km and extending down to the core-mantle-boundary at a depth of about 3 000 km. On geologic time-scales the rocks inside the mantle behave like a highly viscous fluid. A single overturn of the material in the mantle takes about 100 mio. years [1].

A detailed understanding of these processes is of fundamental interest to geophysics, as they are the driving force behind phenomena such as mountain building, back-arc volcanism and finally earthquakes. While the basic principles are those of fluid dynamics and, thus, well understood [2, 3, 4], there are essential details and parameters of mantle convection that are only poorly known [5]. Key among these is the (dynamic) viscosity which describes the rheology of the mantle.

The interior of the Earth is not directly accessible to us. What we know about the inner structure of our planet is derived from indirect observations and inverse reconstructions, such as measurements of the Earth's Geoid or Seismic Tomography [6, 7]. Consequently many research questions can only be studied through numerical simulation. Therefore, computational techniques have for a long time played an important role in geophysics [8, 9, 10, 11, 12].

But there is a constant demand for ever more realistic models. For instance non-linear rheology models with abrupt viscosity changes of several orders of magnitude [13]. Such models need extremely high spatial resolution that significantly increases the computational cost. Another current trend is to perform adjoint simulations in order to link uncertain geodynamic modeling parameters to geologic observables, see e.g. [14]. These models need to propagate back and forth in time multiple times, which also increases the computational cost by another order of magnitude.

In order to run such simulations in an affordable amount of time, we need efficient software that relies on the following components: 1) availability of high-performance compute infrastructure; 2) sophisticated numerical algorithms; 3) efficient implementation that scales up to millions (and beyond) compute cores on a highly heterogeneous hardware where processing units are combined with accelerators like graphics processing units, see section 1.3

The former seems to be readily given by current and future supercomputers. Especially

in the era of exascale, i.e. supercomputers that can perform more than 10^{18} floating-point operations per second, there will be a sheer infinite amount of compute power available. The first exascale supercomputer, *Tianhe-3* in China, is planned to start operation in 2020 [15]. However, in order to make use of this power, the latter two points must be synchronized. In other words, the convergence of numerical algorithms is as important as their efficient, hardware-aware implementation and scalability. This requires the re-thinking and re-design of established methods, as well as an implementation tailored to the targeted architecture.

In this thesis we will tackle some of these challenges to advance the development of exa-ready mantle convection software. This is essential to improve our understanding of Earth's mantle processes.

1.1 Geophysical model

The mathematical model of mantle convection is based on the principles of conservation of momentum, mass and energy. Due to the very slow motion of the fluid, the Coriolis force and inertial terms are insignificant and can be neglected. A detailed explanation can be found in [2]. Thus, the governing equations read as

$$\operatorname{div} \boldsymbol{\sigma} + \rho \mathbf{g} = 0 \quad (1.1)$$

$$\partial_t \rho + \operatorname{div}(\rho \mathbf{u}) = 0 \quad (1.2)$$

$$\partial_t(\rho e) + \operatorname{div}(\rho e \mathbf{u}) + \operatorname{div} \mathbf{q} - H - \boldsymbol{\sigma} : \dot{\boldsymbol{\epsilon}} = 0 \quad (1.3)$$

Here \mathbf{u} represents velocity, ρ density, \mathbf{g} gravitational acceleration, e internal energy, H volumetric radiogenic heat production rate and \mathbf{q} heat flux per unit area. The terms $\boldsymbol{\sigma}$ and $\dot{\boldsymbol{\epsilon}}$ are the stress and rate of strain tensor, respectively. These are coupled to each other, to velocity and to pressure p via

$$\boldsymbol{\sigma} = 2\mu \left(\dot{\boldsymbol{\epsilon}} - \frac{\operatorname{tr} \dot{\boldsymbol{\epsilon}}}{3} \mathbf{I} \right) - p \mathbf{I} \quad , \quad \dot{\boldsymbol{\epsilon}} = \frac{1}{2} \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) \quad , \quad (1.4)$$

where \mathbf{I} represents identity and μ dynamic viscosity, see also [16]. We highlight that the viscosity is the great unknown of the mantle properties. It depends on pressure, temperature, and even on the flow velocities yielding a non-linear rheology.

The density can be split into two contributions $\rho = \rho_{\text{ref}} + \hat{\rho}$ where ρ_{ref} is the radial *background density* and $\hat{\rho}$ describes three-dimensional deviations from the background density due to either: variations in thermal properties, or chemical composition. The deviations $\hat{\rho}$ are relatively small compared to the background density and, thus, can be ignored in the continuity equation (1.2). Moreover, the background density is derived from the time-constant hydrostatic pressure such that its time derivative will be zero. For $\partial_t \rho = 0$, the propagation of acoustic waves is not possible. This motivates the name *anelastic approximation* [17], which is given as

$$\operatorname{div}(\rho_{\text{ref}} \mathbf{u}) = 0 \quad .$$

Like density, also the pressure can be split into two contributions $p = p_s + \hat{p}$ with hydrostatic pressure p_s and dynamic pressure \hat{p} . In the momentum equation the background density only influences the hydrostatic pressure via

$$\nabla p_s = \rho_{\text{ref}} \mathbf{g} \quad .$$

Thus, convection is solely driven by the density variations $\hat{\rho}$ and it cannot be neglected in (1.1).

However, the presence of ρ_{ref} adds an additional complexity for the numerical solution of the system (1.1), (1.2). In particular, it prevents symmetry as the gradient of the pressure from (1.1) is not adjoint to the divergence term $\text{div}(\rho_{\text{ref}}\mathbf{u})$ of (1.2). Depending on how it is treated numerically this will significantly impact the performance of the numerical solver. For instance, the right choice of preconditioner for the non-symmetric Stokes system is unclear. A detailed discussion goes beyond the scope of this thesis and we refer to e.g. [12]. Therein, different approaches are discussed and tested.

For this thesis we will use another common approximation that is similar to the anelastic one and additionally assumes a constant ρ_{ref} in the continuity equation. This is also motivated by the fact, that the lateral density deviations are very small compared to the background density, thus locally the fluid is quasi-incompressible. This *Boussinesq approximation* is widely used in mantle convection simulations, e.g. in [11, 18], and yields for the continuity equation

$$\text{div}(\mathbf{u}) = 0 .$$

This also simplifies $\text{div} \boldsymbol{\sigma}$ in (1.1) to

$$\text{div} \boldsymbol{\sigma} = \text{div} (2\mu\dot{\boldsymbol{\epsilon}}) - \nabla p .$$

For both, anelastic and Boussinesq approximation there is no explicit time-dependence in the momentum and continuity equation. This results in the following *generalized Stokes problem*

$$\begin{aligned} -\text{div} (2\mu\dot{\boldsymbol{\epsilon}}) + \nabla p &= \rho\mathbf{g} \\ \text{div}(\mathbf{u}) &= 0 \end{aligned} \tag{1.5}$$

that models the stationary flow-field for a given buoyancy term $\rho\mathbf{g}$.

The time-dependent energy equation (1.3) can be re-cast in terms of temperature T

$$c_p\rho\partial_t T - \alpha T\partial_t p + c_p\rho\mathbf{u} \cdot \nabla T - \alpha T\mathbf{u} \cdot \nabla p - \text{div}(k\nabla T) - H - \boldsymbol{\sigma} : \dot{\boldsymbol{\epsilon}} = 0 \tag{1.6}$$

with c_p, α, k being the specific heat capacity, thermal expansivity and thermal conductivity.

Note, that the explicit time-dependence of the model is contained solely in the advection-dominated energy equation (1.6) and enters the generalized Stokes part via changes in the buoyancy term that is described by an equation of state. The equation of state relates density to pressure and temperature $\rho = \rho(p, T)$. The exact details of this relation vary between models as the chemical composition and mineralogy of the mantle is still an open question.

In a numerical scheme, the Stokes system will be decoupled from the energy equation and both parts will be solved iteratively. While the temporal discretization of the energy equation must be treated with care, see e.g. [12, 19] and the references therein, the computationally most intensive part is the solution of the generalized Stokes problem (1.5) one or more times per time-step. Several solves per time-step may be necessary in case of a non-linear model. Thus, in this thesis we will focus on the efficient solutions of the Stokes part as this is the crucial component for the overall performance. For time-dependent simulations with our software we refer to [16, 20].

A standard discretization of the generalized Stokes problem, e.g. with the finite element (FE) method as in [19, 20], leads to an algebraic system of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}} \\ \underline{p} \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{f}} \\ \underline{g} \end{bmatrix} . \tag{1.7}$$

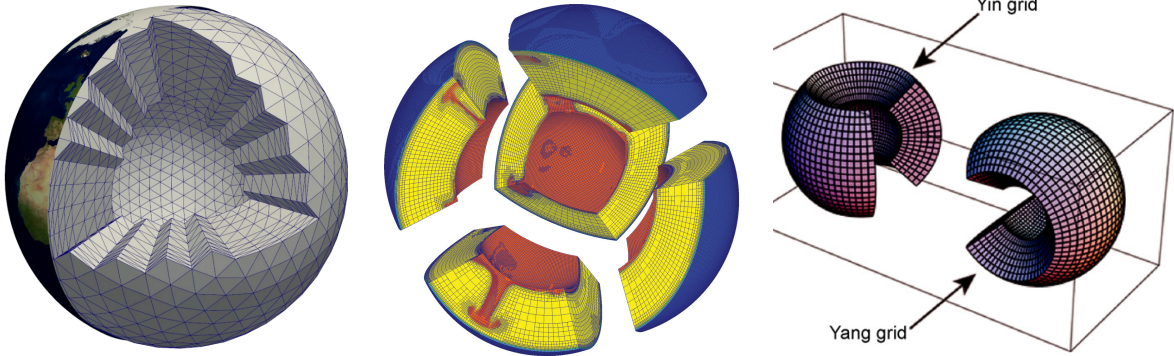


Figure 1.1: Different meshes for thick spherical shell. (Left) Icosahedral mesh from TerraNeo teachlet [28]. (Center) Cubed sphere from [27]. (Right) Yin-Yang grid from [29].

Here, \mathbf{A} represents the discrete viscous operator $-\text{div}(2\mu\hat{\epsilon})$, \mathbf{B}^T a discrete gradient and \mathbf{B} the discrete divergence. Note, that for standard ansatz functions and no-outflow boundary conditions, as prescribed in global convection models, the latter two operators are adjoint to each other. The \mathbf{C} block might represent a stabilization term, that is needed only, if the ansatz functions for velocity and pressure are not inf-sup stable, see [20] for an example. Vectors $\underline{\mathbf{u}}$ and \underline{p} contain the coefficients of the discrete velocity and pressure, while $\underline{\mathbf{f}}$ and \underline{g} represent the discrete right-hand sides, potentially including contributions from the boundary conditions. A detailed analysis of the finite element theory is given in e.g. [21], and, in particular for the Stokes system, in [22, 23].

In geodynamic models the mantle is typically represented as a thick spherical shell. It is commonly discretized either using an icosahedral mesh [24, 25], a Yin-Yang grid [26] or a (modified) cubed sphere [19, 27]. A representative selection of these different meshes is shown in Fig. 1.1. In the TerraNeo software (section 1.5) the icosahedral approach based on tetrahedral elements is used. Both, cubed sphere and Yin-Yang use hexahedral elements instead.

1.2 State of the art convection frameworks

The history of numerical simulations in geodynamics dates back to the 70's [8, 30] starting with simplistic two-dimensional models. The first three-dimensional global convection models were presented in the 80's [9, 31] that were able to produce Earth-like planforms but for simplistic rheologies and at low Rayleigh numbers. Another decade later, parallel computations became feasible and pushed the development of numerical models [3, 10] approaching the dynamical regime of the Earth's mantle. Since then, many more mantle convection codes evolved [11, 26, 32, 33, 34]. The current state of the art mantle convection software is exemplified through [12, 13, 27].

1.2.1 Model complexities and numerical challenges

The complexity of global convection models arises from the large contrast in length scales ranging from the Earth's radius of ~ 6371 km, mantle super-plumes spreading over hundreds of kilometers, tectonic plates with thickness ~ 50 km, shear zones at plate boundaries of a

few kilometers, down to thin plumes of less than 1 km thickness. Further challenges arise through the model complexity that involves compressibility effects, non-linear rheology with lateral variations and discontinuous coefficients, abrupt changes of material properties at phase boundaries and transportation of chemical heterogeneities, see e.g. [12]. In the following we will discuss these challenges in more detail.

The characteristic number in mantle convection simulations is the Rayleigh number Ra that determines the vigor of convection in the system. Essentially it is the ratio of buoyancy force and thermal as well as viscous dissipation, see e.g. [2]. For low Rayleigh numbers the convection is dominated by diffusion, while for high values it is advection-dominated. Furthermore, the Rayleigh number characterizes the thickness of thermal boundary layers that develop at the top and bottom boundary of the spherical shell as well as along rising plumes and down-wellings. The thickness Δ_l of a thermal boundary layer scales as $\Delta_l \sim 1/\sqrt[3]{Ra}$ [4, chapter 8]. In particular, plumes will become thinner and more distinct for high Rayleigh numbers. Within a numerical simulation, a resolution $h < \Delta_l$ that resolves these thermal layers is required. For an assumed value of $Ra = 10^9$ for the Earth's mantle, Δ_l will be around ~ 3 km.

Furthermore, in geophysical models sudden changes in material properties can happen. For instance, steep temperature gradients between cold, subducting slabs and hot mantle material also yield abrupt changes in the temperature-dependent viscosity of several orders of magnitudes [35, 36]. Also at the transition zone between the asthenosphere, which is a mechanically weak layer in the uppermost part of the mantle, and the lower mantle the viscosity changes by several orders of magnitude, see e.g. [37] and the models considered in sections 4.5 and 5.4. Moreover, at phase transitions the mineralogical stability field of the material changes due to small variations in temperature and pressure. This yields abrupt changes of mineralogical composition of the material, and thus, of its viscosity and density [38, 39, 40]. Another aspect of phase transitions are sudden changes of internal energy of the material where latent heat is consumed or released. This must be treated carefully in the energy equation, see [12].

In order to numerically resolve such phenomena we need models with extremely high spatial resolutions in the range of ~ 1 km, or even less. For a global 3D-model this yields systems with $10^{12} - 10^{13}$ degrees of freedom (DoFs). As a consequence, most modern mantle convection codes [12, 13, 27] employ *adaptive mesh refinement* (AMR). The idea is to use a coarse global resolution and refine the mesh locally to increase the resolution in specific regions, e.g. in regions with high temperature gradients, see also Fig. 1.1 (center). While this drastically reduces the number of overall DoFs, it produces significant overhead in terms of mesh handling, dynamic data structures and load balancing. Furthermore, the regions of steep temperature gradients will vary within a dynamic simulation and there is the frequent need of mesh adaptation and re-partitioning of DoFs across compute cores to ensure a good load balance. This becomes especially expensive for massively parallel simulations. Nevertheless, many recent studies successfully employed AMR techniques for large scale global convection simulations [12, 19, 24, 27, 34, 41, 42, 43]. Most prominent, [13] presented a static simulation with $\sim 6 \cdot 10^{11}$ DoFs for a mesh that was refined locally down to 75 m. They demonstrated weak and strong scalability on up to 1.5 million cores, however, without the need for dynamic adaptation of the mesh.

The viscosity is not only subject to sudden changes due to variations in pressure and temperature, but it also depends on the strain rate, and thus the velocity. For instance dislocation creep leads to viscosity variations depending on the second invariant of the strain

rate tensor. Dislocation creep primarily appears in regions of high stresses, like around subducting slabs [2]. This yields a non-linear relation in the Stokes system as $\mu = \mu(\mathbf{u})$. Typically, a non-linear system is solved through an outer iteration of e.g. Newton or Picard type [44] which requires multiple solves of the Stokes system. As the Stokes system is already the dominating factor, the overall cost basically scale with the number of outer iterations. In [13] they use an inexact Newton-Krylov method to solve a static, non-linear system with viscosity contrasts of up to six orders of magnitude.

Another challenge is to include compressibility effects in the mass equation (1.2), i.e. $\text{div}(\rho\mathbf{u})$. In case that ρ depends on the solution variables also the mass equation will be non-linear. This can be circumvented through the anelastic approximation where ρ is replaced by the radial background density ρ_{ref} . But for both cases, the divergence term is not adjoint to the gradient term of the momentum equation (1.1), and thus, the Stokes system is no longer symmetric. In [12] the challenges that arise with compressibility and possible solutions are discussed. In particular, a compressibility formulation, that re-writes the mass equation based on geophysical arguments, is shown.

Furthermore, transportation of certain quantities, like chemical composition along the flow field has to be considered. This can be done either via a particle-in-cell method [45], or through a field-based approach where the quantities of interest are advected along the flow field similar to the temperature, but without diffusion, [12]. In [12] a dynamic 3D simulation with AMR and local resolution of 23 km that includes many of the model complexities, like compressibility, temperature-dependent viscosity and tracking of chemical heterogeneities, is shown on 1 536 CPUs.

While all of these codes and examples [12, 13, 27] demonstrate impressively what is achievable today, it is not yet possible to tackle high spatial resolutions combined with all of the model complexities at once, especially within a dynamic simulation. And it remains debated if the AMR approach is suitable for exascale in case of dynamic mesh adaptation. We pursue another approach based on a global refinement of the Earth’s mantle. In [20] it was already shown that a global resolution of 1 km is feasible for an isoviscous setting. For our new mantle convection framework, we will build on this approach and extend it by some of the aforementioned model complexities.

1.2.2 State of the art numerical solvers

For the numerical solution of the indefinite Stokes system (1.7) most modern convection codes employ multigrid (MG) solvers that are known to yield mesh independent convergence rates [46, 47]. Typically they are used to approximate the application of the inverse of the elliptic operator \mathbf{A} within an outer iteration. Such an outer iteration can be either a SchurCG solver [48], as used in [20], or a Krylov subspace method like MINRES and GMRES [44], that are used in [27] and [19], respectively. In [20] the system with a global resolution of 1 km was solved with a geometric multigrid (GMG) solver and the scalability of the GMG solver was demonstrated on up to $9 \cdot 10^5$ parallel threads. While in [19, 27] an algebraic multigrid (AMG) was used due to its advantage to adjust for adaptively refined meshes. Highly optimized AMG implementations, like BoomerAMG from the hypre library [49] have also proven their scalability up to hundreds of thousands of cores. The disadvantage of AMG compared to GMG is the high setup cost and tricky tuning of the many parameters. Furthermore, AMG methods are not suited for the matrix-free concepts described in section 1.4 as they need the matrix entries to define the operator on the coarser levels. In [13] a novel hybrid spectral-

geometric-algebraic multigrid (HMG) is proposed that scales up to 1.5 million cores.

An alternative is to use multigrid for the full Stokes system. Due to the indefiniteness of the system, specialized smoothers are required. In [50] an all-at-once geometric multigrid solver using Uzawa-type smoothers is presented and its superior performance compared to a SchurCG and preconditioned MINRES solver is shown. Therein, a system with unprecedented number of $\mathcal{O}(10^{13})$ DoFs is solved within a couple of minutes. For this thesis the Uzawa-type MG solver (UMG) will be the default solver for the Stokes system.

1.3 Challenges in high-performance computing for exascale

In this section we will give an overview of current state of the art high-performance computing (HPC) architectures and discuss the challenges for software design that arise on the road to exascale computing. Such an exascale supercomputer will be able to perform more than 10^{18} floating-point operations per second (FLOPs/s).

1.3.1 High-performance computing systems

A typical HPC system is composed of a large amount of compute-nodes that are organized in a cluster and connected through a high-speed network. On these nodes one or multiple processing units are installed. Commonly these are so-called core processing units (CPU). Modern CPUs are equipped with multiple instances of compute units (cores) on which the computations are executed. Such CPUs with multiple cores are called multi-core processors.

Instead of only installing CPUs on one node, there is a trend to use hybrid nodes that combine CPUs with graphics processing unit (GPU) accelerators. Such GPUs have thousands of compute units and provide much higher throughput in terms of FLOPs/s, but compared to CPUs they also have less memory and longer latencies to access this memory. For instance, on the *Summit* supercomputer at ORNL [51] each node is equipped with two IBM Power9 CPUs and six NVIDIA Tesla V100 GPU accelerators. Recently, it crested the number one position in the TOP500 [52] list (June 2018) with an impressive peak performance of 122 PFLOPs/s, i.e. $1.2 \cdot 10^{17}$ operations per second, or about 0.12 Exa-FLOPs/s.

Instead of this hybrid setting, another possibility are many-core CPUs like the SW26010 CPU of the *TaihuLight* supercomputer in Wuxi, China, that follows Summit in the TOP500 list. The SW26010 CPU is a homegrown chip of the Chinese *National Research Center of Parallel Computer Engineering & Technology* with 260 compute cores. A description of the system, recent achievements and a discussion of the challenges that arise with this architecture can be found in, e.g., [53, 54].

Other many-core CPUs like the Intel Xeon Phi Knights Landing (KNL) are also used in HPC systems, like on *Cori* at NERSC [55]. In [56] a comparison of the performance on Intel Haswell and KNL is shown. However, Intel recently announced that there will be no successor of KNL.

Besides the many choices for node setup, also the types of nodes within one HPC system can vary. For instance, on the *Piz Daint* system at CSCS [57] there are compute nodes of type XC50 that are equipped with one CPU (Intel Xeon E5-2690 v3) and one GPU (NVIDIA Tesla P100) and nodes of type XC40 with two CPUs (Intel Xeon E5-2695 v4).

Looking at the current TOP500 list, there is a clear trend to such highly heterogeneous configurations with hybrid nodes. While such configurations provide the highest (theoretic) peak performances, they also pose a great challenge for the design of algorithms and their

implementation. For instance, parts of the code that require a lot of memory accesses should preferably be executed on the CPU, while other parts with large number of arithmetic operations should be performed on the GPU. But a proper selection of the ratio of CPU and GPU computation is a fairly non-trivial task [58]. Furthermore, in order to achieve the best possible performance, it is also essential to optimize the implementation with respect to the specific hardware and exploit its advantages [59]. In section 3.5 we will present a detailed description of such a node-level optimization. However, this requires an in-depth knowledge of the characteristics of the specific architecture and can consume a significant amount of developing time. This has to be done not only for a CPU version, and a GPU version, but also for each different type of processor. If this is done manually, it can become a very tedious and error prone task. So there is also a trend to use automatic code generation approaches that generate optimized code for some given hardware. This is a field of active research and we refer to [60, 61] for some recent developments in the area of climate and weather simulations.

Another important aspect in HPC is the communication that takes place between different cores within the cluster. In a typical parallel computation the workload is distributed over all cores, where each core takes care of a subset of DoFs. Inside each subset computation can be performed independently of all other cores. However, on the interfaces between two different subsets information must be exchanged frequently to keep all values up to date. Thus, the two cores have to communicate with each other. This can be either of intra-node type where information is exchanged only between cores within one node that share some common memory, or of inter-node type where the communication happens between remote nodes and data must be send through the network connections. Another example is the computation of a scalar product where information of all subsets must be gathered to form one final result. Therefore, all cores have to send information to one master-core. Naturally, the cost (time) for such communication becomes higher the more cores are involved, as more data needs to be transferred through the network at the same time. In the worst case, an all-to-all communication takes place where each core communicates with all other cores. This cost scales with the squared number of cores. This has been an issue in HPC since the beginning of parallel computing, and will become ever more important the more cores are used. Nowadays the largest simulations include up to 10 mio. cores [53]. And this will increase by another order of magnitude for exascale. This is a critical factor and communication should be avoided as much as possible. Therefore, we also investigate in section 4.5 the percentage of communication and computation for our application.

Additionally, for the different types of communication also different choices for the parallel programming model exist. For the shared memory intra-node communication commonly either MPI or OpenMP is used, while the distributed-memory inter-node communication is realized via MPI. Also a hybrid approach of MPI and OpenMP is possible and widely used. In case that GPUs are involved, the programming model is further enriched by OpenCL or CUDA. The right choice of the parallel programming model is not trivial and can significantly influence the performance.

Furthermore, with the increase of compute cores, also the probability that one of them fails, increases. This could either be a hard failure where some node crashes that is noticed immediately, or a soft failure, like a *bit-flip*, that happens silently. In such cases so-called fault-tolerant strategies are necessary to recover the current state without re-starting the whole simulation. These may be either hardware-based, software-based or algorithm-based. While traditional approaches like checkpointing are rather expensive, algorithm-based approaches can be very attractive. For a current discussion we refer to [62] and the references therein.

Especially for exascale supercomputers such failures may happen considerably often and then resilient algorithms will be essential.

Also energy consumption is an issue. For instance, the power-consumption of Summit is comparable to that of on average 8 100 US households [63]. Therefore, to save energy, and thus, cost, some of the compute cores may be clocked at lower frequencies. But this adds another level of heterogeneity to the overall system and the node-level optimized code needs to be adapted to this new clock-frequency.

1.3.2 Node-level performance

In the following we will discuss the architecture of modern CPUs, its bottlenecks and the challenges to optimize code on the node-level. A detailed description of CPU architecture and node-level performance engineering is given in [64]. For the examples we will mostly refer to the SuperMUC system at LRZ [65]. Most of the computations for this thesis were executed on that system. A detailed description can be found in section 4.5 (Phase 1) and 3.6 (Phase 2).

Modern processors are highly complex devices: for instance they allow hyper-threading where each physical core behaves like two logical cores. Hyper-threading provides two sets of registers such that on each physical core two independent threads can run simultaneously. Both of the logical cores share the same arithmetic execution unit which could increase the performance by keeping the execution pipeline (see below) more busy than a single thread can do. Note, that hyper-threading differs from oversubscription where more threads are running than the number of (logical) cores. Furthermore, additions and multiplications can be performed simultaneously on one core, called *fused multiply add* (FMA). Additionally, modern CPUs are optimized for vectorization where the same instruction is executed on multiple data (SIMD) to reduce latency. Moreover, in order to perform one floating-point operation (instruction) the processing unit needs a certain number of cycles. To reduce the time for one FLOP most modern CPUs employ the pipelining concept, where the instruction is split into small stages. Each stage takes one cycle and can be executed in parallel. Thus, assuming that one instruction takes five cycles, by pipelining a new instruction can be started after the first stage of the previous instruction is executed. Then, instead of one output every five cycles, a filled pipeline produces an instruction output after each cycle. In case of a perfect pipeline, the theoretic number of arithmetic operations that a core can perform per second (FLOPs/s) is given by its clock-frequency. Typical values of this peak performance are somewhere around 3 GHz. For instance the Intel Haswell processor as used in SuperMUC Phase 2 is regularly clocked at 2.6 GHz and in turbo-mode at 3.6 GHz.

However, the clock-frequency is not the only factor that determines the performance, and in fact, it is also not the dominant one. In order to perform arithmetic operations, first data must be loaded from the memory system into the core's register. The registers are small and can hold only a very limited amount of bytes. The access to the main memory (DRAM) system is terribly slow compared to the number of FLOPs that the CPU can perform. Thus, the CPU is idle for most time as it has to wait until some new data is loaded. Therefore, a hierarchy of memory sub-systems was introduced [66]. Nowadays this hierarchy consists of three levels, called L1, L2 and L3 cache. These caches have much less storage capacities than the main memory, but significantly faster bandwidth. The bandwidth is given in bytes per second and defines how fast data can be transferred. Also the latency, which is an initial waiting time until data can flow is much shorter. The L1 cache provides the highest bandwidth

of the three cache levels and the smallest storage capacities, while the L3 cache is the largest one, but with lowest bandwidth. Concretely one can imagine that the L1 cache is closest to the core, while the DRAM is farthest away. The closer the data is, the faster it can be loaded into the register. In order to keep the administration overhead low, the storage capacities of the cache must be small.

Thus, there are two factors that determine the performance: the clock-frequency of the processor and the memory bandwidth. A popular tool to model the performance of a code is the *roofline* model [67]. It is based on the two potentially restricting factors and defines upper bounds for the maximal achievable performance for a given algorithm. The original model only includes the transfer rates from main memory. This implies that within the caches data is transferred at infinite speed, which yields a model where the bounds are too optimistic. In [68] an extension is proposed to include also cache levels. An illustration of this cache-aware roofline model is shown in Fig. 1.2.

The abscissa shows the arithmetic intensity which is the number of operations that are performed for each byte that is loaded into the register. The arithmetic intensity is given in FLOPs per byte and its value is determined by the implemented algorithm. On the ordinate the performance in terms of FLOPs/s is shown. Depending on the overall number of FLOPs that are required to get the final result, a code with lower performance may yield shorter run-times, see e.g. the discussion in [69]. But in general, for a fixed algorithm, the goal is to achieve the highest possible performance in terms of FLOPs per second.

Whether the performance is limited by the processor's peak performance or by the bandwidth depends on the arithmetic intensity. For the former, the code is said to be compute bound. In this case one could try to modify the algorithm such that the number of operations is reduced. This would be an algorithmic optimization, e.g. as we do in chapter 2, but from an implementational point-of-view running at processor's peak performance is the best one could get.

On the other side, in most situations the intensity will be lower such that the performance is limited by the memory throughput (memory bound). In the model four lines are drawn, each with a fixed slope in byte/s. This slope represents the bandwidth of the respective memory layer. Thus, for a given intensity value, the location of the data will significantly influence the performance. For instance, for an intensity of 0.1, if the data has to be loaded from the main memory the maximal performance will be ~ 0.2 GFLOPs/s. However, if the data is already in the L1 cache, the maximal performance will be ~ 20 GFLOPs/s, thus potentially $100\times$ faster. For simplicity we will not distinguish between the three cache levels in the following, but only between cache and DRAM. Of course, all arguments also hold true for the different cache levels.

Consequently, it is preferable to perform operations on data that is already loaded into the cache. The problem is that its capacities are small, and old data that has not been used for some time will get evicted when new data must be loaded. Thus, the goal is to design code such that as much data as possible can be re-used as long as it is still in the cache. If data is used from the cache it is called a *cache hit*, otherwise a *cache miss*. In order to increase the probability of cache hits, whenever some value is loaded, also the values that are stored next to it in the respective memory layer are loaded. This is called a *cache line* and its size depends on the architecture. Here, the assumption is that values which are stored consecutively in memory will most likely be needed in some successive computation. For the discussion in section 3.5 cache lines will serve as the base unit where one cache line consists of eight double-precision values. Another technique that reduces latency is *cache prefetching*

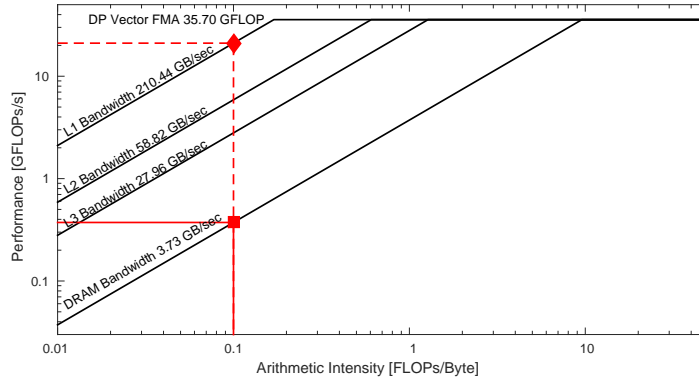


Figure 1.2: Illustration of cache-aware roofline model. Processors peak performance (double precision vectorized FMA) and the four bandwidths (L1, L2, L3 and DRAM) form four different rooftops. These values were measured for a benchmark application on Phase 2 [69]. For a fixed intensity (that is determined by the implemented algorithm) the maximal performance is exemplified for two cases: a) data resides in L1 cache b) data must be loaded from DRAM.

where cache lines are loaded ahead of the actual computation. This is done on the compiler level.

For stencil-based applications, as used in this thesis, an often used term is the so-called layer condition [70]. It is said to be satisfied if all values, except one, that are touched by the stencil are already in the cache. For instance, in 2D for a standard 5-point stencil, exactly two lines of the grid must be kept in memory to fulfill the condition. Then, for a row-based south to north grid traversal all values except for the northern one have already been used in a previous stencil update, but were not yet evicted from the cache. In section 3.5.1 this condition is used for the 3D case where one layer corresponds to one plane.

The modified roofline model is still a simplification. For instance it does not include overlapping of computation and load operations, which requires even more detailed models like the execution-cache memory (ECM) model [71] that is used in section 3.5. Nevertheless, the roofline model gives a good intuition of the observed performance and the potential bottlenecks.

On modern processors, the L1 and L2 cache are exclusively owned by the respective core. That means that for instance core #0 cannot access the L2 cache of core #1, and vice versa. For the L3 cache and DRAM this is different and they are shared among all cores of one processor. Note, that there are also architectures with non-shared L3, see e.g. [64]. The main memory is organized in NUMA (non-uniform memory access) domains [72, 73]. In the NUMA architecture CPUs are directly attached to their local memory to provide fast accesses, and compared to UMA (uniform memory access) architectures, multiple CPUs can simultaneously access the memory. Remote access to memory on other CPUs is also possible, however, then latencies increase by up to 50% [72]. Typically, a NUMA node consists of one CPU and its associated local memory. If the system is configured in so-called *cluster-on-die* (CoD) mode, each NUMA node is further split into two groups. In Fig. 1.3 the topology of one Haswell compute node is shown: it consists of two processors each with 14 cores. The system is running in CoD mode where seven cores form one NUMA domain. The L1 and L2 caches are private to each core, while the L3 and DRAM is shared between seven cores.

In the above paragraph we deliberately used the term processor’s peak performance. This peak value scales linearly with the number of cores and the clock-frequency. When increas-



Figure 1.3: Topology of one Haswell node of SuperMUC Phase 2, taken from [65].

ing the core count on one processor, then shared resources can become a bottleneck. The bandwidth of L3 cache scales also linearly with the number of cores, thus the only critical resource is DRAM bandwidth. When multiple cores simultaneously read from main memory such that the memory interface is continuously occupied the bandwidth will saturate and limit the overall memory throughput. In Fig. 3.13 this effect is illustrated for one NUMA domain on SuperMUC Phase 2. However, we remark that the saturation effect is rather low in this case due to the node-level optimizations performed.

As a consequence, in order to optimize code, all of these aspects, like vectorization, pipelining, layer condition, saturation effects, etc. are important and need to be considered, see e.g [59, 70]. Especially, in most cases memory throughput and memory access latency are more critical than the number of floating-point operations.

1.4 Matrix-free finite element concepts

Finite element (FE) discretizations of a PDE lead to a system of equations of the form

$$Ax = f \tag{1.8}$$

that has to be solved. For the general FE theory we refer to e.g. [21] and a concrete example for the Laplace operator is given in section 3.2.2.

We first introduce the following notation: Let $\{\phi_i, i = 1, \dots, n\}$ be the set of all basis functions of a suitable finite element space, where n is the number of DoFs. Furthermore, i, j are indices for global DoFs, whereas i_{loc}, j_{loc} are the associated local indices w.r.t. element t of the FE triangulation. For local indices the basis functions are denoted as $\phi_{i_{loc}}$. The mapping from the unit reference element \hat{t} , that is used for quadrature, is defined by the Jacobian J_t . We further denote by $\{\hat{\phi}_i, i = 1, \dots, m\}$ the set of basis functions on the reference element. Here, m depends on the dimension and the order of the finite elements.

In classical FE codes, first the $n \times n$ system matrix A will be assembled and stored. The

entries of the system matrix A are defined by the bilinear form $a(\cdot, \cdot)$ that stems from the PDE operator and it is $A_{ij} = a(\phi_i, \phi_j)$.

The result of $a(\phi_i, \phi_j)$ will be non-zero only for combinations of basis function ϕ_i, ϕ_j that have a common support, i.e. that share at least one element. So, most of the entries of A will be zero. In fact, for a scalar operator with linear finite elements on regular tetrahedral meshes there will be exactly 15 non-zero entries per row. For boundary nodes this may vary depending on type of boundary conditions, but on average, only $15n$ out of n^2 matrix entries are non-zero. Such a matrix is called *sparse* and there exist special formats to store only the non-zero values, which significantly reduces the memory requirements. For instance the *Compressed Row Storage* (CRS) format that stores all non-zero entries within an array, and within a second array the associated column indices. Both of these arrays have $15n$ entries. In case that n is large the indices must be stored in 64-bit integer format to avoid integer overflow. Additionally, also pointers that indicate the beginning of a new row must be stored, which adds another n values. Overall, it requires $31n \cdot 8$ bytes to store the matrix A in the CRS format for this particular discretization.

The assembly of the matrix A , which means the computation of its entries, is typically performed through a loop over all elements. On each element t the local element matrix E^t can be assembled via $E_{i_{loc}j_{loc}}^t = a_t(\phi_{i_{loc}}, \phi_{j_{loc}})$. Here, $a_t(\cdot, \cdot)$ is the same bilinear as $a(\cdot, \cdot)$ where the integration is restricted to element t . Concretely, for the Laplacian this reads as

$$E_{i_{loc}j_{loc}}^t = \int_t \nabla \phi_{i_{loc}} \cdot \nabla \phi_{j_{loc}}$$

The integral is computed numerically via a suitable quadrature rule on the reference element, see (1.10). We highlight that this integration is the most expensive part of the assembly procedure. But, as the matrix is stored in memory it has to be done only once. In a similar way, also the right-hand side f of the discrete system can be assembled.

After the assembly procedure, the system (1.8) can be solved, typically with an iterative solver like multigrid or a Krylov subspace method. In every iterative solver (sparse) matrix-vector products (MVP) must be frequently applied which constitutes the most compute intense part within an iterative solver. The disadvantage of this approach is that for each MVP all matrix entries must be read from memory. If n is small, than the values may be kept in the cache, but for larger n these values come from main memory. And as each entry is needed exactly once per MVP, there is also no re-use of the data. In fact, the arithmetic intensity of the stored matrix MVP is very low and, as we discussed in section 1.3, memory throughput and memory access are the limiting factor, such that these codes are strictly memory bound. This is verified in e.g. [69].

Furthermore, for a standard CRS format also indirect addressing for the position of the non-zero entries must be performed, which can further slow down the performance. Finally, the storage requirements also limit the size of problems that can be simulated. For instance, the system considered in section 4.5, with $1.1 \cdot 10^{12}$ DoFs needed more than 500 TB to store all non-zero matrix entries, and another 500 TB to store the 64-bit integer indices. This exceeds by large the memory capacities of even modern supercomputer systems. For comparison, the available memory of the SuperMUC Phase 2 system is only about 194 TB. Note, that for this example there are four DoFs associated with each mesh node, and thus, there are on average $15 \cdot 4$ non-zero entries per row.

To circumvent these issues, a current trend in modern high-performance codes is to use *matrix-free* techniques. These approaches do not assemble the full system matrix, but rather

perform the MVP locally on-the-fly.

Therefore, we recapitulate, that a MVP with the system matrix A involves two steps: first a summation of the local contributions E^t over all elements to assemble A , and then a multiplication with the vector. According to the law of distribution, these two steps can also be switched. On each element the local element matrix E^t can be directly multiplied with the corresponding vector entries and its result added to the final vector. In particular, the MVP $v = Au$ can be expressed as

$$v_i = \sum_{t \in \mathcal{N}(i)} \sum_{j_{\text{loc}}=1}^m E_{i_{\text{loc}}j_{\text{loc}}}^t u_{j_{\text{loc}}} \quad (1.9)$$

where $\mathcal{N}(i)$ is the set of elements including DoF i . One of the first paper that describes this element-by-element (EBE) approach is [74]. The local element matrices E^t could either be stored, which also produces a large memory footprint, or they can be computed on-the-fly. Due to the high memory requirements or the expensive on-the-fly computation this approach is not widely used on unstructured meshes. Nevertheless, on structured meshes the same E^t can be used for all elements of the same shape, thus have to be computed and stored only for a small subset of elements. We note that this is quite similar to the approach that we will use within this thesis. A variable material parameter could either be included directly in E^t , or by an appropriate scaling of the contributions of the local element matrices as developed in chapter 2. The EBE approach has been successfully applied on hexahedral meshes with tri-linear finite elements, see e.g. [75].

Another matrix-free approach is to fuse the local MVP directly with the numerical quadrature of the weak form, instead of assembling the local element matrices [76]. We will discuss this approach for the prototype Laplace operator with material parameter $k \equiv 1$. To this end we recapitulate that the computation of E^t involves an integral that is computed through a numerical quadrature formula which is defined by its quadrature weights w_q and quadrature points x_q . This is performed on the reference element \hat{t} and we denote by \hat{x}_q the corresponding quadrature point on the reference element. Thus,

$$E_{i_{\text{loc}}j_{\text{loc}}}^t = \sum_q \left(J_t^{-T}(\hat{x}_q) \nabla \hat{\phi}_{i_{\text{loc}}}(\hat{x}_q) \cdot J_t^{-T}(\hat{x}_q) \nabla \hat{\phi}_{j_{\text{loc}}}(\hat{x}_q) \right) w_q |\det(J_t)| \quad (1.10)$$

We note that this formula is similar to (4) from section 4.3. There, linear finite elements are considered for which the quadrature points are identical to the four nodes of one element with uniform quadrature weights.

Then, the two steps (1.9) and (1.10) can be merged into one form

$$v_i = \sum_{t \in \mathcal{N}(i)} \sum_q \left(J_t^{-T}(\hat{x}_q) \nabla \hat{\phi}_{i_{\text{loc}}}(\hat{x}_q) \cdot \nabla u^h(x_q) \right) w_q |\det(J_t)| \quad (1.11)$$

with $\nabla u^h(x_q) = J_t^{-T}(\hat{x}_q) \sum_{j_{\text{loc}}=1}^m \nabla \hat{\phi}_{j_{\text{loc}}}(\hat{x}_q) u_{j_{\text{loc}}}$ and $u^h(x_q) = \sum_{j_{\text{loc}}=1}^m \hat{\phi}_{j_{\text{loc}}}(\hat{x}_q) u_{j_{\text{loc}}}$ the local FE interpolation at point x_q . The gradients of the basis functions on the reference element and the inverse of the Jacobian for all quadrature points can be pre-computed and stored at low memory cost such that (1.11) can be evaluated on-the-fly very efficiently. Compared to a classical CRS implementation this method tends to be compute bound [77] and for higher order elements it is significantly faster than the stored matrix approach [76, 78]. In principle it works for arbitrarily shaped elements, but for tensor-product spaces the efficiency of the

sum evaluation over q can be improved through sum factorization. This sum factorization is more efficient the higher the order, but it does not pay off for linear elements [76]. In fact, in this case the CRS method is even faster. For higher order elements (≥ 2), this matrix-free approach has been widely used, see e.g. [76, 77, 78, 79, 80].

In this thesis, we will build on a different matrix-free approach for low-order finite elements on hybrid hierarchical grids with tetrahedral elements. This provides more flexibility than uniform hexahedral elements that are necessary for an efficient EBE approach. In contrast to the element-based approaches discussed above, we employ a node-based approach that uses so-called stencils similar to finite differences. We will explain this approach in more detail in the next section 1.5, where we also discuss its advantages and, in particular, its disadvantages. Within this thesis, we will develop two novel node-based matrix-free approaches to overcome these disadvantages.

1.5 TerraNeo - A prototype mantle convection framework

This thesis is part of the *TerraNeo* project, funded by the DFG Priority Programme on Exascale Computing (SPPEXA) [81]. The ambitious goal of TerraNeo is to provide a new high-performance community code for mantle convection simulations. Compared to other state of the art convection codes that are based on pre-existing libraries, TerraNeo is a complete re-design and re-implementation of its predecessor TERRA [9, 10, 36]. Notable examples for such libraries are the *DEAL.II* library [82, 83] for finite element discretization, the *Trilinos* package [84] for scalable linear algebra operations, as well as *p4est* [42] for parallel adaptive mesh refinement and management.

TerraNeo is carefully implemented and optimized for high-performance computing (HPC) in order to exploit the tremendous compute capacities of the soon-to-come exascale systems. Furthermore, it uses tetrahedral elements which provide more flexibility in the geometries that can be used than hexahedral elements.

While TerraNeo is still in its development phase, its concepts are based on the *Hierarchical Hybrid Grids* (HHG) software framework [16, 20, 85, 86] for low order finite elements. A flexible extension of the HHG concept for higher-order elements is given by [87]. The HHG framework serves as the prototype for our mantle convection software. During the course of the project, many new ideas and features were added to HHG such that it evolved to a full mantle convection framework for geophysical research. For instance, [37] employs HHG to investigate the effect of asthenosphere thickness on flow velocities. And in chapter 5 we will show the influence of lateral viscosity variations on dynamic topography. Thus, it can be seen as the zero-th version of TerraNeo. All concepts and results within this thesis were obtained with HHG. Here, we will give a brief overview of HHG and discuss its advantages and disadvantages. A more detailed description is given in sections 3.2.3, 4.2 and the references therein. The basic idea of HHG is that it combines the flexibility of an unstructured mesh with the superior performance of structured refinement. More precisely, the domain Ω is discretized by an initial tetrahedral macro triangulation \mathcal{T}_0 . These macro elements are grouped into geometric primitive classes, namely *vertices*, *edges*, *faces* and *volumes*. Each primitive class will be further refined following the rules of [88]. This generates a structured hierarchy of uniformly refined meshes $\mathcal{T} = \{\mathcal{T}_\ell : \ell = 0, 1, 2, \dots, L\}$. Note, on \mathcal{T}_0 there exist only DoFs on vertex primitives. On \mathcal{T}_1 DoFs on edges are generated, and only from \mathcal{T}_2 there are DoFs associated with face and volume primitives.

The primitives also serve as containers with array-like data structures to store the associated nodal degrees of freedom. This contiguous memory layout provides extremely efficient memory accesses without the need for indirect addressing. The containers can be distributed via MPI, where each primitive is uniquely owned by one MPI rank. Updates between two adjacent primitives on different ranks are performed via ghost layer exchanges.

Due to the embedding of multiple mesh levels, the HHG concept is naturally suited for geometric multigrid solvers [20, 50]. The most compute intense components of a MG solver are the residual computation, smoothing and the coarse grid solver. For the latter one could either choose a suitable Krylov subspace solver like preconditioned *MINRES*, or, due to the unstructuredness of the macro mesh an algebraic multigrid. In case that the coarse grid is not too large, also a sparse direct solver may be considered. For this an extension was implemented in HHG to link with the external library *PETSc* [89] that provides a easy way to switch between different numerical solvers. For instance the *BoomerAMG* from the hypre library [49], or the direct solver *MUMPS* [90]. All of these choices have their own advantages, but also disadvantages, and it is still a field of active research which solver should be taken [91]. Nevertheless, also the HHG internal Krylov subspace solver provides reasonably good performance, as shown for instance by the results in section 4.5. Thus we will not further dwell on the discussion about the coarse grid solver and employ only HHG internal solvers.

The residual application corresponds to a sparse matrix-vector multiplication plus one additional subtraction, $r = f - Au$. In HHG a node-based matrix-free concept is employed for linear finite elements. Note that for linear finite elements the DoFs are directly located at the nodes. The MVP is performed as follows: first, for each node the contributions of the attached local elements are combined on-the-fly to assemble so-called *stencils*. Let s_{ij} be the stencil weight that connects nodes i and j . It can be computed by

$$s_{ij} = \sum_{t \in \mathcal{N}(i,j)} E_{i_{loc}j_{loc}}^t . \quad (1.12)$$

Here, $\mathcal{N}(i, j)$ is the set of elements with common nodes i and j . For our discretization we have in general 15 stencil weights per node. Fig. 1.4 illustrates the stencil pattern and all elements involved for its assembly. These stencils represent matrix entries of the corresponding matrix row. In the next step, the stencil weights are multiplied with the associated vector entries. Afterwards, all products are accumulated and subtracted from the right-hand side f . Then, the residual application can be expressed as

$$r_i = f_i - \sum_{j=1}^{15} s_{ij} u_i .$$

The big advantage of this approach is, that on block-wise uniform meshes the stencils do not have to be re-computed and can be re-used for the vast majority of points. In particular, the structured refinement of the primitives generates only six different classes of sub-tetrahedra for each primitive. Furthermore, the types of sub-tetrahedra adjacent to one node will be identical for all points within one macro element. This yields the same stencil values for all points. Consequently, (1.12) has to be evaluated only once per macro element, and the stencil weights s_{ij} can be stored and re-used. For example, for a volume primitive on level $l = 8$ there are $2.7 \cdot 10^6$ nodes. But instead of computing $2.7 \cdot 10^6$ stencils on-the-fly, only one stencil has to be computed and can be used for all of the $2.7 \cdot 10^6$ nodes. This significantly

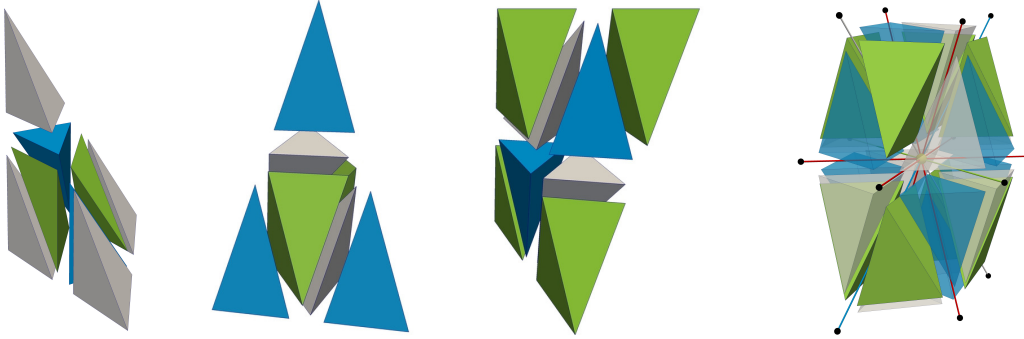


Figure 1.4: Three types of sub-tetrahedra (the three pictures on the left) and 15-point stencil (right), see also Fig. 3.3. Each type includes two congruency classes of sub-tetrahedra.

reduces the number of operations, and also the amount of bytes that have to be loaded. In fact, the stencils, and thus in principle the full matrix can be kept in the L1 cache.

Similarly to the residual, also the smoother can be expressed in stencil form. In particular, one sweep of the Uzawa-smoother is carried out by a combination of Gauss-Seidel (GS) sweeps of the operators \mathbf{A} , \mathbf{C} and applications of \mathbf{B} and \mathbf{B}^T , see [50] for details. The main difference between the residual and smoother application is that the GS smoother requires the inverse of the central stencil value that can be easily computed in a node-based approach. Note, for element-based implementations this is not straight-forward as the central stencil entry includes information of all neighboring elements. In particular, it is the negative sum of all other stencil entries. An alternative is to use polynomial smoothers like the Chebychev smoother. A current discussion of different multigrid smoother for massively parallel applications is given in [92]. We further remark that due to the HHG communication pattern between primitives some dependencies are neglected such that low-order primitives can have points that are updated in a Jacobi-like fashion yielding a hybrid GS update scheme. As shown in [85] this does not effect the multigrid convergence.

The fast performance of HHG was already mentioned several times, for instance it solved a system $\mathcal{O}(10^{13})$ with DoFs [50]. In [86] a detailed hardware-aware performance analysis is presented that quantifies the performance of HHG in a qualitative manner, and not just by giving some speed-up values. In particular, the performance is measured in the so called *parTME* metric that takes into account algorithmic scalability, node-level performance and parallel efficiency. Those three are exactly the components, as we already highlighted in the introduction part of chapter 1, that are essential for an exa-ready software.

However there are two caveats that we did not mention so far. First, the efficiency of the matrix-free concept is based on the assumption that FE stencils do not change within one macro element. But this does not hold true in case of PDEs with variable coefficients, like the viscosity μ that enters the elliptic operator \mathbf{A} . In that case, (1.12) must include the variable coefficient μ and then reads as

$$s_{ij} = \sum_{t \in \mathcal{N}(i,j)} E_{i_{loc}j_{loc}}^t \bar{\mu}_t \quad (1.13)$$

where $\bar{\mu}_t$ is the element averaged viscosity. Note, that for linear FE we can use a vertex based quadrature rule with uniform weights to compute the integral over μ on each element, see also (4) from section 4.3. Then, for a viscosity μ that is not constant within a macro element,

also the stencils will vary within this macro element. Consequently the stencils have to be re-assembled for each node on-the-fly. The contributions of the local element matrices E^t only depend on the shape of the local sub-tetrahedra which is independent of μ . Thus, there are still only six different classes of sub-tetrahedra and for those the contributions E^t can be pre-computed and stored. In fact, they require only 768 byte¹ and fit easily into the L1 cache. Therefore, the expensive on-the-fly computation of E^t can be avoided. But still, compared to the case of block-wise constant μ , the additional evaluation of the sum in (1.13) is necessary for variable coefficients. By employing the elimination of common sub-expressions [93] the number of additional operations can be reduced, but overall this impacts the performance by a factor of about $4\times$, see e.g. [86]. In chapter 2 we will present a new method that is $3\times$ more efficient.

The second, even more problematic point is the geometry. So far, we assumed that the geometry is sufficiently approximated by the macro mesh. This works well for polyhedral domains. But it does not hold true for a spherical domain like the Earth’s mantle. Let Ω_S be the original spherical shell domain. No matter how often the initial macro mesh \mathcal{T}_0 is refined, all fine grid nodes will always be part of the convex hull of \mathcal{T}_0 . In other words, refining the mesh will not improve the geometry approximation. To this end a geometry projection of all fine grid points was implemented in HHG [94] such that all surface points are projected outwards onto the spherical surface. For interior points a similar projection is performed such that all points form concentric layers. This can be done in a pre-processing step and ensures that $\cup_{T \in \mathcal{T}_\ell} \bar{T} \rightarrow \Omega_S$ for $\ell \rightarrow \infty$. But it comes at the cost that sub-tetrahedra, and consequently the stencil values will vary for each individual node. An illustration is shown in Fig. 3.1 and 3.2. As a consequence, there is no information that can be re-used and all stencil values have to be re-computed on-the-fly. In particular, this includes the expensive on-the-fly evaluation of E^t via quadrature. So, coming back to our previous example, instead of computing just one stencil and use it $2.7 \cdot 10^6$ times, an expensive computation has to be performed for each of the $2.7 \cdot 10^6$ nodes, which has to be done in each smoothing step and for each residual application. In section 3.6 we show that this literally kills the performance by a factor of $100\times$. In chapter 3 an efficient method for curved geometries will be presented.

1.6 Objective of the thesis

The hybrid grids concept provides several advantages compared to other state of the art convection software that build on AMR techniques. Especially there is no need for complex mesh handling and re-partitioning, which will become particularly expensive for exascale computations. Furthermore, it is based on a matrix-free concept with very low memory footprint. However, in the previous section we also identified two issues with this concept. These represent two major roadblocks for TerraNeo to become a full, exa-ready mantle convection software.

Within this thesis, we will develop new matrix-free approaches that tackle both of these roadblocks. In chapter 2 we will introduce a stencil scaling approach in which pre-computed reference stencils are scaled on-the-fly with the appropriate viscosity values. A rigorous mathematical analysis will be presented and the performance will be discussed in terms of reduced number of operations as well as in terms of memory traffic. Overall, we achieve a performance improvement of $3\times$ compared to the existing approach. In chapter 3 we will

¹ E^t is a 4×4 matrix and there are six different classes of sub-tetrahedra. Thus, $6 \times 4 \times 4 \times 8$ byte.

present a novel approach for an efficient stencil assembly on curved domains. It is based on an approximation of the original stencils by low order polynomials. The method greatly improves the performance and almost achieves the time-to-solutions of the original HHG approach, with the advantage of a correct geometry approximation. The accuracy of the polynomial approximation is numerically tested and verified. This novel method will be extended in chapter 4 to the full Stokes system including variable viscosity values. Its performance and scalability is investigated for a realistic geophysical application scenario. In chapter 5 we present benchmark results for this new method and investigate the effect of lateral viscosity variations on dynamic topography. The global resolution used in these simulations is less than 1.5 km. A summary and outlook will be given in chapter 6.

1.7 References

- [1] H. P. Bunge, M. Richards, C. Lithgow-Bertelloni, J. Baumgardner, S. Grand, and B. Romanowicz, “Time scales and heterogeneous structure in geodynamic earth models,” *Science*, vol. 280, pp. 91 – 95, 1998.
- [2] Y. Ricard, “Physics of Mantle Convection,” in *Treatise of Geophysics* (G. Schubert, ed.), vol. 7, Elsevier, 2007.
- [3] H.-P. Bunge, M. A. Richards, and J. R. Baumgardner, “A sensitivity study of three-dimensional spherical mantle convection at 10^8 rayleigh number: Effects of depth-dependent viscosity, heating mode, and an endothermic phase change,” *Journal of Geophysical Research: Solid Earth*, vol. 102, no. B6, pp. 11991–12007.
- [4] G. F. Davies, *Dynamic Earth: Plates, Plumes and Mantle Convection*. Cambridge University Press, 1999.
- [5] T. Höink and A. Lenardic, “Three-dimensional mantle convection simulations with a low-viscosity asthenosphere and the relationship between heat flow and the horizontal length scale of convection,” *Geophys. Res. Lett.*, vol. 35, p. L10304, 2008.
- [6] B. H. Hager and R. J., “Long-wavelength variations in earth’s geoid: physical models and dynamical implications,” *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 328, no. 1599, pp. 309–327, 1989.
- [7] S. P. Grand, R. D. van der Hilst, and S. Widiyantoro, “Global Seismic Tomography: A Snapshot of Convection in the Earth,” *GSA Today*, vol. 7, no. 4, pp. 1–7, 1997.
- [8] D. P. McKenzie, J. M. Roberts, and N. O. Weiss, “Convection in the earth’s mantle: towards a numerical simulation,” *Journal of Fluid Mechanics*, vol. 62, no. 3, pp. 465–538, 1974.
- [9] J. R. Baumgardner, “Three-dimensional treatment of convective flow in the earth’s mantle,” *Journal of Statistical Physics*, vol. 39, pp. 501–511, Jun 1985.
- [10] H.-P. Bunge and J. Baumgardner, “Mantle convection modeling on parallel virtual machines,” *Computers in Physics*, vol. 9, pp. 207–215, 1995.
- [11] S. Zhong, A. McNamara, E. Tan, L. Moresi, and M. Gurnis, “A benchmark study on mantle convection in a 3-D spherical shell using CitcomS,” *Geochem. Geophys. Geosyst.*, vol. 9, p. Q10017, 2008.
- [12] T. Heister, J. Dannberg, R. Gassmüller, and W. Bangerth, “High accuracy mantle convection simulation through modern numerical methods. II: Realistic models and problems,” *Geophysical Journal International*, vol. 210, no. 2, pp. 833–851, 2017.
- [13] J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. J. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas, “An extreme-scale implicit solver for complex

- pdes: Highly heterogeneous flow in earth's mantle," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, (New York, NY, USA), pp. 5:1–5:12, ACM, 2015.
- [14] L. Colli, S. Ghelichkhan, H.-P. Bunge, and J. Oeser, "Retrodictions of mid paleogene mantle flow and dynamic topography in the atlantic region from compressible high resolution adjoint mantle convection models: Sensitivity to deep mantle viscosity and tomographic input model," *Gondwana Research*, vol. 53, pp. 252 – 272, 2018. Rifting to Passive Margins.
- [15] M. Feldmann, "Prototypes of China's Exascale Supercomputers Point to Some New Realities." <https://www.top500.org/news/prototypes-of-chinas-exascale-supercomputers-point-to-some-new-realities/>. (TOP500, retrieved on 7 September 2018).
- [16] S. Bauer, H.-P. Bunge, D. Drzisga, B. Gmeiner, M. Huber, L. John, M. Mohr, U. Rüde, H. Stengel, C. Waluga, J. Weismüller, G. Wellein, M. Wittmann, and B. Wohlmuth, *Hybrid Parallel Multigrid Methods for Geodynamical Simulations*, pp. 211–235. Cham: Springer International Publishing, 2016.
- [17] G. T. Jarvis and D. P. Mckenzie, "Convection in a compressible fluid with infinite prandtl number," *Journal of Fluid Mechanics*, vol. 96, no. 3, pp. 515–583, 1980.
- [18] L. Alisic, M. Gurnis, G. Stadler, C. Burstedde, and O. Ghattas, "Multi-scale dynamics and rheology of mantle flow with plates," *Journal of Geophysical Research: Solid Earth*, vol. 117, no. B10.
- [19] M. Kronbichler, T. Heister, and W. Bangerth, "High Accuracy Mantle Convection Simulation through Modern Numerical Methods," *Geophysical Journal International*, vol. 191, no. 1, pp. 12–29, 2012.
- [20] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, "Performance and Scalability of Hierarchical Hybrid Multigrid Solvers for Stokes Systems," *SIAM J. Sci. Comput.*, vol. 37, no. 2, pp. C143–C168, 2015.
- [21] G. Strang and G. Fix, *An analysis of the finite element method*. Wellesley-Cambridge Press, Wellesley, MA, second ed., 2008.
- [22] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*. New York: Springer, 1991.
- [23] V. Girault and P. A. Raviart, *Finite Element Methods for Navier-Stokes Equations*. New York: Springer, 1986.
- [24] D. R. Davies, J. H. Davies, P. C. Bollada, and a. P. N. O. Hassan, K. Morgan, "A hierarchical mesh refinement technique for global 3-D spherical mantle convection modelling," *Geoscientific Model Development*, vol. 6, no. 4, pp. 1095–1107, 2013.
- [25] J. R. Baumgardner and P. O. Frederickson, "Icosahedral discretization of the two-sphere," *SIAM Journal on Numerical Analysis*, vol. 22, no. 6, pp. 1107–1115, 1985.

-
- [26] P. J. Tackley, “Modelling compressible mantle convection with large viscosity contrasts in a three-dimensional spherical shell using the yin-yang grid,” *Phys. Earth Planet. Inter.*, vol. 171, pp. 7–18, 2008.
- [27] C. Burstedde, G. Stadler, L. Alisic, L. C. Wilcox, E. Tan, M. Gurnis, and O. Ghattas, “Large-scale adaptive mantle convection simulation,” *Geophysical Journal International*, vol. 192, no. 3, pp. 889–906, 2013.
- [28] TerraNeo, “Teachlet.” http://terraneo.fau.de/Files/teachlet_v1.0_1080p.mp4, 2016.
- [29] M. Kameyama, A. Kageyama, and T. Sato, “Multigrid-based simulation code for mantle convection in spherical shell using yin-yang grid,” *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1, pp. 19 – 32, 2008. Recent Advances in Computational Geodynamics: Theory, Numerics and Applications.
- [30] D. L. Turcotte, K. E. Torrance, and A. T. Hsui, “Convection in the earth’s mantle,” in *Methods in computational physics. Volume 13. New York, Academic Press, Inc., 1973, p. 431-454. Research supported by Cornell University*, vol. 13, pp. 431–454, 1973.
- [31] G. A. Glatzmaier, “Numerical simulations of mantle convection: Time-dependent, three-dimensional, compressible, spherical shell,” *Geophysical & Astrophysical Fluid Dynamics*, vol. 43, no. 2, pp. 223–264, 1988.
- [32] M. Kameyama, A. Kageyama, and T. Sato, “Multigrid iterative algorithm using pseudo-compressibility for three-dimensional mantle convection with strongly variable viscosity,” *Journal of Computational Physics*, vol. 206, pp. 162–181, 2005.
- [33] K. Stemmer, H. Harder, and U. Hansen, “A new method to simulate convection with strongly temperature and pressure-dependent viscosity in a spherical shell: Applications to the Earths mantle,” *Physics of the Earth and Planetary Interiors*, vol. 157, pp. 223–249, 2006.
- [34] G. Stadler, M. Gurnis, C. Burstedde, L. C. Wilcox, L. Alisic, and O. Ghattas, “The dynamics of plate tectonics and mantle flow: From local to global scales,” *Science*, vol. 329, no. 5995, pp. 1033–1038, 2010.
- [35] P. J. Tackley, “Effects of strongly variable viscosity on three-dimensional compressible convection in planetary mantles,” *J. Geophys. Res.*, vol. 101, pp. 3311–3332, 1996.
- [36] W.-S. Yang and J. R. Baumgardner, “A matrix-dependent multigrid method for strongly variable viscosity infinite prandtl number thermal convection,” *Geophysical and Astrophysical Fluid Dynamics*, vol. 92, no. 3-4, pp. 151–195, 2000.
- [37] J. Weismüller, B. Gmeiner, S. Ghelichkhan, M. Huber, L. John, B. Wohlmuth, U. Rüde, and H.-P. Bunge, “Fast asthenosphere motion in high-resolution global mantle flow models,” *Geophysical Research Letters*, vol. 42, no. 18, pp. 7429–7435, 2015.
- [38] A. Ringwood, “Phase transformations and their bearing on the constitution and dynamics of the mantle,” *Geochimica et Cosmochimica Acta*, vol. 55, no. 8, pp. 2083 – 2110, 1991.

- [39] P. J. Tackley, D. J. Stevenson, G. A. Glatzmaier, and G. Schubert, “Effects of multiple phase transitions in a three-dimensional spherical model of convection in earth’s mantle,” *J. Geophys. Res.*, vol. 99(B8), pp. 15877–15901, 1994.
- [40] L. Stixrude and C. Lithgow-Bertelloni, “Thermodynamics of mantle minerals - II. Phase equilibria,” *Geophysical Journal International*, vol. 184, no. 3, pp. 1180–1213, 2011.
- [41] C. Burstedde, O. Ghattas, M. Gurnis, G. Stadler, E. Tan, T. Tu, L. C. Wilcox, and S. Zhong, “Scalable adaptive mantle convection simulation on petascale supercomputers,” in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC ’08, (Piscataway, NJ, USA), pp. 62:1–62:15, IEEE Press, 2008.
- [42] C. Burstedde, L. C. Wilcox, and O. Ghattas, “Scalable algorithms for parallel adaptive mesh refinement on forests of octrees,” *SIAM J. Sci. Comput.*, vol. 33, no. 3, pp. 1103–1133, 2011.
- [43] D. R. Davies, C. R. Wilson, and S. C. Kramer, “Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics,” *Geochem. Geophys. Geosyst.*, vol. 12, no. 6, p. 20 pp., 2011.
- [44] C. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995.
- [45] R. Gassmüller, E. Heien, E. Puckett, and W. Bangerth, “Flexible and scalable particle-in-cell methods for massively parallel computations,” *preprint arXiv:1612.03369*, 2016.
- [46] W. Hackbusch, *Multigrid methods and applications*. Berlin: Springer, 1985.
- [47] U. Trottenberg, U., C. W. Oosterlee, and A. Schuller, *Multigrid*. Elsevier Science, 2000.
- [48] R. Verfürth, “A Combined Conjugate Gradient-Multigrid Algorithm for the Numerical Solution of the Stokes Problem,” *IMA Journal of Numerical Analysis*, vol. 4, pp. 441–455, 1984.
- [49] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, *Scaling Hypr’s Multigrid Solvers to 100,000 Cores*, pp. 261–279. London: Springer London, 2012.
- [50] B. Gmeiner, M. Huber, L. John, U. Rude, and B. Wohlmuth, “A quantitative performance study for stokes solvers at the extreme scale,” *Journal of Computational Science*, vol. 17, no. Part 3, pp. 509 – 521, 2016. Recent Advances in Parallel Techniques for Scientific Computing.
- [51] Oak Ridge National Laboratory (ORNL), “Summit.” <https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/> (retrieved on 2 September 2018).
- [52] TOP500, “June 2018.” <https://www.top500.org/lists/2018/06/>.
- [53] C. Yang, W. Xue, H. Fu, H. You, X. Wang, Y. Ao, F. Liu, L. Gan, P. Xu, L. Wang, G. Yang, and W. Zheng, “10m-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’16, (Piscataway, NJ, USA), pp. 6:1–6:12, IEEE Press, 2016.

-
- [54] H. Fu, C. He, B. Chen, Z. Yin, Z. Zhang, W. Zhang, T. Zhang, W. Xue, W. Liu, W. Yin, G. Yang, and X. Chen, “18.9pflops nonlinear earthquake simulation on sunway taihulight: Enabling depiction of 18-hz and 8-meter scenarios,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '17*, (New York, NY, USA), pp. 2:1–2:12, ACM, 2017.
- [55] National Energy Research Scientific Computing Center (NERSC), “Cori.” <http://www.nersc.gov/users/computational-systems/cori/configuration/> (retrieved on 2 September 2018).
- [56] C. Uphoff, S. Rettenberger, M. Bader, E. H. Madden, T. Ulrich, S. Wollherr, and A.-A. Gabriel, “Extreme scale multi-physics simulations of the tsunamigenic 2004 sumatra megathrust earthquake,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '17*, (New York, NY, USA), pp. 21:1–21:16, ACM, 2017.
- [57] Swiss National Supercomputing Centre (CSCS), “Piz Daint & Piz Dora .” <https://www.cscs.ch/computers/piz-daint/> (retrieved on 2 September 2018).
- [58] J. Langguth, M. Sourouri, G. T. Lines, S. B. Baden, and X. Cai, “Scalable heterogeneous cpu-gpu computations for unstructured tetrahedral meshes,” *IEEE Micro*, vol. 35, pp. 6–15, July 2015.
- [59] K.-R. Wichmann, M. Kronbichler, R. Löhner, and W. A. Wall, “Practical applicability of optimizations and performance models to complex stencil-based loop kernels in cfd,” *The International Journal of High Performance Computing Applications*, vol. 0, no. 0, p. 1094342018774126, 2018.
- [60] R. Torres, L. Lindarkis, J. Kunkel, and T. Ludwig, “ICON DSL: A domain-specific language for climate modeling,” in *WOLFHPC 2013 Third International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing*, 11 2013.
- [61] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess, “STELLA: A domain-specific tool for structured grid methods in weather and climate models,” in *Proceedings International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 41:1–41:12, ACM, nov 2015.
- [62] M. Huber, B. Gmeiner, U. Rude, and B. Wohlmuth, “Resilience for Massively Parallel Multigrid Solvers,” *SIAM Journal on Scientific Computing*, vol. 38, pp. S217–S239, Jan. 2016.
- [63] S. Lohr, “Move Over, China: U.S. Is Again Home to World’s Speediest Supercomputer.” <https://www.nytimes.com/2018/06/08/technology/supercomputer-china-us.html>. (The New York Times, retrieved on 10 February 2019).
- [64] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*. Computational Science Series, CRC Press, 2011.
- [65] LRZ, “SuperMUC Petascale System.” <https://www.lrz.de/services/compute/supermuc/systemdescription/> (retrieved on 2 September 2018).

- [66] D. E. Comer, *Essentials of Computer Architecture*. Pearson Prentice Hall, New Jersey, 2005.
- [67] S. Williams, A. Waterman, and D. Patterson, “Roofline: An insightful visual performance model for multicore architectures,” *Commun. ACM*, vol. 52, pp. 65–76, Apr. 2009.
- [68] A. Ilic, F. Pratas, and L. Sousa, “Cache-aware Roofline model: Upgrading the loft,” *IEEE Computer Architecture Letters*, vol. 13, no. 1, pp. 21–24, 2013.
- [69] S. Bauer, D. Drzisga, M. Mohr, U. Rde, C. Waluga, and B. Wohlmuth, “A stencil scaling approach for accelerating matrix-free finite element implementations,” *SIAM Journal on Scientific Computing*, vol. 40, no. 6, pp. C748–C778, 2018.
- [70] H. Stengel, J. Treibig, G. Hager, and G. Wellein, “Quantifying performance bottlenecks of stencil computations using the execution-cache-memory model,” in *Proceedings of the 29th International Conference on Supercomputing, ICS ’15*, (New York, NY, USA), pp. 207–216, ACM, 2015.
- [71] G. Hager, J. Treibig, J. Habich, and G. Wellein, “Exploring performance and power properties of modern multicore chips via simple machine models,” *Concurrency and Computation: Practice and Experience*, 2014.
- [72] C. Lameter, “Numa (non-uniform memory access): An overview,” *Queue*, vol. 11, pp. 40:40–40:51, July 2013.
- [73] C. Hollowell, C. Caramarcu, W. Strecker-Kellogg, A. Wong, and A. Zaytsev, “The effect of numa tunings on cpu performance,” *Journal of Physics: Conference Series*, vol. 664, no. 9, p. 092010, 2015.
- [74] G. F. Carey and B.-N. Jiang, “Element-by-element linear and nonlinear solution schemes,” *Communications in Applied Numerical Methods*, vol. 2, no. 2, pp. 145–153, 1986.
- [75] C. Flaig and P. Arbenz, “A Highly Scalable Matrix-Free Multigrid Solver for μ FE Analysis Based on a Pointer-Less Octree,” in *Large-Scale Scientific Computing: 8th International Conference, LSSC 2011, Sozopol, Bulgaria, June 6-10, 2011, Revised Selected Papers* (I. Lirkov, S. Margenov, and J. Waśniewski, eds.), pp. 498–506, Springer Berlin Heidelberg, 2012.
- [76] M. Kronbichler and K. Kormann, “A generic interface for parallel cell-based finite element operator application,” *Computers and Fluids*, vol. 63, pp. 135–147, 2012.
- [77] D. A. May, J. Brown, and L. L. Pourhiet, “A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow,” *Computer Methods in Applied Mechanics and Engineering*, vol. 290, pp. 496–523, 2015.
- [78] K. Ljungkvist, “Matrix-free Finite-element Computations on Graphics Processors with Adaptively Refined Unstructured Meshes,” in *Proceedings of the 25th High Performance Computing Symposium, HPC ’17*, pp. 1:1–1:12, Society for Computer Simulation International, 2017.

-
- [79] J. Brown, “Efficient Nonlinear Solvers for Nodal High-Order Finite Elements in 3D,” *J. Scientific Computing*, vol. 45, no. 1-3, pp. 48–63, 2010.
- [80] K. Ljungkvist and M. Kronbichler, “Multigrid for Matrix-Free Finite Element Computations on Graphics Processors,” Tech. Rep. 2017-006, Department of Information Technology, Uppsala University, 2017.
- [81] German Priority Programme 1648, “Software for exascale computing.” <http://www.sppexa.de/>, 2018.
- [82] W. Bangerth, R. Hartmann, and G. Kanschat, “deal.II – a general purpose object oriented finite element library,” *ACM Trans. Math. Softw.*, vol. 33, no. 4, pp. 24/1–24/27, 2007.
- [83] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmüller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells, “The deal.II library, version 9.0,” *Journal of Numerical Mathematics*, 2018, accepted.
- [84] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, “An overview of the trilinos project,” *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 397–423, 2005.
- [85] B. Bergen and F. Hülsemann, “Hierarchical hybrid grids: data structures and core algorithms for multigrid,” *Numerical Linear Algebra with Applications*, vol. 11, pp. 279–291, 2004.
- [86] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, “Towards textbook efficiency for parallel multigrid,” *Numer. Math. Theory Methods Appl.*, vol. 8, 2015.
- [87] N. Kohl, D. Thönnies, D. Drzisga, D. Bartuschat, and U. Rüde, “The HyTeG finite–element software framework for scalable multigrid solvers,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 0, no. 0, pp. 1–20, 2018.
- [88] J. Bey, “Tetrahedral grid refinement,” *Computing*, vol. 55, no. 4, pp. 355–378, 1995.
- [89] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, “PETSc Web page.” <http://www.mcs.anl.gov/petsc>, 2018.
- [90] P. R. Amestoy, A. Guermouche, J.-Y. L’Excellent, and S. Pralet, “Hybrid scheduling for the parallel solution of linear systems,” *Parallel Computing*, vol. 32, no. 2, pp. 136–156, 2006.
- [91] M. Huber, *Massively parallel and fault-tolerant multigrid solvers on peta-scale systems*. PhD thesis, Technical University of Munich, 2018.
- [92] A. Baker, R. Falgout, T. Kolev, and U. Yang, “Multigrid smoothers for ultraparallel computing,” *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2864–2887, 2011.

- [93] B. Gmeiner, *Design and Analysis of Hierarchical Hybrid Multigrid Methods for Peta-Scale Systems and Beyond*. PhD thesis, University of Erlangen-Nuremberg, 2013.
- [94] J. Weismüller, *Development and application of high performance software for mantle convection modeling*. PhD thesis, Ludwig-Maximilians-Universität München, 2015.

Chapter 2

Stencil scaling approach for variable coefficient operators

This chapter includes parts of a manuscript that was published in the *SIAM Journal on Scientific Computing* in 2018 [1]. We will present here all necessary lemmata to develop the theory. For the proofs we refer the interested reader to the full manuscript.

In this chapter we introduce a new operator that is obtained by appropriately scaling the reference stiffness matrix from the constant coefficient case. Assuming sufficient regularity, an a priori analysis shows that solutions obtained by this approach are unique and have asymptotically optimal order convergence in the H^1 - and the L^2 -norm on hierarchical hybrid grids. For the pre-asymptotic regime, we present a local modification that guarantees uniform ellipticity of the operator. Cost considerations show that our novel approach requires roughly one third of the floating-point operations compared to a classical finite element assembly scheme employing nodal integration. Our theoretical considerations are illustrated by numerical tests that confirm the expectations with respect to accuracy and run-time. A large scale application with more than a hundred billion ($1.6 \cdot 10^{11}$) degrees of freedom executed on 14310 compute cores demonstrates the efficiency of the new scaling approach.

In mantle convection simulations this method will be applied for the elliptic operator \mathbf{A} of the Stokes system that includes the viscosity μ as PDE coefficient. This chapter establishes the method for a Darcy type PDE. An extension to the Stokes system is possible, but the treatment of the cross-coupling terms will require special care that goes beyond the scope of this thesis.

2.1 Motivation of scaling approach

Let us now consider the setting of an elliptic PDE with piecewise smooth variable coefficients, assuming that the macro mesh resolves jumps in the coefficients. In this case, a standard finite element formulation is based on quadrature formulas and introduces a variational crime. According to [2, 3], there is flexibility how the integrals are approximated without degenerating the order of convergence.

For this chapter, we restrict ourselves to the lowest order case of conforming finite elements on simplicial meshes. Then the most popular quadrature formula is the one point Gauss rule which in the simplest case of $\text{div}(k\nabla u)$ as PDE operator just weights the element based reference stiffness matrix of the Laplacian by the factor of $k(x_T)$ where x_T is the barycenter of

the element T . Alternatively, one can select a purely vertex-based quadrature formula. Here, the weighting of the element matrix is given by $\sum_{i=1}^{d+1} k(x_T^i)/(d+1)$, where d is the space dimension and x_T^i are the vertices of element T . Using a vertex-based quadrature formula saves function evaluations and is, thus, attractive whenever the evaluation of the coefficient function is expensive and it pays off to reuse once computed values in several element stiffness matrices. Note that reusing barycentric data on general unstructured meshes will require nontrivial storage schemes.

In the case of variable coefficient functions, stencil entries can vary from one mesh node to another. The number of possibly different stencils within each macro element becomes $\frac{1}{d!}2^{d\ell} + \mathcal{O}(2^{(d-1)\ell})$, where ℓ is the number of uniform refinement steps for HHG. Now we can resort to two options: Either these stencils are computed once and then saved, effectively creating a sparse matrix data structure, or they are computed on-the-fly each time when they are needed. Neither of these techniques is ideal for extreme scale computations. While for the first option $\mathcal{O}(2^{d\ell})$ extra memory is consumed and extensive memory traffic occurs, the second option requires re-computation of $\mathcal{O}(2^{d\ell})$ local contributions.

The efficiency of a numerical PDE solver can be analyzed following the *textbook paradigm* [4] that defines a work unit (WU) to be the cost of one application of the discrete operator for a given problem. With this definition, the analysis of iterative solvers can be conducted in terms of WU. Classical multigrid textbook efficiency is achieved when the solution is obtained in less than 10 WU. For devising an efficient method it is, however, equally critical to design algorithms that reduce the cost of a WU without sacrificing accuracy. Clearly, the real life cost of a WU depends on the computer hardware and the efficiency of the implementation, as e.g., analyzed for parallel supercomputers in [5]. On the other side, matrix-free techniques, as the one proposed in this chapter, seek opportunities to reduce the cost of a WU by a clever rearrangement of the algorithms or by exploiting approximations where this is possible; see e.g., also [6].

These preliminary considerations motivate our novel approach to reduce the cost of a WU by recomputing the surrogate stencil entries for a matrix-free solver more efficiently. We find that these values can be assembled from a reference stencil of the constant coefficient case which is scaled appropriately using nodal values of the coefficient function. We will show that under suitable conditions, this technique does not sacrifice accuracy. However, we also demonstrate that the new method can reduce the cost of a WU considerably and in consequence helps to reduce the time-to-solution.

The rest of this chapter is structured as follows: In section 2.2, we define our new scaling approach. The variational crime is analyzed in section 2.3 where optimal order a priori results for the L^2 - and H^1 -norm are obtained. In section 2.4, we consider modifications in the pre-asymptotic regime to guarantee uniform ellipticity. Section 2.5 is devoted to the reproduction property and the primitive concept which allows for a fast on-the-fly reassembling in a matrix-free software framework. In section 2.6, we discuss the cost compared to a standard nodal based element-wise assembling. Finally, in section 2.7 we perform numerically an accuracy study and a run-time comparison to illustrate the performance gain of the new scaling approach.

2.2 Problem setting and definition of the scaling approach

We consider a scalar elliptic partial differential equation of Darcy type, i.e.,

$$-\operatorname{div} K \nabla u = f, \quad \text{in } \Omega, \quad \operatorname{tr} u = 0 \quad \text{on } \partial\Omega$$

where tr stands for the boundary trace operator and $f \in L^2(\Omega)$. Here $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, is a bounded polygonal/polyhedral domain, and K denotes a uniformly positive and symmetric tensor with coefficients specified through a number of functions k_m , $m = 1, \dots, M$, where $M \leq 3$ in 2D and $M \leq 6$ in 3D due to symmetry.

For the Darcy operator with a scalar uniform positive permeability, i.e., $-\operatorname{div}(k \nabla u)$, we can set $M = 1$ and $k_1 := k$. The above setting also covers blending finite elements approaches [7]. Here K is related to the Jacobian of the blending function. For example, if the standard Laplacian model problem is considered on the physical domain Ω_{phy} but the actual assembly is carried out on a reference domain $\Omega := \Phi(\Omega_{\text{phy}})$, we have

$$a(v, w) = \int_{\Omega_{\text{phy}}} \nabla v_{\text{phy}} \cdot \nabla w_{\text{phy}} \, dx_{\text{phy}} = \int_{\Omega} \nabla v \cdot \frac{(D\Phi)(D\Phi)^\top}{|\det D\Phi|} \nabla w \, dx, \quad (2.1)$$

where $D\Phi$ is the Jacobian of the mapping Φ , and $v_{\text{phy}} := v \circ \Phi$, $w_{\text{phy}} := w \circ \Phi$.

2.2.1 Definition of our scaling approach

The weak form associated with the partial differential equation is defined in terms of the bilinear form $a(v, w) := \int_{\Omega} \nabla v \cdot K \nabla w \, dx$, and the weak solution $u \in V_0 := H_0^1(\Omega)$ satisfies:

$$a(u, v) = (f, v), \quad v \in V_0.$$

This bilinear form can be affinely decomposed as

$$a(v, w) := \sum_{m=1}^M a_m(v, w), \quad a_m(v, w) := \int_{\Omega} k_m(x) (D_m v, D_m w) \, dx, \quad v, w \in V := H^1(\Omega) \quad (2.2)$$

where D_m is a first order partial differential operator and (\cdot, \cdot) stands for some suitable inner product. In the case of a scalar permeability we find $D_1 := \nabla$ and (\cdot, \cdot) stands for the scalar product in \mathbb{R}^d . While for (2.1) in 2D one can, as one alternative, e.g. define

$$\begin{aligned} k_1 &:= (K_{11} - K_{12}), & k_2 &:= K_{12}, & k_3 &:= (K_{22} - K_{11}), \\ D_1 &:= \nabla, & D_2 &:= \partial/\partial x + \partial/\partial y, & D_3 &:= \partial/\partial y, \end{aligned}$$

where $K = (K_{ij})$ and the same scalar product (\cdot, \cdot) as above. Note that this decomposition reduces to the one in case of a scalar permeability, i.e. for $K = \operatorname{diag}(k, k)$.

Let \mathcal{T}_H , $H > 0$ fixed, be a possibly unstructured simplicial triangulation resolving Ω . We call \mathcal{T}_H also macro-triangulation and denote its elements by T . Using uniform mesh refinement, we obtain $\mathcal{T}_{h/2}$ from \mathcal{T}_h by decomposing each element into 2^d sub-elements, $h \in \{H/2, H/4, \dots\}$; see [8] for the 3D case. The elements of \mathcal{T}_h are denoted by t . The macro-triangulation is then decomposed into the following geometrical primitives: *elements*, *faces*, *edges*, and *vertices*. Each of these geometric primitives acts as a container for a subset

of unknowns associated with the refined triangulations. These sets of unknowns can be stored in array-like data structures, resulting in a contiguous memory layout that conforms inherently to the refinement hierarchy; see [5, 9]. In particular, the unknowns can be accessed without indirect addressing such that the overhead is reduced significantly when compared to conventional sparse matrix data structures. Associated with \mathcal{T}_h is the space $V_h \subset V$ of piecewise linear finite elements. In V_h , we do not include the homogeneous boundary conditions. We denote by $\phi_i \in V_h$ the nodal basis functions associated to the i -th mesh node. Node i is located at the vertex x_i . For $v_h := \sum_i \nu_i \phi_i$ and $w_h := \sum_j \chi_j \phi_j$, we define our scaled discrete bilinear forms $a_h(\cdot, \cdot)$ and $a_m^h(\cdot, \cdot)$ by

$$a_h(v_h, w_h) := \sum_{m=1}^M a_m^h(v_h, w_h), \quad (2.3a)$$

$$a_m^h(v_h, w_h) := \frac{1}{4} \sum_{T \in \mathcal{T}_H} \sum_{i,j} (k_m|_T(x_i) + k_m|_T(x_j)) (\nu_i - \nu_j) (\chi_j - \chi_i) \int_T (D_m \phi_i, D_m \phi_j) dx. \quad (2.3b)$$

This definition is motivated by the fact that $a_m(v_h, w_h)$ can be written as

$$a_m(v_h, w_h) = \frac{1}{2} \sum_{T \in \mathcal{T}_H} \sum_{i,j} (\nu_i - \nu_j) (\chi_j - \chi_i) \int_T k_m(x) (D_m \phi_i, D_m \phi_j) dx. \quad (2.4)$$

Here we have exploited symmetry and the row sum property. It is obvious that if k_m is a constant restricted to T , we do obtain $a_m^h(v_h, w_h) = a_m(v_h, w_h)$. In general however, the definition of $a_h(\cdot, \cdot)$ introduces a variational crime and it does not even correspond to an element-wise local assembling based on a quadrature formula. We note that each node on ∂T is redundantly existent in the data structure and that we can easily account for jumps in the coefficient function when resolved by the macro-mesh elements T .

Similar scaling techniques have been used in [10] for a generalized Stokes problem from geodynamics with coupled velocity components. However, for vectorial equations such a simple scaling does asymptotically not result in a physically correct solution. For the computation of integrals on triangles containing derivatives in the form of (2.2), cubature formulas of the form (2.3b) in combination with Euler-MacLaurin type asymptotic expansions have been applied [11, Table 1].

Remark 1 *At first glance the Definition (2.3b) might not be more attractive than (2.4) regarding the computational cost. In a matrix-free approach, however, where we have to re-assemble the entries in each matrix call, (2.3b) turns out to be much more favorable. In order to see this, we have to recall that we work with hybrid hierarchical meshes. This means that for each inner node i in T , we find the same entries in the sense that*

$$\int_T (D_m \phi_i, D_m \phi_j) dx = \int_T (D_m \phi_l, D_m \phi_{x_j + \delta x}) dx.$$

Here we have identified the index notation with the vertex notation, and the vertex x_l is obtained from the vertex x_i by a shift of δx , i.e., $x_l = x_i + \delta x$. Consequently, the values of $\int_T (D_m \phi_i, D_m \phi_j) dx$ do not have to be re-computed but can be efficiently stored.

For simplicity of notation, we shall restrict ourselves in the following to the case of the Darcy equation with a scalar uniformly positive definite permeability; i.e., $M = 1$ and drop the index m . However, the proofs in section 2.3 can be generalized to conceptually the same type of results for $M > 1$. In section 2.7.3, we also show numerical results for $M = 6$ in 3D.

2.2.2 Stencil structure

We exploit the hierarchical grid structure to save a significant amount of memory compared to classical sparse matrix formats. Any direct neighbor $x_j \in \mathcal{N}_T(x_i)$ can be described through a direction vector w_j such that $x_j = x_i + w_j$. The regularity of the grid in the interior of a macro element T implies that these vectors remain the same, when we move from one node to another node. Additionally, for each neighbor $x_j \in \mathcal{N}_T(x_i) \setminus \{x_i\}$ there is a mirrored neighbor x'_j of x_i reachable by $w_j = -w_{j'}$; see Fig. 2.1.

Let $n_i = |\mathcal{N}_T(x_i)|$ denote the stencil size at mesh node x_i . We define the stencil $\hat{s}_{x_i}^T \in \mathbb{R}^{n_i}$ associated to the i -th mesh node x_i restricted on T as

$$(\hat{s}_{x_i}^T)_j := \int_T (\nabla \phi_{x_i+w_j}, \nabla \phi_{x_i}) \, dx .$$

The symmetry of the bilinear form yields

$$(\hat{s}_{x_i}^T)_j = (\hat{s}_{x_i+w_j}^T)_{j'} .$$

We recall that for each mesh node x_i we have $n_i \leq 7$ in 2D and $n_i \leq 15$ in 3D. Out of these entries only 3 in 2D and 7 in 3D have to be computed since the remaining ones follow from symmetry arguments and the observation that $\sum_j (\hat{s}_{x_i}^T)_j = 0$.

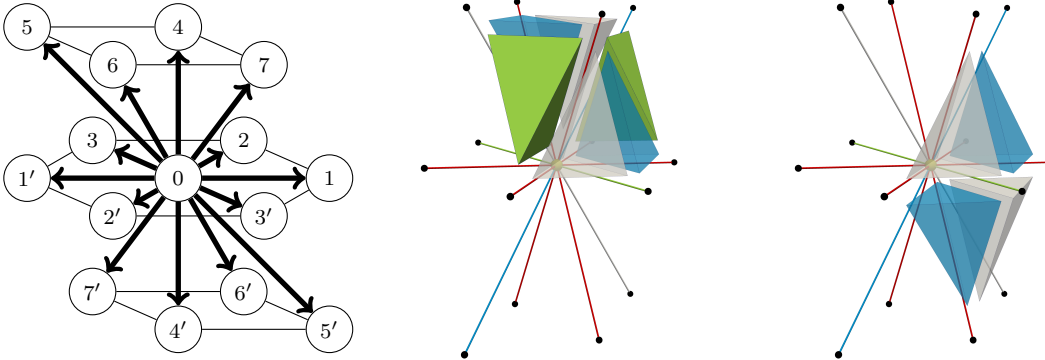


Figure 2.1: From left to right: Exemplary local indices j and their corresponding direction vectors of a 15 point stencil in 3D; Six elements attached to one edge; Four elements attached to one edge

Due to the hierarchical hybrid grid structure, two stencils $\hat{s}_{x_{i_1}}^T$ and $\hat{s}_{x_{i_2}}^T$ are exactly the same if x_{i_1} and x_{i_2} are two nodes belonging to the same primitive; i.e., we find only 15 different kinds of stencils per macro element in 3D, one for each of its 15 primitives (4 vertices, 6 edges, 4 faces, 1 volume), and 7 in 2D. This observation allows for an extremely fast and memory-efficient on-the-fly (re)assembly of the entries of the stiffness matrix in stencil form. For each node x_i in the data structure, we save the nodal values of the coefficient function k . With these considerations in mind, the bilinear form (2.3b) can be evaluated very efficiently and requires only a suitable scaling of the reference entries; see section 2.6 for detailed cost considerations.

2.3 Variational crime framework and a priori analysis

In order to obtain order h and h^2 a priori estimates of the modified finite element approximation in the H^1 - and L^2 -norm, respectively, we analyze the discrete bilinear form. From now on, we assume that $k|_T \in W^{2,\infty}(T)$ for each $T \in \mathcal{T}_H$. Moreover, we denote by $\|\cdot\|_0$ the L^2 -norm on Ω and $\|\cdot\|_\infty := \sup_{T \in \mathcal{T}_H} \|\cdot\|_{\infty;T}$ defines a broken L^∞ -norm. We recall that the coefficient function is only assumed to be element-wise smooth with respect to the macro triangulation. Existence and uniqueness of a finite element solution $u_h \in V_h \cap V_0$ of

$$a_h(u_h, v_h) = f(v_h), \quad v_h \in V_h \cap V_0$$

is given provided that the following assumption (A1) holds true:

(A1) $a_h(\cdot, \cdot)$ is uniformly coercive on $V_h \cap V_0$

(A2) $|a(v_h, w_h) - a_h(v_h, w_h)| \lesssim h \|\nabla k\|_\infty \|\nabla v_h\|_0 \|\nabla w_h\|_0, \quad v_h, w_h \in V_h$

Here and in the following, the notation \lesssim is used as abbreviation for $\leq C$, where $C < \infty$ is independent of the mesh-size h . The assumption (A2), if combined with Strang's first lemma, yields that the finite element solution results in $\mathcal{O}(h)$ a priori estimates with respect to the H^1 -norm; see, e.g., [2, 3]. We note that for h small enough, the uniform coercivity (A1) follows from the consistency assumption (A2), since for $v_h \in V_h$

$$a_h(v_h, v_h) \geq a(v_h, v_h) - |a_h(v_h, v_h) - a(v_h, v_h)| \geq C(1 - ch) \|\nabla v_h\|_0^2.$$

Remark 2 *As it is commonly done in the finite element analysis in unweighted Sobolev norms, we allow the generic constant C to be dependent on the global contrast of k defined by $\sup_\Omega k / \inf_\Omega k$. Numerical results, however, show that the resulting bounds may be overly pessimistic for coefficients with large global variations. In [12] and the references therein, methods to improve the bounds in this case are presented. The examples show that the bounds may be improved significantly for coefficients with a global contrast in the magnitude of about 10^5 . We are mainly interested in showing alternative assembly techniques to the standard finite element method and in comparing them to the well-established approaches in standard norms. Moreover, in our modification only the local variation of the coefficient k is important, therefore we shall not work out these subtleties here.*

2.3.1 Abstract framework for L^2 -norm estimates

Since (A2) does not automatically guarantee optimal order L^2 -estimates, we employ duality arguments. To get a better feeling on the required accuracy of $a_h(\cdot, \cdot)$, we briefly recall the basic steps occurring in the proof of the upper bound. As it is standard, we assume H^2 -regularity of the primal and the dual problem. Restricting ourselves to the case of homogeneous Dirichlet boundaries, the dual PDE and boundary operators coincide with the primal ones. Let us denote by $P_h u$ the standard Galerkin approximation of u , i.e., the finite element solution obtained as the solution of a discrete problem using the bilinear form $a(\cdot, \cdot)$. It is well-known that under the given assumptions $\|u - P_h u\|_0 = \mathcal{O}(h^2)$. Now, to obtain an L^2 -estimate for u_h , we consider the dual problem with $u_h - P_h u$ on the right-hand side. Let $w \in V_0$ be the solution of $a(v, w) = (u_h - P_h u, v)_0$ for $v \in V_0$. Due to the standard Galerkin orthogonality, we obtain

$$\|u_h - P_h u\|_0^2 = a(u_h - P_h u, w) = a(u_h - P_h u, P_h w) = a(u_h, P_h w) - a_h(u_h, P_h w). \quad (2.5)$$

This straightforward consideration shows us that compared to (A2), we need to make stronger assumptions on the mesh-dependent bilinear form $a_h(\cdot, \cdot)$. We define (A3) by

$$(A3) \quad |a(v_h, w_h) - a_h(v_h, w_h)| \lesssim h^2 \|Hk\|_\infty \|\nabla v_h\|_0 \|\nabla w_h\|_0 + h \|\nabla k\|_\infty \|\nabla v_h\|_{0;S_h} \|\nabla w_h\|_{0;S_h},$$

where Hk denotes the Hessian of k and $S_h := \cup_{T \in \mathcal{T}_H} S_h(T)$ with $S_h(T) := \{t \in \mathcal{T}_h; \partial t \cap \partial T \neq \emptyset\}$; see Fig. 2.2 for a 2D illustration. The semi-norm $\|\cdot\|_{0;S_h}$ stands for the L^2 -norm restricted to S_h .

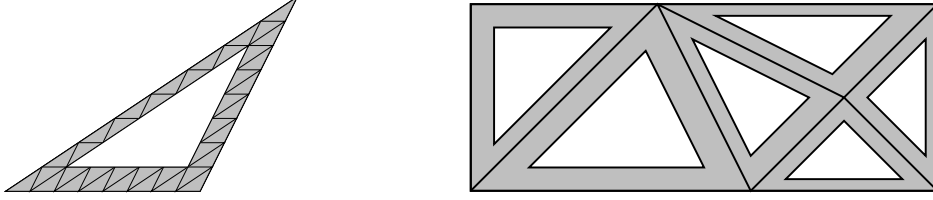


Figure 2.2: Elements in $S_h(T)$ (left) and S_h (right) for $d = 2$

Lemma 1 *Let the problem under consideration be H^2 -regular, h be sufficiently small and (A3) be satisfied. Then we obtain a unique solution and optimal order convergence in the H^1 - and the L^2 -norm, i.e.,*

$$\|u_h - u\|_0 + h \|\nabla(u_h - u)\|_0 \lesssim h^2 (\|Hu\|_0 + \|Hk\|_\infty \|\nabla u\|_0 + \|\nabla k\|_\infty (\|\nabla u\|_0 + \|Hu\|_0)). \quad (2.6)$$

Proof 1 see [1];

2.3.2 Verification of the assumptions

It is well-known [2] that assumptions (A1)-(A3) are satisfied for the bilinear form

$$\tilde{a}_h(u_h, v_h) := \sum_{t \in \mathcal{T}_h} \frac{|t|}{d+1} \sum_{i=1}^{d+1} k|_t(x_i^t) \nabla u_h|_t(x_i^t) \cdot \nabla v_h|_t(x_i^t) = \sum_{t \in \mathcal{T}_h} \bar{k}_t \int_t \nabla u_h \cdot \nabla v_h \, dx, \quad (2.7)$$

Here x_i^t denotes the vertices of the d -dimensional simplex t , i.e., we approximate the integral by a nodal quadrature rule and $\bar{k}_t := (\sum_{i=1}^{d+1} k|_t(x_i^t))/(d+1)$. Thus, to verify the assumptions also for $a_h(\cdot, \cdot)$, it is sufficient to consider $a_h(v_h, w_h) - \tilde{a}_h(v_h, w_h)$ in more detail with $a_h(\cdot, \cdot)$ given by (2.3). Let

$$\hat{A}_t := \begin{pmatrix} a_{1,1}^t & a_{1,2}^t & \cdots & a_{1,d+1}^t \\ a_{1,2}^t & a_{2,2}^t & \cdots & a_{2,d+1}^t \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,d+1}^t & a_{2,d+1}^t & \cdots & a_{d+1,d+1}^t \end{pmatrix}, \quad K_t := \begin{pmatrix} k_{1,1}^t & k_{1,2}^t & \cdots & k_{1,d+1}^t \\ k_{1,2}^t & k_{2,2}^t & \cdots & k_{2,d+1}^t \\ \vdots & \vdots & \ddots & \vdots \\ k_{1,d+1}^t & k_{2,d+1}^t & \cdots & k_{d+1,d+1}^t \end{pmatrix},$$

be the local stiffness matrix associated with the nodal basis functions ϕ_i^t , i.e., $a_{i,j}^t := \int_t \nabla \phi_i^t \cdot \nabla \phi_j^t \, dx$ and the local coefficient function with $k_{i,j}^t := \frac{1}{2} (k|_t(x_i^t) + k|_t(x_j^t))$, $i \neq j$. The diagonal entries of K_t are defined differently as

$$k_{i,i}^t := \frac{-1}{a_{i,i}^t} \sum_{j \neq i} k_{i,j}^t a_{i,j}^t. \quad (2.8)$$

We introduce the component-wise Hadamard product between two matrices as $(B \circ C)_{ij} := B_{ij}C_{ij}$ and define the rank one matrix \tilde{K}_t by $(\tilde{K}_t)_{ij} := \bar{k}_t$.

Due to the symmetry of \hat{A}_t and the fact that the row sum of \hat{A}_t is equal to zero, we can rewrite the discrete bilinear forms. With $v_h, w_h \in V_h$, we associate locally elements $\mathbf{v}_t, \mathbf{w}_t \in \mathbb{R}^{d+1}$ with $(\mathbf{v}_t)_i := v_h(x_i^t)$, $(\mathbf{w}_t)_i := w_h(x_i^t)$. We recall that if $v_h, w_h \in V_h \cap V_0$ and x_i^t is a boundary node, then $v_h(x_i^t) = 0 = w_h(x_i^t)$.

Lemma 2 *The bilinear forms given by (2.3) and (2.7) have the algebraic form*

$$a_h(v_h, w_h) = \frac{1}{2} \sum_{t \in \mathcal{T}_h} \sum_{i,j=1}^{d+1} ((\mathbf{v}_t)_i - (\mathbf{v}_t)_j) (K_t \circ \hat{A}_t)_{ij} ((\mathbf{w}_t)_j - (\mathbf{w}_t)_i), \quad v_h, w_h \in V_h, \quad (2.9)$$

$$\tilde{a}_h(v_h, w_h) = \frac{1}{2} \sum_{t \in \mathcal{T}_h} \sum_{i,j=1}^{d+1} ((\mathbf{v}_t)_i - (\mathbf{v}_t)_j) (\tilde{K}_t \circ \hat{A}_t)_{ij} ((\mathbf{w}_t)_j - (\mathbf{w}_t)_i), \quad v_h, w_h \in V_h. \quad (2.10)$$

Proof 2 *see [1];*

The implication of this lemma for large scale simulations cannot be underestimated. In 2D, the number of different edge types per macro element is three while in 3D it is seven assuming uniform refinement. All edges in 3D in the interior of a macro element T share only four or six elements; see Fig. 2.1. We have three edge types that have four elements attached to them and four edge types with six adjacent elements.

The algebraic formulations (2.10) and (2.9) allow us to estimate the effects of the variational crime introduced by the stencil scaling approach.

Lemma 3 *Assumptions (A2) and (A3) hold true.*

Proof 3 *see [1].*

2.4 Guaranteed uniform coercivity

While for our hierarchical hybrid mesh framework the assumptions (A2) and (A3) are satisfied and thus asymptotically, i.e., for h sufficiently small, also (A1) is satisfied, (A1) is not necessarily guaranteed for any given mesh \mathcal{T}_h .

Lemma 4 *If the matrix representation of the discrete Laplace operator is an M-matrix, then the scaled bilinear form $a_h(\cdot, \cdot)$ is positive semi-definite on $V_h \times V_h$ for k globally smooth.*

Proof 4 *see [1];*

Remark 3 *In 2D it is well-known [13] that if all elements of the macro mesh have no obtuse angle, then \hat{A}_h is an M-matrix and we are in the setting of Lemma 4.*

2.4.1 Pre-asymptotic modification in 2D based on (A2)

Here we work out the technical details of a modification in 2D that guarantees uniform ellipticity assuming that at least one macro element T has an obtuse angle. Our modification yields a linear condition on the local mesh-size depending on the discrete gradient of k . It only applies to selected stencil directions. In 2D our 7-point stencil associated with an interior fine grid node has exactly two positive off-center entries if the associated macro element has an obtuse angle. We call the edges associated with a positive reference stencil entry to be of gray type. With each macro element T , we associate the reference stiffness matrix \hat{A}_T . Without loss of generality, we assume that the local enumeration is done in such a way that the largest interior angle of the macro element T is located at the local node 3, i.e., if T has an obtuse angle then $a_{1,2}^T > 0$, $a_{1,3}^T < 0$, and $a_{2,3}^T < 0$ and otherwise $a_{i,j}^T \leq 0$, $1 \leq i < j \leq 3$. By λ_{\min}^T we denote the smallest non-degenerated eigenvalue of the generalized eigenvalue problem

$$\hat{A}_T \mathbf{x} := \begin{pmatrix} a_{1,1}^T & a_{1,2}^T & a_{1,3}^T \\ a_{1,2}^T & a_{2,2}^T & a_{2,3}^T \\ a_{1,3}^T & a_{2,3}^T & a_{3,3}^T \end{pmatrix} \mathbf{x} = \lambda \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^3.$$

We note that both matrices in the eigenvalue problem are symmetric, positive semi-definite and have the same one dimensional kernel and thus $\lambda_{\min}^T > 0$.

Let e be a gray type edge. For each such edge e , we possibly adapt our approach locally. We denote by $\omega_{e;T}$ the element patch of all elements $t \in \mathcal{T}_h$, such that $t \subset T$ and $e \subset \partial t$. Then we define

$$k_{e;\min} := \min_{\tilde{e} \in \mathcal{E}_h^e} k_{\tilde{e}}, \quad k_{\tilde{e}} := \frac{1}{2}(k|_T(x_1^{\tilde{e}}) + k|_T(x_2^{\tilde{e}}))$$

where \mathcal{E}_h^e is the set of all edges being in $\omega_{e;T}$, and $x_1^{\tilde{e}}$ and $x_2^{\tilde{e}}$ are the two endpoints of \tilde{e} . In the pre-asymptotic regime, i.e., if

$$(k_e - k_{e;\min})a_{1,2}^T > k_{e;\min}\lambda_{\min}^T \tag{MA2}$$

we replace the scaling factor $k_e = \frac{1}{2}(k|_T(x_1^e) + k|_T(x_2^e))$ in definition (2.3) by

$$k_e^{\text{mod}} := k_{e;\min} \left(1 + \frac{\lambda_{\min}^T}{a_{1,2}^T} \right).$$

Then it is obvious that $k_{e;\min} < k_e^{\text{mod}} < k_e$. We note that $2a_{1,2}^T$ is the value of the 7-point stencil associated with a gray edge and thus trivial to access.

Lemma 5 *Let the bilinear form be modified according to (MA2), then it is uniformly positive definite on $V_h \cap V_0 \times V_h \cap V_0$ for all simplicial hierarchical meshes.*

Proof 5 *see [1];*

Remark 4 *From the proof it is obvious that any other positive scaling factor less or equal to k_e^{mod} also preserves the uniform ellipticity.*

Remark 5 *We can replace in the modification criterion the local condition (MA2) by*

$$l_T 2^{-(\ell+1)} \|\nabla k\|_{L^\infty(\omega_{e;T})} a_{1,2}^T \geq \inf_{x \in \omega_{e;T}} k(x) \lambda_{\min}^T,$$

where \mathcal{T}_h is obtained by ℓ uniform refinement steps from \mathcal{T}_H , and l_T is the length of the second longest edge in T . Both these criteria allow a local marking of gray type edges which have to be modified. To avoid computation of $k_{e;\min}$ each time it is needed, and thus further reducing computational cost, we set the scaling factor for all gray type edges in a marked T to

$$\frac{1}{2}(k_{\min}(x_1^e) + k_{\min}(x_2^e)), \quad k_{\min}(x_i) := \min \{k(x_j) \mid x_j \in \mathcal{N}_T(x_i)\} \quad (2.11)$$

where $\mathcal{N}_T(x_i)$ denotes the set of all mesh nodes that are connected to node x_i via an edge and belonging to the macro element \bar{T} including x_i itself. The quantity k_{\min} can be pre-computed for each node x_i once at the beginning and stored as a node based vector such as k is. For non-linear problems where k depends on the solution itself, k_{\min} can be updated directly after the update of k .

The presented pre-asymptotic modification based on (A2) yields a condition on the local mesh-size and only affects edge types associated with a positive stencil entry. However, the proof of (A3) shows that a condition on the square of the mesh-size is basically sufficient to guarantee (A1). This observation allows us to design an alternative modification yielding a condition on the square of the local mesh-size and involving the Hessian of k , except for the elements which are in S_h . However, in contrast to the option discussed before, we possibly also have to alter entries which are associated with negative reference stencil entries, and therefore we do not discuss this case in detail. It is obvious that for piecewise smooth k there exists an ℓ_0 such that for all refinement levels $\ell \geq \ell_0$ no local modification has to be applied. This holds true for both types of modifications. Thus, all the a priori estimates also hold true for our modified versions. Since we are interested in piecewise moderate variations of k and large scale computations, i.e., large ℓ , we assume that we are already in the asymptotic regime, i.e., that no modification has to be applied for our 3D numerical test cases, and we do not work out the technical details for the modifications in 3D.

2.4.2 Numerical counter example in the pre-asymptotic regime

In the case that we are outside the setting of Lemma 4, it is easy to come up with an example where the scaled bilinear form is not positive semi-definite, even for a globally smooth k . For a given mesh size h , one can always construct a k with a variation large enough such that the scaled stiffness matrix has negative eigenvalues.

Here we consider a 2D setting on the unit square with an initial mesh which is not Delaunay; see left part of Fig. 2.3. For the coefficient function k , we use a sigmoid function defined as

$$k(x, y; m, \eta) = \frac{\eta}{1 + \exp(-m(y - x - 0.2))} + 1,$$

with $m = 50$ which yields a steep gradient. Further, we vary the magnitude of k by setting $\eta \in \{1, 10, 100, 1000\}$.

Now, we assemble the global stiffness matrix and report in Tab. 2.1 for the different choices of η its minimal and maximal eigenvalue over a sequence of uniform refinement steps. We show the eigenvalues for our scaling approach (2.3) with and without modification (MA2) and for the standard nodal integration assembly (2.7). We find that for $\eta \geq 100$ the stiffness matrix of the scaling approach has negative eigenvalues in the pre-asymptotic regime. In these cases three or four refinement steps are required, respectively, to enter the asymptotic regime.

Table 2.1: Minimal and maximal eigenvalues of the global stiffness matrix A . Italic entries highlight application of (MA2).

scaling approach												
$\eta \setminus \ell$	λ_{\min}^A						λ_{\max}^A					
	0	1	2	3	4	5	0	1	2	3	4	5
1	2.7	0.72	0.18	0.046	0.011	0.0029	9.1	13.3	15.1	17.4	19.8	21.3
10	2.8	0.89	0.24	0.063	0.016	0.0039	40.0	69.3	80.7	93.2	108	116
100	-8.9	-10.6	-5.5	0.066	0.018	0.0045	353	633	739	853	985	1066
1000	-128	-148	-94.5	-3.0	0.019	0.0049	3486	6265	7317	8450	9759	10562
scaling approach modified with (MA2)												
$\eta \setminus \ell$	λ_{\min}^A						λ_{\max}^A					
	0	1	2	3	4	5	0	1	2	3	4	5
1	2.7	0.72	0.18	0.046	0.011	0.0029	9.1	13.3	15.1	17.4	19.8	21.3
10	<i>3.8</i>	<i>1.1</i>	<i>0.26</i>	0.063	0.016	0.0039	<i>40.8</i>	<i>69.4</i>	<i>80.7</i>	93.2	108	116
100	<i>4.0</i>	<i>1.3</i>	<i>0.30</i>	<i>0.072</i>	<i>0.018</i>	0.0045	<i>364</i>	<i>634</i>	<i>739</i>	<i>853</i>	985	1066
1000	<i>4.4</i>	<i>1.3</i>	<i>0.33</i>	<i>0.078</i>	<i>0.019</i>	0.0049	<i>3598</i>	<i>6278</i>	<i>7321</i>	<i>8453</i>	9759	10562
standard nodal integration												
$\eta \setminus \ell$	λ_{\min}^A						λ_{\max}^A					
	0	1	2	3	4	5	0	1	2	3	4	5
1	2.8	0.72	0.18	0.046	0.011	0.0029	9.0	12.8	14.7	17.1	19.7	21.2
10	5.7	1.3	0.27	0.065	0.016	0.0040	36.6	64.9	78.4	90.7	107	116
100	27.7	1.8	0.34	0.076	0.018	0.0045	316	590	717	829	977	1064
1000	247	1.9	0.38	0.084	0.020	0.0049	3116	5840	7104	8211	9682	10549

However, the positive definiteness of the matrix can be recovered on coarser resolutions if (MA2) is applied. Asymptotically, the minimal or maximal eigenvalues of all three approaches tend to the same values.

In Fig. 2.3, we illustrate the action of the modification and show how the region of elements where it has to be applied is getting smaller and finally vanishes with increasing number of refinement steps. To further illustrate how the region that is affected by (MA2) changes, we show a second example where all initial elements have an obtuse angle. The underlying color bar represents the coefficient function k with $\eta = 1000$ for the first (left), and $\eta = 100$ for the second (right) example. The location of the steepest gradient of k is marked by a dashed line. Note that the gradient of k is constant along lines parallel to the dashed one. For all edges that are modified by (MA2) the adjacent elements are shown. The color intensity refers to the refinement level and goes from bright (initial mesh) to dark.

2.4.3 The sign of the stencil entries in 3D

In contrast to the 2D setting, a macro-mesh with no obtuse angle does not yield that \hat{A}_h is an M-matrix. Here the uniform refinement rule yields that for each macro element three sub-classes of tetrahedra (gray, blue, green) exist. To each of these we associate one interior edge type (gray, blue, green) defined by not being parallel to any of the six edges of the respective tetrahedron type. Fig. 2.4 shows the sub-classes and as example the gray edge type. The coloring of the sub-classes is up to now arbitrary. We always associate the gray color with the macro element and call the associated interior edge, a gray type edge. The interior edges associated with the blue and green elements are called blue and green type edges, respectively. All other remaining edges are by notation red type edges. If the macro element T has no obtuse angle between two faces, then it follows from [14, 15] that the reference stencil entries, i.e., the entries associated with the Laplace operator, associated with gray type edges have a positive

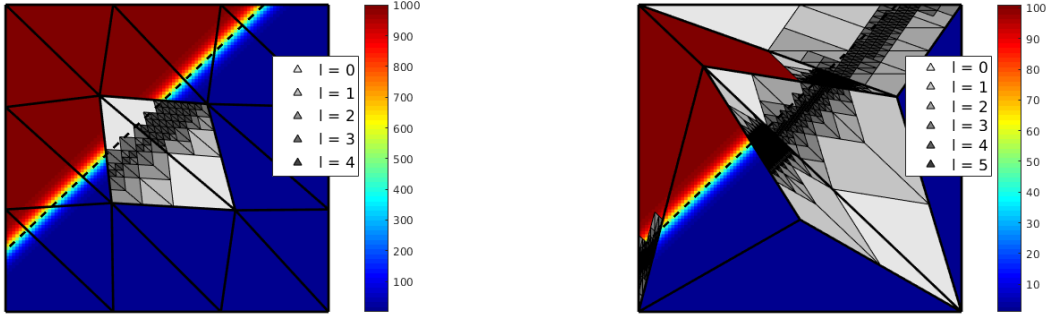


Figure 2.3: Application of (MA2): Mesh with two critical macro elements used for Tab. 2.1 (left) and mesh where all macro elements have an obtuse angle (right).

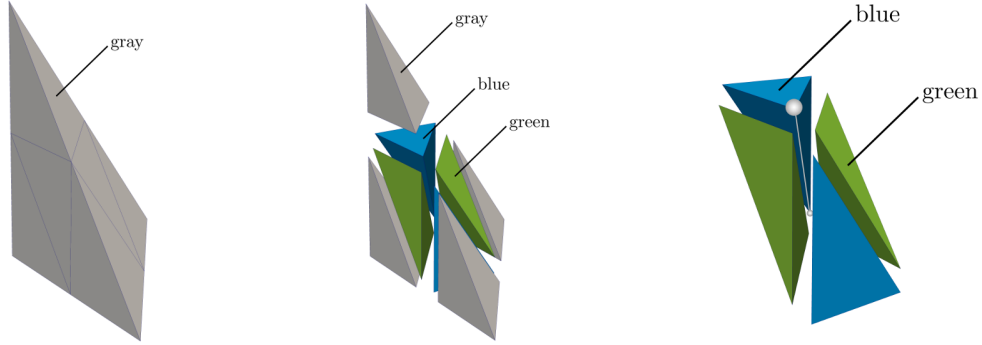


Figure 2.4: Uniform refinement of one macro element T (left) into three subclasses (middle); gray edge between blue and green sub-tetrahedra (right).

sign; see Fig. 2.5. However, it can be shown by some simple geometrical considerations that not both, green and blue type edges, can have a positive sign. If one of these has a positive sign, we call this the blue- and the other one the green-type edge. Conversely, if both have a negative sign, the coloring is arbitrary. In case the macro element T has no obtuse angle, the sign associated with all red type edges is automatically not positive. Thus, we find for such elements in our 15-point stencil either two or four positive off-diagonal entries. Then a modification similar to the one proposed for the 2D case can be now applied to the edges of gray and possibly blue type. The situation can be drastically different if the macro element T has an obtuse angle between two faces. In that case, we find up to four edge directions which carry a positive sign in the stencil.

2.5 Reproduction property and primitive concept

As already mentioned, our goal is to reduce the cost of a WU and the run-times while preserving discretization errors that are qualitatively and quantitatively on the same level as for standard conforming finite elements. Here we focus on the 3D case, but similar results can be obtained for the 2D setting. The a priori bounds of Lemma 1 do not necessarily guarantee that for an affine coefficient function k and affine solution u the error is equal to zero. In the upper bound (2.6) a term of the form $\|\nabla k\|_\infty \|\nabla u\|_0$ remains. A closer look at

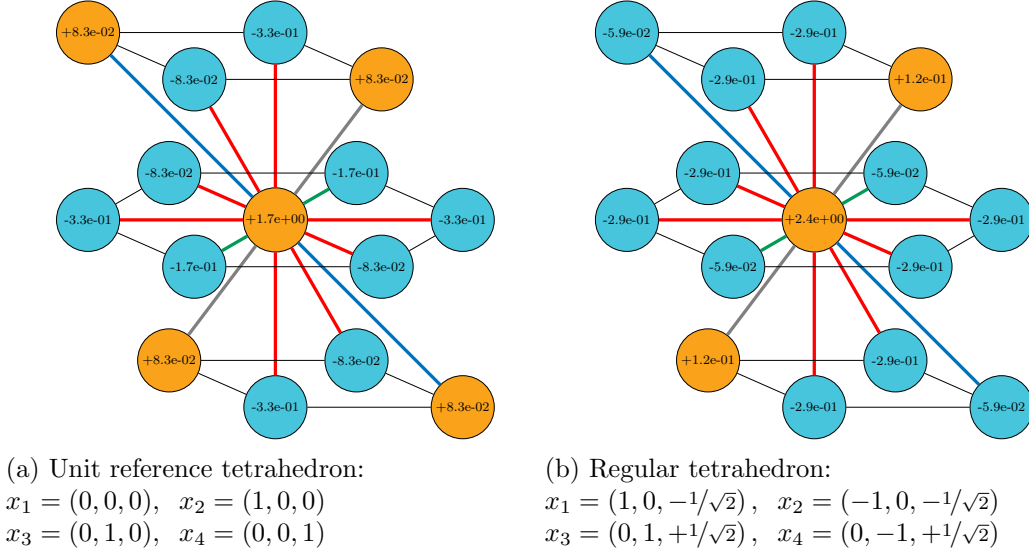


Figure 2.5: Stencil entries colored by their sign at an inner node of two times refined tetrahedra, each without any obtuse angles between faces. On the left (A), the stencil of the unit reference tetrahedron and on the right (B), the stencil of a regular tetrahedron is depicted. The gray edge corresponds to the interior edge through $\frac{1}{2}(x_1 + x_3)$ and $\frac{1}{2}(x_2 + x_4)$. The green edge is the one in direction between x_1 and x_3 and the blue one in direction between x_2 and x_4 . All other edge directions are marked in red.

the proof (see [1]) reveals that this non-trivial contribution can be traced back to the terms associated with nodes on the boundary of a macro element. This observation motivates us to introduce a modification of our stencil scaling approach. As already mentioned, all nodes are grouped into primitives and we have easy access to the elements of these primitives. Recall that $\tilde{a}_h(\cdot, \cdot)$ defined as in (2.7) is associated with the standard finite element approach with nodal quadrature. Let us by \mathcal{W}_V , \mathcal{W}_E , and \mathcal{W}_F denote the set of all nodes associated with the vertex, edge, and face primitives, respectively. Now we introduce a modified stencil scaling approach, cf. (2.3) and (2.7),

$$a_{h;\mathcal{I}}(v_h, \phi_i) := \begin{cases} \tilde{a}_h(v_h, \phi_i), & i \in \mathcal{I} \\ a_h(v_h, \phi_i) & i \notin \mathcal{I} \end{cases}. \quad (2.12)$$

Replacing $a_h(\cdot, \cdot)$ by $a_{h;\mathcal{I}}(\cdot, \cdot)$ with $\mathcal{I} \subset \mathcal{W} := \mathcal{W}_V \cup \mathcal{W}_E \cup \mathcal{W}_F$ still yields that all node stencils associated with a node in a volume primitive are cheap to assemble. The number of node stencils which have to be more expensively assembled grow only at most with 4^ℓ while the total number of nodes grows with 8^ℓ .

Remark 6 *The modification (2.12) introduces an asymmetry in the definition of the stiffness matrix to which multigrid solvers are not sensitive. Moreover, the asymmetry tends asymptotically to zero with $\mathcal{O}(h^2)$.*

Lemma 6 *Let $\mathcal{I} = \mathcal{W}$ or $\mathcal{I} = \mathcal{W}_V \cup \mathcal{W}_E$, then an affine solution can be reproduced if k is affine, i.e., $u_h = u$.*

Proof 6 *see [1];*

Used bilinear form to define FE approximation	Discretization error in discrete L^2 -norm
$\tilde{a}_h(\cdot, \cdot)$ standard FE	3.85104e-15
$a_{h;\mathcal{I}}(\cdot, \cdot)$, $\mathcal{I} = \mathcal{W}$	2.10913e-15
$a_{h;\mathcal{I}}(\cdot, \cdot)$, $\mathcal{I} = \mathcal{W}_V \cup \mathcal{W}_E$	1.75099e-15
$a_{h;\mathcal{I}}(\cdot, \cdot)$, $\mathcal{I} = \mathcal{W}_V \cup \mathcal{W}_F$	6.82596e-04
$a_{h;\mathcal{I}}(\cdot, \cdot)$, $\mathcal{I} = \mathcal{W}_E \cup \mathcal{W}_F$	9.19145e-07
$a_{h;\mathcal{I}}(\cdot, \cdot)$, $\mathcal{I} = \mathcal{W}_V$	6.82032e-04
$a_{h;\mathcal{I}}(\cdot, \cdot)$, $\mathcal{I} = \mathcal{W}_E$	9.17523e-07
$a_h(\cdot, \cdot)$ stencil scaling	6.81942e-04

Table 2.2: Reproduction property

To illustrate Lemma 6, we consider a 3D example on the unit cube $\Omega = (0, 1)^3$. We use as solution $u(x, y, z) = -7x + y + 3z$ and $k(x, y, z) = 2x + 3y + 5z + 1$ for the coefficient function. The standard finite element solution reproduces the exact solution up to machine precision. We test the influence of the stencil scaling approach on different sets of primitives. In Tab. 2.2, we report the discretization error in the discrete L^2 -norm for different combinations. Note that the macro triangulation of the unit cube consists of 12 tetrahedral elements and of one non-boundary macro vertex at $(0.5, 0.5, 0.5)$. These considerations show that the choice $\mathcal{I} = \mathcal{W}_V \cup \mathcal{W}_E$ is quite attractive. The number of stencils which have to be expensively evaluated grows only with 2^ℓ while we still can guarantee the reproduction property for an affine solution in the case of an affine coefficient function.

2.6 Cost of a work unit

The stencil scaling has been introduced as means to reduce the cost of a WU and thus help to design less expensive and thus more efficient PDE solvers. We will first employ a cost metric based on operation count. While we are aware that real run-times will be influenced by many additional factors, including e.g. the quality of the implementation, compiler settings, and various hardware details, the classic measure still provides useful insight. Another important aspect that we will address here is the question of memory accesses. A more technical hardware-aware performance analysis, as conducted in [5] to evaluate the efficiency on real computers, is beyond our current scope. Since asymptotically the contributions from the element primitives dominate the cost, we restrict ourselves to the study of stencils for nodes located in the interior of a macro element T .

2.6.1 Cost for stencil scaling

Let \hat{s}_i^T be the 15-point stencil associated with the Laplacian at an inner node i of T which is independent of the node location within T . Recall that the scaled stencil is given by

$$s_{ij}^T := \frac{1}{2} \left(\kappa(x_i) + \kappa(x_j) \right) \hat{s}_{ij}^T, \quad \forall j \in \mathcal{N}_T(i) \setminus \{i\}, \quad s_{ii}^T := - \sum_{j \in \mathcal{N}_T(i) \setminus \{i\}} s_{ij}^T$$

where $\kappa(x_i)$ represents either the value of the parameter k at node x_i or a modified version of it as suggested in (2.11). Note that this definition is the direct translation of the approach described in section 2.2.1 from the bilinear form to the pure stencil. Assuming that the constant $1/2$ factor is incorporated into the stencil \hat{s}_i^T at setup and taking into account the

number of edges emanating from x_i , being six in 2D and 14 in 3D, we can compute the non-central stencil entries with 12 and 28 operations, respectively. The computation of the central entry via the zero row-sum property takes 5 and 13 additions, respectively. This is summarized in Tab. 2.3.

We next compare the cost for the stencil scaling approach to an on-the-fly computation based on classic FEM techniques which, however, exploits the advantages of hierarchical hybrid grids, see also [16].

Remark 7 *The cost to approximate the stencil coefficients with the two-scale interpolation method that will be introduced in chapter 3 amounts to q operations per stencil entry when the polynomial degree is chosen as q . When the coefficients are locally smooth, $q = 2$ delivers good results. The cost of this method becomes $2 \times 7 = 14$ operations in 2D, and $2 \times 15 = 30$ operations in 3D, respectively. This is slightly cheaper than the stencil scaling of this chapter. However, the stencil scaling approach yields asymptotic optimality for all choices of macro mesh sizes, while the two-scale approach relies on a proper combination of macro scale and fine scale.*

2.6.2 Cost of the on-the-fly computation for the classical FEM

Employing the bilinear form (2.7) we can write the stencil coefficients as

$$\tilde{s}_{ij}^T = \tilde{a}_h(\phi_j, \phi_i) = \sum_{t \in \mathcal{T}_h^{e;T}} \sum_{\ell=1}^{d+1} \frac{k(x_\ell^t)}{(d+1)} \nabla \phi_{j_t} \cdot \nabla \phi_{i_t} |t| = \sum_{t \in \mathcal{T}_h^{e;T}} (E^t)_{i_t, j_t} \sum_{\ell=1}^{d+1} k(x_\ell^t) \quad (2.13)$$

where

$$(E^t)_{i_t, j_t} := \frac{1}{(d+1)} \int_t \nabla \phi_{j_t} \cdot \nabla \phi_{i_t} \, dx$$

denotes the local stiffness matrix including the volume averaging factor $1/(d+1)$. The regularity of the mesh inside a macro element implies that there exist only a fixed number of differently shaped elements t . Thus, we can compute stencil entries on-the-fly from one (2D) or three (3D) pre-computed stiffness matrices and the node-based coefficient values¹. In 2D, always six elements t are attached to an interior node. Summing nodal values of k for each element first would require 12 operations. Together with the fact that two elements are attached to each edge, computing all six non-central stencil entries via (2.13) would then require a total of 30 operations. This number can be reduced by eliminating common sub-expressions. We pre-compute the values

$$k(x_i) + k(j_2) \quad , \quad k(x_i) + k(j_4) \quad , \quad k(x_i) + k(j_6) \quad ,$$

i.e., we sum k for the vertices at the ends of the edges marked as dashed in Fig. 2.6. Then, we obtain $\sum_{\ell=1}^3 k(x_\ell^t)$ by adding the value of k at the third vertex of t to the pre-computed expression. In this fashion, we can compute all six sums with 9 operations and obtain a total of 27 operations for the non-central entries.

The situation in 3D is more complicated. This stems mainly from the fact that the number of elements sharing the edge from node i to node j is no longer a single value as in 2D. Instead

¹Note that in our HHG implementation, we use six element-matrices as this is advantageous with respect to local to global indexing; see [17] for details.

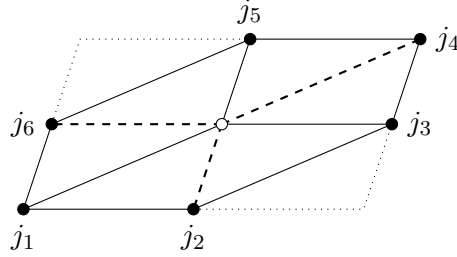


Figure 2.6: Neighborhood of node i and edges [dashed] selected for common sub-expression elimination.

we have in 3D that $|\mathcal{T}_h^{e;T}| = 4$ for the gray, blue, and green type edges that emanate from node i , while $|\mathcal{T}_h^{e;T}| = 6$ for the remaining 8 red type edges. Thus, for the computation of the non-central stencil entries we obtain the cost

$$6(4 + 3) + 8(6 + 5) + 40 = 170 .$$

The summand 40 represents the operations required for summing the nodal k values employing again elimination of common sub-expressions; see [16] for details. Assembling the central entry involves contributions from 6 elements in 2D and 24 in 3D. Thus, executing this via (2.13) cannot require less operations than using the row-sum property.

2.6.3 Cost of stencil application

The cost of the application of the assembled stencil is independent of the approach to compute the stencil. Using the first expression in

$$s_i^T v = \sum_{j \in \mathcal{N}_T(i)} s_{ij}^T v(x_j) = \sum_{j \in \mathcal{N}_T(i) \setminus \{i\}} s_{ij}^T (v(x_j) - v(x_i))$$

results in 13 operations in 2D and 29 in 3D. Exploiting the row-sum property avoids computing the central entry, but increases the cost for the stencil application, cf. second expression, so that in total only a single operation is saved. Tab. 2.3 summarizes the results of this section. We observe that assembling the stencil and applying it once in the scaling approach requires in 2D only about $2/3$ and in 3D about $1/3$ of the operations compared to a classical on-the-fly variant.

approach	dimension	non-central entries	central entry	assembly + application
stencil scaling	2D	6 add / 6 mult	5 add / 0 mult	17 add / 12 mult
	3D	14 add / 14 mult	13 add / 0 mult	41 add / 28 mult
on-the-fly FEM	2D	15 add / 12 mult	5 add / 0 mult	26 add / 18 mult
	3D	98 add / 72 mult	13 add / 0 mult	125 add / 86 mult

Table 2.3: Comparison of operation count for assembling and applying the stencil using either stencil scaling or on-the-fly assembly.

In 3D, the stencil scaling approach results in a total of 69 operations per node, and is thus only 2.4 times more expensive than the constant-coefficient case treated in e.g. [17]. The 69 operations save a factor of 3 compared to the 211 operations per node for the conventional,

yet highly optimized on-the-fly assembly and roughly an order of magnitude when compared with unoptimized variants of on-the-fly-assembly techniques.

To which extent a factor three savings in floating point operations will be reflected in run-time (or other practical cost metrics, such as e.g., energy consumption), depends on many details of the hardware and system software. For instance, we note that in the above algorithms, the addition and multiplication operations are not ideally balanced and they do not always occur such that the fused multiply-add operations of a modern processor architecture can be used. Furthermore, advanced optimizing compilers will restructure the loops and will attempt to find a scheduling of the instruction stream that avoids dependencies. Thus, in effect, the number of operations executed on the processor may not be identical to the number of operations calculated from the abstract algorithm or the high level source code. Nevertheless, though the operation count does not permit a precise prediction of the run-times, we will see in the following that our effort to reduce the number of operations pays off in terms of accelerated execution.

2.6.4 Memory accesses

One other key aspect governing the performance of any algorithm is the number of read and write operations it needs to perform and their pattern with respect to spatial and temporal locality. The influence of these properties results from the large disparity between peak floating point performance of modern CPUs and the latency and bandwidth limitation of memory access. All modern architectures employ a hierarchy of caches [18, 19] that helps accelerate memory access but that also make an a-priori prediction of run-times quite difficult

As in the previous subsections for the number of operations, we will perform here a high-level analysis of memory traffic. Experimental results can be found in [1]. As a baseline we are including in our consideration not only the on-the-fly and the stencil scaling approach, but also a *stored stencil* version. For the latter we assume that in a setup phase the stencil for each node was assembled, by whatever method, and then stored. For the stencil application the weights of the local stencil are then loaded from main memory and applied to the associated degrees of freedom. Since the stencil weights correspond to the non-zero matrix entries of the corresponding row, this would be the HHG analogue of performing a sparse matrix-vector multiplication with a matrix stored in a standard sparse storage format, such as e.g. Compressed Row Storage (CRS). Note, however, that due to the structuredness of the mesh inside a volume primitive, less organizational overhead and indirections are required than for a sparse matrix format. Most importantly we do neither need to store nor transfer over the memory system any information on the position of the non-zero matrix entries.

For this article we do not study the full details of algorithmic optimization for best usage of the memory sub-system. Instead, we are going to compare two idealized cases. These are the *optimistic* version, in which we assume perfect re-use of each data item loaded to the caches and a *pessimistic* version, where there is no re-use at all. Any actual implementation will lie somewhere in between these two cases and the closeness to one of them being determined by algorithmic properties and the quality of its implementation. Furthermore, we will only consider the 3D problem.

Common to all three approaches under consideration is that, in order to apply the local stencil and compute the residuum at a node, they need to load the DoFs at the node and its 14 neighbors and the value of the right-hand side at the node itself. After the stencil application, the resulting nodal value must be written back. Thus, we are not going to inspect these parts

Table 2.4: Cost of the three different approaches measured in total bytes loaded for computation of stencil weights, assuming the use of the IEEE binary64 data-type.

approach	optimistic	pessimistic
stored stencils	$120N$	$120N$
on-the-fly FEM	$768 + 8N$	$768 + 120N$
stencil scaling	$112 + 8N$	$112 + 120N$

of the stencil application and also neglect questions of write-back strategies for the caches.

With respect to the stencil weights the situation is, of course, a different one. Let us start with the *stored stencil* approach. We denote by N the number of DoFs inside a single volume primitive. In 3D a scalar operator using our discretization is represented by a 15-point-stencil. This structure is invariant inside the volume primitive due to the regular mesh structure. Hence, we obtain for the total number of data items to be loaded from memory $\mathcal{N}_{\text{stored}} = 15N$.

In the on-the-fly assembly we start by loading the six pre-computed element matrices, which are in $\mathbb{R}^{4 \times 4}$ for the 3D case. Note that these are loaded only once when the first node of the volume primitive is treated and can stay in the L1 cache during the complete loop over the volume primitive as they only occupy 768 bytes. Assembling the local stencil from these matrices requires information on the coefficient function k . Assuming perfect re-use of the nodal k information, i.e. each nodal value needs to be loaded only once, we obtain $\mathcal{N}_{\text{fly}}^{\text{opt}} = 6 \times (4 \times 4) + N = 96 + N$. In the pessimistic case, where we assume absolutely no cache effects, we need to reload neighboring k values each time we update another node. This gives us $15N$ load operations per node resulting in $\mathcal{N}_{\text{fly}}^{\text{pes}} = 96 + 15N$.

Finally we consider the stencil scaling approach. Here it is sufficient to load once the 14 non-central weights of the reference stencil \hat{s} for the volume primitive². These are 112 bytes and the values can, as in the on-the-fly approach, remain in the L1 cache. The non-central weights are then scaled depending on the neighboring k values, while its central weight is derived using the zero-sum property. This gives us $\mathcal{N}_{\text{scale}}^{\text{opt}} = 14 + N$ and $\mathcal{N}_{\text{scale}}^{\text{pes}} = 14 + 15N$.

Table 2.4 sums up our results. Comparing the entries for the stored stencil approach to the pessimistic bounds for the other two approaches we find the same pre-factors for N plus small constant terms. One should note, however, that in the case of the stored stencils approach there will be no temporal cache effects when we proceed from one node to the next, since the stencil values/row entries cannot be re-used. In the two other approaches, we expect to see positive cache effects due to temporal and spatial re-use of some values of the coefficient function k , i.e. results closer to \mathcal{N}^{opt} . This can be seen in the comparisons in [1].

2.7 Numerical accuracy study and run-time comparison

In this section, we provide different numerical results which illustrate the accuracy and run-time of the new scaling approach in comparison to the element-wise finite element assembling based on nodal integration within a matrix-free framework. We consider different cases such as scalar and tensorial coefficient functions k and the scenario of a geometry mapping. Through-

²The current HHG implementation for technical reasons assembles the stencil from the pre-computed element matrices also in this case, so the constant term in the memory access is the same 768 bytes as for the on-the-fly approach.

out this section, we denote the time-to-solution by `tts` and by relative `tts` always mean the ratio of the time-to-solution of the stencil scaling approach with respect to the nodal integration. From our theoretical considerations for one stencil application from Tab. 2.3, we expect a relative `tts` of roughly one third. Further, we denote the estimated order of convergence by `eoc` and the asymptotic convergence rates of the multigrid solver by ρ defined as $\rho = (r^{(i^*)}/r^{(5)})^{1/i^*-5}$ where $r^{(i)}$ is the L^2 residual at iteration i , and i^* the final iteration of the solver. Each of the following 3D computations was conducted on SuperMUC Phase 2 using the Intel 17.0 compiler together with the Intel 2017 MPI library. For all runs, we specify the compiler flags `-O3 -march=native -xHost`. Note that the serial runs using only a single compute core are not limited to run on large machines like SuperMUC but can also be run on usual modern desktop workstations with enough memory. The peak memory usage by our largest serial run was at about 4.46 GiB.

2.7.1 A quantitative comparison in 2D for a scalar permeability

In this example, we consider as domain the unit-square $\Omega = (0, 1)^2$ and use a non-polynomial manufactured solution

$$u(x, y) = \frac{x^4 y}{xy + 1} .$$

We employ as coefficient function $k(x, y; m) = 2 + \sin(m\pi x) \sin(m\pi y)$ with $m \in \{2, 4, 8\}$. The right-hand side is computed by inserting the above definitions into the equation. This construction has the advantage that we can study the effect of the magnitude of ∇k in a systematic fashion by adjusting m . We perform a study on a regular triangular mesh comparing errors of the discrete solutions obtained by two standard Galerkin finite element approaches and our proposed scaling approach. The error norms are approximated using a 5th order quadrature rule, while we use a 2nd order scheme to evaluate the weak right-hand side. Results are listed in Tab. 2.5. Here, u_h^n denotes the approximation obtained by employing a nodal quadrature rule resulting in bilinear form (2.7), while u_h^b uses a quadrature rule that evaluates k at the triangle's barycenter.

2.7.2 A quantitative comparison in 3D for a scalar permeability

As a second test, we consider a non-linear solution similar to the 2D tests in the previous section

$$u = \frac{x^3 y + z^2}{xyz + 1} ,$$

with a parameter dependent scalar coefficient function $k(x, y, z; m) = \cos(m\pi xyz) + 2$ on the unit-cube $\Omega = (0, 1)^3$ discretized by six tetrahedra. Here, and in the second 3D example below, the weak right-hand side is computed by interpolating the right-hand side associated with our manufactured solution into our finite element ansatz space and subsequent multiplication with the mass matrix. The L^2 error is approximated by a discrete version, i.e. an appropriately scaled nodal L^2 error. While more advanced approaches could be used here, the ones chosen are completely sufficient for our purpose, which is to demonstrate that our stencil scaling approach behaves analogously to a classical FE approach. We employ a multigrid solver with a V(3,3) cycle on a single compute core and stop after 10 multigrid iterations, i.e., $i^* = 10$, which is enough to reach the asymptotic regime. In Tab. 2.6, we report convergence of the discretization error for the three approaches, i.e., classical FE with nodal integration, stencil

scaling on volumes and faces, but classical FE assembly on edges and vertices, and stencil scaling on all primitives. The refinement is given by L where $L = -2$ denotes the macro mesh. We observe quadratic convergence of the discrete L^2 error for all three approaches and a relative tts of about 32% on level $L = 6$.

2.7.3 A quantitative comparison in 3D for a permeability tensor

As a third test, we consider a full symmetric and positive definite permeability tensor K with non-linear components. The off-diagonal components are negative or zero:

$$K = \begin{pmatrix} x^2 + 2y^2 + 3z^2 + 1 & -y^2 & -z^2 \\ -y^2 & 2x^2 + 3y^2 + z^2 + 1 & -x^2 \\ -z^2 & -x^2 & 3x^2 + y^2 + 2z^2 + 1 \end{pmatrix}$$

The manufactured solution is set to

$$u = \frac{x^4 y + 2z}{xyz + 1},$$

and we consider as domain the unit-cube discretized by twelve tetrahedra. We employ the same multigrid solver as in the previous subsection on a single compute core, but stop the iterations if the residual is reduced by a factor of 10^{-9} . We denote the final iteration by i^* . The results for the classical FE approach and the stencil scaling approach on volumes and faces with classical FE assembly on edges and vertices are reported in Tab. 2.7. We observe quadratic convergence of the discrete L^2 error for both approaches and a relative tts of 31% on our finest level $L = 6$. Note that, as in Tab. 2.6, the relative tts exhibits a slightly non-monotonic behaviour. This stems from the fact that different types of primitives (e.g. faces and volume) have a different tts behaviour for increasing L and profit differently from the

L	midpoint integration				nodal integration				stencil-approach			
	$\ u - u_h^b\ _0$	eoc	$ u - u_h^b _1$	eoc	$\ u - u_h^n\ _0$	eoc	$ u - u_h^n _1$	eoc	$\ u - u_h^s\ _0$	eoc	$ u - u_h^s _1$	eoc
$m = 2$												
0	3.75e-03	0.00	9.13e-02	0.00	3.25e-03	0.00	9.15e-02	0.00	3.34e-03	0.00	9.14e-02	0.00
1	9.61e-04	1.96	4.60e-02	0.99	8.59e-04	1.92	4.61e-02	0.99	8.65e-04	1.95	4.61e-02	0.99
2	2.42e-04	1.99	2.31e-02	1.00	2.18e-04	1.97	2.31e-02	1.00	2.18e-04	1.98	2.31e-02	1.00
3	6.06e-05	2.00	1.15e-02	1.00	5.50e-05	1.99	1.15e-02	1.00	5.48e-05	2.00	1.15e-02	1.00
4	1.51e-05	2.00	5.77e-03	1.00	1.37e-05	2.00	5.78e-03	1.00	1.37e-05	2.00	5.78e-03	1.00
$m = 4$												
0	3.61e-03	0.00	9.25e-02	0.00	3.64e-03	0.00	9.32e-02	0.00	3.54e-03	0.00	9.36e-02	0.00
1	9.14e-04	1.98	4.62e-02	1.00	1.02e-03	1.83	4.70e-02	0.99	9.40e-04	1.91	4.67e-02	1.00
2	2.30e-04	1.99	2.31e-02	1.00	2.81e-04	1.87	2.33e-02	1.02	2.43e-04	1.95	2.31e-02	1.01
3	5.77e-05	2.00	1.15e-02	1.00	7.27e-05	1.95	1.15e-02	1.01	6.13e-05	1.98	1.15e-02	1.00
4	1.44e-05	2.00	5.78e-03	1.00	1.83e-05	1.99	5.78e-03	1.00	1.53e-05	2.00	5.78e-03	1.00
$m = 8$												
0	4.06e-03	0.00	1.01e-01	0.00	4.71e-03	0.00	1.07e-01	0.00	4.47e-03	0.00	1.03e-01	0.00
1	9.16e-04	2.15	4.87e-02	1.06	1.14e-03	2.04	5.09e-02	1.08	1.09e-03	2.04	5.23e-02	0.98
2	2.32e-04	1.98	2.35e-02	1.05	4.02e-04	1.51	2.55e-02	1.00	3.29e-04	1.73	2.44e-02	1.10
3	5.97e-05	1.96	1.16e-02	1.02	1.21e-04	1.73	1.20e-02	1.08	8.80e-05	1.90	1.17e-02	1.06
4	1.50e-05	1.99	5.78e-03	1.01	3.20e-05	1.92	5.84e-03	1.04	2.24e-05	1.97	5.80e-03	1.02

Table 2.5: Results for a regular triangular mesh. Here, L denotes the (uniform) refinement level and eoc the estimated order of convergence.

		nodal integration			scale Vol+Face				scale all			
L	DoF	error	eoc	ρ	error	eoc	ρ	rel. tts	error	eoc	ρ	rel. tts
$m = 3$												
1	3.43e+02	2.46e-03	–	0.07	2.42e-03	–	0.06	1.13	2.51e-03	–	0.07	0.70
2	3.38e+03	7.06e-04	1.80	0.13	5.97e-04	2.02	0.12	0.45	6.05e-04	2.05	0.12	0.45
3	2.98e+04	1.80e-04	1.97	0.16	1.46e-04	2.03	0.15	0.34	1.47e-04	2.05	0.15	0.28
4	2.50e+05	4.46e-05	2.01	0.18	3.59e-05	2.02	0.15	0.30	3.59e-05	2.03	0.15	0.29
5	2.05e+06	1.11e-05	2.01	0.17	8.88e-06	2.01	0.14	0.31	8.88e-06	2.02	0.14	0.32
6	1.66e+07	2.75e-06	2.01	0.16	2.21e-06	2.01	0.13	0.32	2.21e-06	2.01	0.13	0.32
$m = 8$												
1	3.43e+02	3.17e-03	–	0.07	5.43e-03	–	0.09	1.33	5.21e-03	–	0.08	0.70
2	3.38e+03	1.50e-03	1.08	0.15	1.54e-03	1.82	0.15	0.53	1.56e-03	1.74	0.15	0.45
3	2.98e+04	4.83e-04	1.63	0.19	4.04e-04	1.93	0.17	0.31	4.06e-04	1.94	0.16	0.36
4	2.50e+05	1.31e-04	1.89	0.19	1.01e-04	1.99	0.16	0.30	1.02e-04	2.00	0.17	0.29
5	2.05e+06	3.32e-05	1.98	0.20	2.52e-05	2.01	0.16	0.31	2.53e-05	2.01	0.17	0.32
6	1.66e+07	8.30e-06	2.00	0.21	6.29e-06	2.01	0.16	0.33	6.29e-06	2.01	0.16	0.32

Table 2.6: 3D results in the case of a scalar coefficient function with errors measured in the discrete L^2 -norm.

scaling approach. For large L , tts is dominated by the work performed on the volume DoFs as is the relative tts between two approaches. A test with a single macro tetrahedron showed a monotonic behaviour for the relative tts.

2.7.4 Application to a blending setting and large scale results

To demonstrate the advantages of our novel scaling approach also for a more realistic scenario, we consider an example using a blending function, as mentioned in section 2.2. To this end, we consider a half cylinder mantle with inner radius $r_1 = 0.8$ and outer radius $r_2 = 1.0$, height $z_1 = 4.0$ and with an angular coordinate between 0 and π as our physical domain Ω_{phy} . The cylinder mantle is additionally warped inwards by $w(z) = 0.2 \sin(z\pi/z_1)$ in axial direction. The mapping $\Phi : \Omega_{\text{phy}} \rightarrow \Omega$ is given by

$$\Phi(x, y, z) = \begin{pmatrix} \sqrt{x^2 + y^2} + w(z) \\ \arccos(x/\sqrt{x^2 + y^2}) \\ z \end{pmatrix}$$

		nodal integration			scale Vol+Face			rel.
L	DoF	error	eoc	ρ	error	eoc	ρ	tts
1	8.55e+02	1.92e-03	–	0.07	2.21e-03	–	0.07	0.60
2	7.47e+03	4.40e-04	2.13	0.16	5.19e-04	2.09	0.16	0.35
3	6.26e+04	1.06e-04	2.05	0.24	1.27e-04	2.03	0.24	0.25
4	5.12e+05	2.60e-05	2.02	0.30	3.15e-05	2.01	0.30	0.26
5	4.15e+06	6.47e-06	2.01	0.35	7.86e-06	2.00	0.35	0.30
6	3.34e+07	1.61e-06	2.00	0.37	1.96e-06	2.00	0.37	0.31

Table 2.7: 3D results in the case of a tensorial permeability with discretization errors measured in the discrete L^2 -norm.

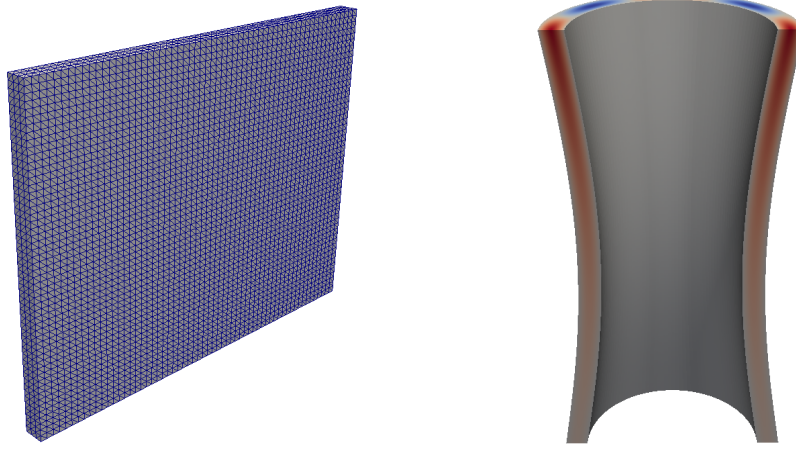


Figure 2.7: Macro mesh of the reference domain Ω (left) and analytic solution u (right) mapped to the physical domain Ω_{phy} .

L	DoF	nodal integration				scale Vol+Face				rel.
		error	eoc	ρ	tts	error	eoc	ρ	tts	tts
1	4.7e+06	2.43e-04	-	0.522	2.5	2.38e-04	-	0.522	2.0	0.80
2	3.8e+07	6.00e-05	2.02	0.536	4.2	5.86e-05	2.02	0.536	2.6	0.61
3	3.1e+08	1.49e-05	2.01	0.539	12.0	1.46e-05	2.01	0.539	4.5	0.37
4	2.5e+09	3.72e-06	2.00	0.538	53.9	3.63e-06	2.00	0.538	15.3	0.28
5	2.0e+10	9.28e-07	2.00	0.536	307.2	9.06e-07	2.00	0.536	88.9	0.29
6	1.6e+11	2.32e-07	2.00	0.534	1822.2	2.26e-07	2.00	0.534	589.6	0.32

Table 2.8: Results for large scale 3D application with errors measured in the discrete L^2 -norm.

with the reference domain $\Omega = (r_1, r_2) \times (0, \pi) \times (0, z_1)$. Using (2.1), it follows for the mapping tensor K

$$K = \frac{(D\Phi)(D\Phi)^\top}{|\det D\Phi|} = \sqrt{x^2 + y^2} \begin{pmatrix} w'(z)^2 + 1 & 0 & w'(z) \\ 0 & 1/x^2 + y^2 & 0 \\ w'(z) & 0 & 1 \end{pmatrix}.$$

Obviously, this tensor is symmetric and positive definite. In addition to the geometry blending, we use a variable material parameter $a(x, y, z) = 1 + z$. On the reference domain Ω this yields the PDE $-\operatorname{div} aK\nabla u = f$. As analytic solution on the reference domain we set

$$u(\hat{x}, \hat{y}, \hat{z}) = \sin\left(\frac{\hat{x} - r_1}{r_2 - r_1}\pi\right) \cos(4\hat{y}) \exp(\hat{z}/2).$$

The analytic solution mapped to the physical domain is illustrated in the right part of Fig. 2.7.

For our numerical experiments, we employ a macro mesh composed of 9540 hexahedral blocks, where each block is further split into six tetrahedral elements; see Fig. 2.7 (left). The resulting system is solved using 14 310 compute cores, i.e., we assign four macro elements per core. For the largest run we have a system with $\mathcal{O}(10^{11})$ DoF. We employ a multigrid solver with a V(3,3) cycle. The iteration is stopped when the residual has been reduced by a factor of 10^{-8} . In Tab. 2.8, we report the resulting discretization error, the asymptotic multigrid convergence order ρ , and the time-to-solution.

These results demonstrate that the new scaling approach maintains the discretization error, as is expected on structured grids from our variational crime analysis, as well as the multigrid convergence rate. For small L we observe that the influence of vertex and edge primitives is more pronounced as the improvement in time-to-solution is only small. But for increasing L this influence decreases and for $L \geq 4$ the run-time as compared to the nodal integration approach is reduced to about 30%.

2.8 References

- [1] S. Bauer, D. Drzisga, M. Mohr, U. Rde, C. Waluga, and B. Wohlmuth, “A stencil scaling approach for accelerating matrix-free finite element implementations,” *SIAM Journal on Scientific Computing*, vol. 40, no. 6, pp. C748–C778, 2018.
- [2] P. G. Ciarlet, *The finite element method for elliptic problems. Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1978.
- [3] G. Strang and G. Fix, *An analysis of the finite element method*. Wellesley-Cambridge Press, Wellesley, MA, second ed., 2008.
- [4] A. Brandt, *Barriers to achieving textbook multigrid efficiency (TME) in CFD*. Institute for Computer Applications in Science and Engineering, NASA saidLangley Research Center, 1998.
- [5] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, “Towards textbook efficiency for parallel multigrid,” *Numer. Math. Theory Methods Appl.*, vol. 8, 2015.
- [6] S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann, and B. Wohlmuth, “A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes,” *Applied Numerical Mathematics*, vol. 122, pp. 14 – 38, 2017.
- [7] W. J. Gordon and C. A. Hall, “Transfinite element methods: Blending-function interpolation over arbitrary curved element domains,” *Numer. Math.*, vol. 21, pp. 109–129, 1973.
- [8] J. Bey, “Tetrahedral grid refinement,” *Computing*, vol. 55, no. 4, pp. 355–378, 1995.
- [9] B. Bergen, G. Wellein, F. Hülsemann, and U. Rüde, “Hierarchical hybrid grids: achieving teraflop performance on large scale finite element simulations,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 22, no. 4, pp. 311–329, 2007.
- [10] W.-S. Yang, *Variable viscosity thermal convection at infinite Prandtl number in a thick spherical shell*. PhD thesis, University of Illinois at Urbana-Champaign, 1997.
- [11] J. Lyness and U. Rüde, “Cubature of integrands containing derivatives,” *Numerische Mathematik*, vol. 78, no. 3, pp. 439–461, 1998.
- [12] C. Pechstein and R. Scheichl, “Weighted Poincaré inequalities,” *IMA Journal of Numerical Analysis*, vol. 33, no. 2, p. 652, 2013.
- [13] P. G. Ciarlet and P.-A. Raviart, “Maximum principle and uniform convergence for the finite element method,” *Computer Methods in Applied Mechanics and Engineering*, vol. 2, no. 1, pp. 17–31, 1973.
- [14] M. Křížek and L. Qun, “On diagonal dominance of stiffness matrices in 3d,” *East–West J. Numer. Math.*, vol. 3, pp. 59–69, 1995.

- [15] S. Korotov, M. Křížek, and P. Neittaanmäki, “Weakened acute type condition for tetrahedral triangulations and the discrete maximum principle,” *Math. Comp.*, vol. 70, no. 233, pp. 107–119, 2001.
- [16] B. Gmeiner, *Design and Analysis of Hierarchical Hybrid Multigrid Methods for Peta-Scale Systems and Beyond*. PhD thesis, Technische Fakultät der Friedrich-Alexander-Universität Erlangen-Nürnberg, 2013.
- [17] B. Bergen, *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*. PhD thesis, Technische Fakultät der Friedrich-Alexander-Universität Erlangen-Nürnberg, 2006. SCS Publishing House.
- [18] D. E. Comer, *Essentials of Computer Architecture*. Pearson Prentice Hall, New Jersey, 2005.
- [19] G. Hager and G. Wellein, *Introduction to High Performance Computing for Scientists and Engineers*. Computational Science Series, CRC Press, 2011.

Chapter 3

A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes

This chapter was published in the Journal *Applied Numerical Mathematics* in 2017.

A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes, S. Bauer, M. Mohr, U. Rüde, J. Weismüller, M. Wittmann and B. Wohlmuth.

<http://dx.doi.org/10.1016/j.apnum.2017.07.006>

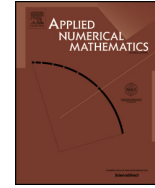
It presents a novel approach for an efficient on-the-fly FE stencil assembly technique on curved domains. The approach is based on a low-order polynomial approximation of the exact FE stencils. This method is perfectly suited for the spherical shell domain used in mantle convection simulations and yields significant performance improvements of up to $50\times$ speed-up compared to a traditional FE stencil assembly scheme.



Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum



A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes



S. Bauer^{a,*}, M. Mohr^a, U. Rude^b, J. Weismüller^{a,e}, M. Wittmann^c,
B. Wohlmuth^d

^a Dept. of Earth and Environmental Sciences, Ludwig-Maximilians-Universität München, Theresienstraße 41, D-80333 München, Germany

^b Dept. of Computer Science 10, FAU Erlangen-Nürnberg, Cauerstraße 6, D-91058 Erlangen, Germany

^c Erlangen Regional Computing Centre (RRZE), FAU Erlangen-Nürnberg, Martensstraße 1, D-91058 Erlangen, Germany

^d Institute for Numerical Mathematics (M2), Technische Universität München, Boltzmannstrasse 3, D-85748 Garching b. München, Germany

^e Leibniz Supercomputing Centre (LRZ), Boltzmannstraße 1, 85748 Garching b. München, Germany

ARTICLE INFO

Article history:

Received 23 September 2016

Received in revised form 12 July 2017

Accepted 18 July 2017

Available online 26 July 2017

Keywords:

Two-scale PDE discretization
Massively parallel multigrid
Matrix free on-the-fly assembly
ECM-model
Large scale scaling results
Surrogate operator

ABSTRACT

Large scale matrix-free finite element implementations save memory and are often significantly faster than implementations using classical sparse matrix techniques. They are especially well suited for massively parallel geometric multigrid solvers in combination with hierarchical hybrid grids on polyhedral domains. In the case of constant coefficients, the number of different stencil entries depends only on the coarse grid and does not increase with the number of refinement levels. However, for non-polyhedral domains the situation changes. Then even for the Laplace operator, the element mapping leads to fine grid stencils that can vary from grid point to grid point. Traditional matrix-free techniques that are based on an element-wise assembly then result in a considerably increase in computational cost. To compensate for this shortcoming, we introduce a new two-scale approach that uses a surrogate operator. It exploits a piecewise polynomial approximation of the entries of the stencil of the fine grid operator with respect to the coarse mesh size. The low-cost evaluation of these surrogate polynomials results in an efficient stencil assembly on-the-fly for non-polyhedral domains. We discuss and illustrate numerically two-scale a priori bounds. The accuracy of the approximate solution can be further improved if combined with a double discretization technique. A careful performance analysis in combination with a hardware-aware code optimization based on the Execution-Cache-Memory model yields a significant speed up. Weak and strong scaling results illustrate the potential of this new two-scale approach within large scale PDE simulations.

© 2017 IMACS. Published by Elsevier B.V. All rights reserved.

* Corresponding author.

E-mail address: simon.bauer@geophysik.uni-muenchen.de (S. Bauer).

1. Introduction

Classical low order finite elements for the numerical approximation of partial differential equations are well-established for highly efficient and massively parallel implementations. However large scale applications require a careful, memory aware realization.

While developing more efficient sparse matrix techniques is an active field of research, see e.g. [27,20,15], pre-computing and storing the entries of the element matrix inevitably requires an order of magnitude more memory than just storing the solution and right hand data vectors. Moreover the frequent access to the matrix coefficients produces massive memory traffic in the solution phase of the algebraic system, [11]. Transferring large amounts of data from memory to the CPU is considered to be a critical cost factor, e.g. in terms of energy consumption. Memory caching as well as spatial and temporal blocking techniques can only partly alleviate this problem [16,26]. Although the consequences of these simple considerations can be most easily grasped for extreme scale computing on super-computers, they also affect the feasibility of moderate to large scale simulations on mid-size clusters.

Let us consider an extreme scale example from geophysics. The global resolution of the earth's mantle with 1 km already leads to a mesh with 10^{12} nodes. Then the solution vector for a scalar PDE operator requires 8 TByte of memory. Such a large vector still fits well in the aggregate memory of current peta-scale supercomputers. This becomes different, when we consider storing the system matrix. Then, even in classical compressed row storage (CRS) format, see e.g. [4], such a matrix with its on average 15 non-zeros per row will require 240 TByte of storage. For most super-computers of today this is beyond their limits.

To avoid the bottleneck of storing huge matrices, so-called matrix-free approaches [33,9,1,28,30] must be employed when possible. Here either all matrix entries are re-computed on-the-fly when needed, or matrix-vector-products are evaluated on-the-fly using efficient quadrature approaches for the local contributions. On polyhedral domains, simple on-the-fly assembly routines can be developed that make use of the similarity of the finite elements in the refinement hierarchy. However, the situation is more complex for non-polyhedral domains. A naïve approach which ignores the physically correct domain and resolves the geometry only with respect to a fixed coarser mesh will result in a reduced accuracy and asymptotically in a loss of convergence.

Resolving the geometry by nodal low order interpolation and using affine element mappings, gives optimal order results for linear finite elements. However, resolving the geometry more and more accurately with increasing refinement level is computationally expensive, since now the refinement process generates new non-similar elements and thus the local assembly process cannot benefit to the full extent from the uniform refinement structure. These fundamental observations are the reason why current matrix-free codes reach significantly better performance on polyhedral domains than in more general geometries. In many real-life applications, e.g., from structural mechanics, fluid dynamics, or geophysics, curved boundaries of the computational domain have to be taken into account see e.g. [14,7,25,23]. We refer to [2,3,5,17,31,35] for state of the art massively parallel solvers and large scale applications.

The main contribution of this article is a novel technique that improves the performance of matrix-free techniques in the case of complex geometries and curved boundaries without sacrificing accuracy. To achieve this, we will propose and analyze a new two-scale combination of polynomial approximation with finite element discretization.

The outline of the paper is as follows: We start with the formulation of the model problem, posed on a thick spherical shell, and a brief discussion of hybrid hierarchical grids in Sec. 2. In Sec. 3 we introduce the new two-scale method based on a higher order polynomial approximation of the low order finite element stencil. We discuss two different variants using standard nodal interpolation and least square fitting techniques and comment on a priori estimates. In Sec. 4 we focus on a study of the numerical accuracy and cost considerations. The influence of the polynomial degree in the stencil approximation on the accuracy as well as the effect of a double discretization [12, Section 10.2] used for residual computation and smoother application is illustrated. Sec. 5 is devoted to the kernel optimization as it is essential for large scale high performance computations. The analysis is based on an enhanced Execution-Cache-Memory (ECM) model. In Sec. 6 we present weak and strong scaling results on a current peta-scale architecture. Finally, in Sec. 7 we extend our model problem to two alternative geometries and illustrate by this the flexibility of the approach.

2. Model problem and hybrid hierarchical meshes

Matrix-free implementations of finite element schemes for partial differential equations are attractive on modern heterogeneous architectures. However, special care is required in the case of PDEs on domains with curvilinear boundary surfaces or interfaces. Here we propose a novel two-scale scheme that is perfectly suited to large scale matrix-free geometric multi-grid solvers for low order conforming finite element discretizations.

2.1. Model problem

We consider the Laplace equation with homogeneous Dirichlet boundary conditions on a spherical shell as our model problem

$$-\Delta u = f \quad \text{in } \Omega \quad \text{and} \quad u = 0 \quad \text{on } \partial\Omega \quad (1)$$

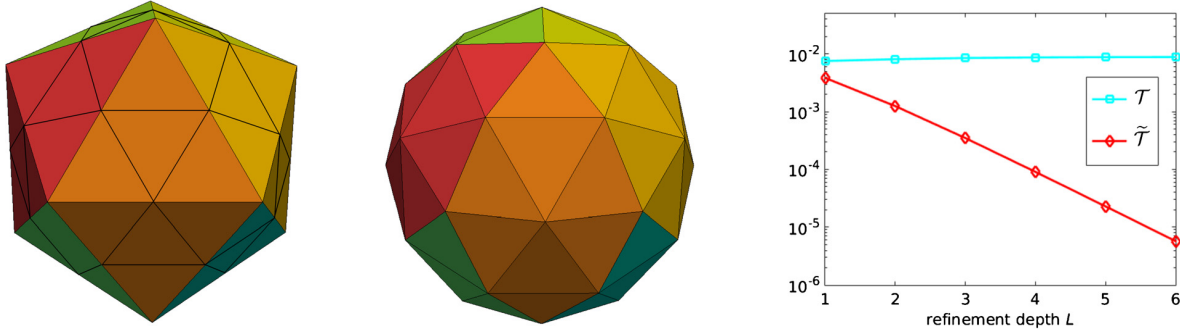


Fig. 2.1. Approximation of a spherical shell with \mathcal{T}_{-1} (left) and $\tilde{\mathcal{T}}_{-1}$ (center). In the right plot the discretization error of the numerical experiments carried out in Sec. 4 is shown over a sequence of refinement levels L . While $\tilde{\mathcal{T}}_\ell$ is linked to Ω_ℓ which approximates Ω with higher accuracy the larger ℓ is, \mathcal{T}_ℓ is associated with a fixed Ω_H independent of ℓ . Thus, the discretization error is almost independent of ℓ and completely dominated by the domain approximation error of Ω_H .

where $\Omega := \{(x, y, z) \in \mathbb{R}^3 : r_1^2 < x^2 + y^2 + z^2 < r_2^2\}$ with $0 < r_1 < r_2$ and $f \in L^2(\Omega)$. But we point out that our approach can be also applied for more complex systems of PDEs, variable coefficients PDEs and more general geometries.

The standard Galerkin formulation of (1) reads as: find $u \in H_0^1(\Omega)$ which satisfies

$$a(u, v) = (f, v) \quad \forall v \in H_0^1(\Omega) \tag{2}$$

where the bilinear and linear forms are given by

$$a(u, v) := \langle \nabla u, \nabla v \rangle_\Omega, \quad (f, v) := (f, v)_\Omega,$$

for all $u, v \in H_0^1(\Omega)$.

2.2. Domain and finite element approximations

If a polyhedral domain is considered, the input mesh can be selected such that the domain is equal to the union of all elements of the mesh. Thus, the domain can be represented exactly. However, for our model domain of a spherical shell this is not possible when using tetrahedra that are affinely mapped from the reference tetrahedron \hat{T} with nodes $(0, 0, 0)$, $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$. We use two different sequences of triangulations. Both of them are based on a possibly unstructured simplicial initial mesh \mathcal{T}_{-2} of mesh size H defining $\overline{\Omega}_H := \cup_{T \in \mathcal{T}_{-2}} \overline{T}$. Each element in \mathcal{T}_{-2} is obtained by the affine mapping Ψ_T from \hat{T} , and the elements T of \mathcal{T}_{-2} are named *macro elements*.

Following the successive regular refinement rules in [10], a hierarchy of uniformly refined triangulations $\mathcal{T} := \{\mathcal{T}_\ell, \ell = 0, 1, \dots, L\}$ is found. We point out that our notation guarantees that for every mesh in the sequence \mathcal{T} each macro element has at least one inner node, i.e., a node not lying on its boundary. The level mesh size is given by $h_\ell = 2^{-(\ell+2)}H$ and for each mesh $\mathcal{T}_\ell \in \mathcal{T}$, we have $\overline{\Omega}_H = \cup_{T \in \mathcal{T}_\ell} \overline{T}$. Consequently, the quality of the geometry approximation is fixed and does not improve with increasing the refinement level. This can e.g. be seen in Fig. 2.1.

Thus, we need a second sequence of meshes $\tilde{\mathcal{T}}$ which is in the limit able to resolve Ω . Such a sequence can be constructed with the help of a global mapping Φ which is assumed to be globally continuous and satisfies $\Phi(\Omega_H) = \Omega$. This global mapping is defined element-wise on each macro element T by $\Phi|_T := \Phi_T$ with a blending function Φ_T reflecting the domain of interest, i.e., $\overline{\Omega} = \cup_{T \in \mathcal{T}_{-2}} \overline{\Phi_T(T)}$. (See Fig. 2.2.) In case of a polyhedral domain Φ_T can be selected as identity and in more general, but piecewise smooth, settings we have $\|(\Phi_T - \text{Id}) \circ \Psi_T\|_\infty = \mathcal{O}(H^2)$ characterizing the difference in the local domain approximation. We refer to [18] for an explicit proof in 2D, if Φ_T is defined by a smooth surface representation.

The meshes $\tilde{\mathcal{T}}_\ell$ of the new sequence $\tilde{\mathcal{T}}$ are now given as the simplicial triangulation of the vertices of \mathcal{T}_ℓ after mapping by Φ . The edge graphs of $\tilde{\mathcal{T}}$ are topologically equivalent to the ones of \mathcal{T} , i.e., the node connectivity remains unchanged. Under the assumption that Φ is a piecewise smooth bijection, $\tilde{\mathcal{T}}$ satisfies a uniform shape regularity and standard a priori finite element estimates hold. Moreover the vertices of $\tilde{\mathcal{T}}_\ell$ are also vertices of $\tilde{\mathcal{T}}_{\ell+1}$. However the midpoints of the edges on level ℓ are, in general, not the new vertices on level $\ell + 1$. Consequently the arising sub-elements are not similar to one of the three sub-element classes which occur, if one tetrahedron is uniformly refined, see also Fig. 2.3.

Typically, low order finite element approximations are then based on the mesh sequence $\tilde{\mathcal{T}}$ and not on \mathcal{T} . The standard finite element space is defined as

$$V_\ell = \{v \in H_0^1(\Omega_\ell) : v|_T \in P_1(T), \forall T \in \tilde{\mathcal{T}}_\ell\},$$

where $\overline{\Omega}_\ell := \cup_{T \in \tilde{\mathcal{T}}_\ell} \overline{T}$. We point out that Ω_ℓ forms a sequence of domains approximating Ω and that due to the domain approximation there holds $V_\ell \not\subset V_{\ell+1} \subset H_0^1(\Omega)$. Thus, we are formally in a non-conforming setting, and in the a priori analysis variational crimes arise. The discrete Galerkin formulation of (2) then reads as: find $u_\ell \in V_\ell$ that satisfies

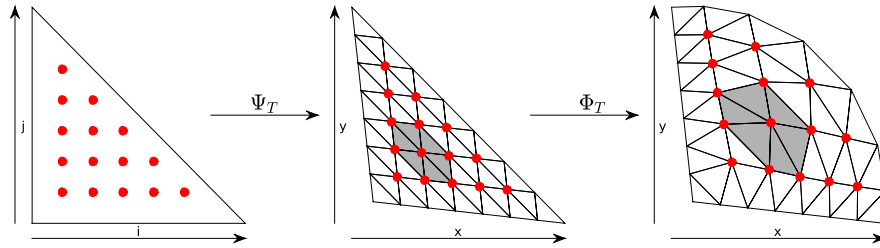


Fig. 2.2. Sketch of the different macro element mappings. In the reference domain (left) nodes are associated with the discrete indices (i, j, k) . Here k is fixed to get a 2D clipping, and we call it an index plane. Firstly the reference element \hat{T} is affinely mapped by Ψ_T onto a macro-element T (center). Secondly, the macro element T is mapped by Φ_T to the blended element (right).

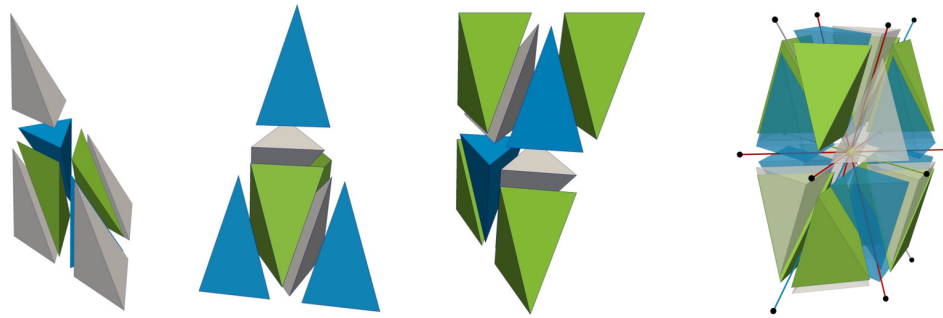


Fig. 2.3. 15-point stencil (right) and the three classes of sub-tetrahedra (the three pictures on the left). We note that refining a green or blue element does not yield a new class of sub-elements.

$$a_\ell(u_\ell, v_\ell) = \langle f, v_\ell \rangle_{\Omega_\ell}, \quad \forall v_\ell \in V_\ell \tag{3}$$

where $a_\ell(u_\ell, v_\ell) := \langle \nabla u_\ell, \nabla v_\ell \rangle_{\Omega_\ell}$. Let $\{\phi_I\}$ be the nodal basis of V_ℓ ; for simplicity we ignore in the notation of ϕ_I the level index ℓ , if there is no ambiguity. Then this transforms to the algebraic system

$$\mathcal{L}_\ell \mathbf{u}_\ell = \mathbf{f}_\ell \tag{4}$$

where \mathbf{u}_ℓ denotes the vector of nodal values of the finite element approximations, i.e., $u_\ell = \sum_I (\mathbf{u}_\ell)_I \phi_I$, and \mathbf{f}_ℓ stands for the algebraic representation of the linear form $(\mathbf{f}_\ell)_I = \langle f, \phi_I \rangle_{\Omega_\ell}$. As it is standard, the entries of $(\mathcal{L}_\ell)_{I,J}$ are given by $a_\ell(\phi_I, \phi_J)$. The standard matrix formulation (4) can be related to a representation of \mathcal{L}_ℓ in stencil form, i.e., each line of the matrix is associated with a node I , and its non-zero entries form the weights of the associated node stencil s^I .

2.3. Hybrid hierarchical mesh stencils

Massively parallel and highly scalable PDE software frameworks depend on a sophisticated data structure based on ghost layers. To reduce communication and memory traffic, the abstract concept of different hierarchical primitives can be applied. Here we use the hybrid hierarchical grids (HHG) framework which provides an excellent compromise between flexibility and performance [9,8,19]. While the coarse mesh can be completely unstructured, the fine mesh is obtained by uniform refinement following the rules given in [10]. Each element is refined into eight sub-elements each of which belongs to one of three different sub-classes, see the left of Fig. 2.3. All nodes of the triangulation can then be grouped into vertex, edge or volume primitives depending on their position with respect to the macro elements of the initial mesh \mathcal{T}_{-2} , see the right of Fig. 2.4. The vertex, edge and face primitives form the lower dimensional primitive classes and are associated with the surface/boundary of the macro elements. Such a structured block refinement then guarantees in the case of affine element mappings that the node stencils for partial differential equations with constant coefficients do not depend on the node location within one primitive. More precisely, for each macro element all volume primitive stencils are exactly the same and are given by a symmetric 15-point stencil, see the right of Fig. 2.3.

In large scale applications, the number of nodes in the volume primitive containers is significantly higher than the number of nodes in the lower dimensional primitive classes. It grows with $\mathcal{O}(2^{3\ell})$ whereas the number of remaining nodes increases only with $\mathcal{O}(2^{2\ell})$. Since we are interested in large scale simulations, the computational cost associated with the node stencils on the lower primitives is rather small compared to the cost associated with the nodes in the volume primitives. Thus we compute on-the-fly the node stencils for nodes in a lower dimensional primitive class in a straightforward way by re-assembling the contributions of local element matrices. Only the on-the-fly computation of the node stencils for nodes in a volume primitive class will be replaced by a surrogate stencil operator. We recall that the

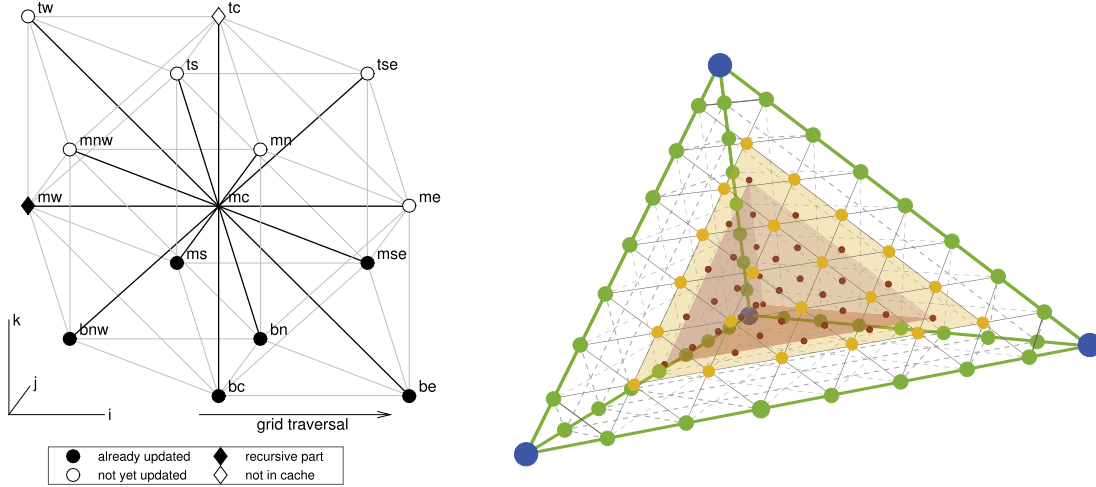


Fig. 2.4. (Left) Layout of a typical low order 15 pt stencil, see also [8]; node symbols will become relevant in Sec. 5; (right) association of nodes to the vertex (large circle), edge (medium circle), face (square) and volume (small circle) primitives for a single macro-element.

volume primitives are associated with the interior of the macro elements. Let $T \in \mathcal{T}_{-2}$ be fixed, then we denote by \mathcal{M}_ℓ the set of all points at refinement level $0 \leq \ell \leq L$ that are in the interior of T . These nodes belong to the volume primitive associated with T . For $\ell = 0$ this consists of exactly one single point. The number n_ℓ of elements in \mathcal{M}_ℓ is given by

$$n_\ell = \frac{1}{6} (2^{\ell+2} - 3) (2^{\ell+2} - 2) (2^{\ell+2} - 1) . \quad (5)$$

Remark 1. We point out that all our techniques can also be applied to the construction of stencil entries associated with face and edge nodes. This is of interest in case of moderate level hierarchies.

For a fixed node I , the stencil weight for accessing another node J is given by $a_\ell(\phi_I, \phi_J)$ and will only be non-zero, if both nodes are connected by an edge in \mathcal{T}_ℓ . In case of our uniform mesh refinement, the stencil weights can be identified by their cardinal directions $w \in \mathcal{W} = \{be, bc, \dots, mc\}$, where, as noted before, we have $|\mathcal{W}| = 15$. The first character in w denotes the bottom, middle and top plane whereas the second specifies the orientation within the plane, as illustrated in the left of Fig. 2.4. Then the stencil for node I can be represented by a vector $s^I = (a_\ell(\phi_I, \phi_{J_{be}}), \dots, a_\ell(\phi_I, \phi_{J_{mc}}))$ in \mathbb{R}^{15} . We note that $J_{mc} = I$.

Moreover there exists a unique mapping between the nodes in \mathcal{M}_ℓ and the set of index triples \mathcal{I}_ℓ given by

$$\left\{ (i, j, k) \mid 0 < i, j, k < 2^{\ell+2} - 2, i + j + k < 2^{\ell+2} \right\} .$$

Hence the node stencils on level ℓ can be associated with a stencil function $s: \mathcal{I}_\ell \rightarrow \mathbb{R}^{15}$, i.e. $s^I = s(i_I, j_I, k_I)$ where (i_I, j_I, k_I) is the index triple associated with node I . If $\Phi = \text{Id}$ then all $s^I, I \in \mathcal{M}_\ell$ are the same, i.e., s is a constant function. Moreover it is obtained by simple scaling from a reference stencil \hat{s} , i.e., $s^I = 2^{-\ell} \hat{s}, I \in \mathcal{M}_\ell$, with \hat{s} being the stencil associated with the single node in \mathcal{M}_0 . This is not the case for our model domain. Here s is a non-constant function and consequently all stencil weights have to be computed on-the-fly. To do so for each node $I \in \mathcal{M}_\ell$, we have to consider the 24 tetrahedra adjacent to it and their associated local element matrices which can be different from element to element due to a general Φ . Thus this process, although linear with respect to n_ℓ , can be rather cost intense. For $L = 6$ we have to compute on-the-fly 2,731,135 different stencils per macro element.

In Fig. 2.6, we illustrate the influence of Φ on the node stencil restricted to the index plane shown in Fig. 2.5. Note that for $\Phi = \text{Id}$ not only the element matrix but also the stencil is symmetric, i.e., opposite stencil entries, e.g., tw and be , are identical. While this kind of symmetry is not completely preserved in the more general case, it is still reflected. Hence, we show only seven stencil entries plus the central one in Fig. 2.6. One can clearly see that the stencil entries are smooth functions that can easily be approximated by polynomials. This observation motivates our new approach.

3. Two-scale approach

Our idea now is to replace the components of the exact stencil function s associated with a volume primitive node by surrogate polynomials of moderate order. More precisely, we consider for each cardinal direction $w \in \mathcal{W}$ a polynomial defined either by interpolation or alternatively by a discrete L^2 -best fit. We will only perform the replacement of the stencil

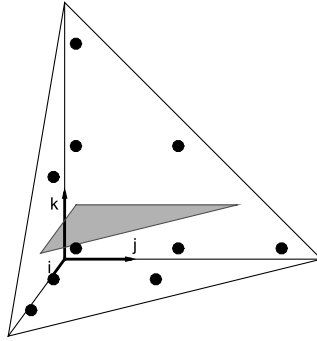


Fig. 2.5. Uniformly refined reference tetrahedron. Highlighted plane shows the location of nodal centers for the stencils visualized in Fig. 2.6 and Fig. 3.1. Black dots mark the ten sampling points for the IPOLY approach being introduced in Sec. 3.1.

function on levels $\ell \geq 1$ as on the coarsest level $\ell = 0$ a macro element has only one interior node with a single associated stencil.

3.1. Interpolation of stencils (IPOLY)

A tri-variate polynomial of degree less or equal to q can be represented as

$$p(i, j, k) = \sum_{\substack{l,m,n=0 \\ l+m+n \leq q}} a_{l,m,n} \lambda_{l,m,n}(i, j, k) , \tag{6}$$

where the $\lambda_{l,m,n}$ form a basis. Solving for the coefficients $\mathbf{a} = (a_{q,0,0}, \dots, a_{0,0,0}) \in \mathbb{R}^{m_q}$, where

$$m_q = \frac{1}{6}(q+3)(q+2)(q+1) , \tag{7}$$

of an interpolation polynomial in general leads to a linear system $\mathbf{A}\mathbf{a} = \mathbf{b}$ with a Vandermonde-type matrix $A \in \mathbb{R}^{m_q \times m_q}$ and sampling values $\mathbf{b} \in \mathbb{R}^{m_q}$. The special choice of a Lagrange interpolation basis yields $A = \text{Id}$. As example we consider the case $q = 2$. Here we have $m_2 = 10$ and for sampling we use the quadratic nodal interpolation points of a shrunk macro element. More precisely we use the nodes with indices

$$\begin{aligned} &(1, 1, 1), \\ &(1, 1, a), (1, a, 1), (a, 1, 1), \\ &(1, 1, b), (1, b, 1), (b, 1, 1), (1, b, b), (b, 1, b), (b, b, 1) \end{aligned}$$

where $a = 2^{\ell+2} - 3$, $b = 2^{\ell+1} - 1$ as sampling points, see also Fig. 2.5.

3.2. Least-squares approximation (LSQP)

Applying an interpolation polynomial implies that the polynomial is exact at the sampling points, which elevates these points. Alternatively, we can use a discrete L^2 -best approximation polynomial. This leads to a system with an $n_\ell \times m_q$ matrix, as we now use all points of \mathcal{M}_ℓ as our sampling set S_ℓ . More generally, we define the coefficients \mathbf{a} of the approximating polynomial as solution of the least-squares problem

$$\mathbf{a} := \underset{\mathbf{z} \in \mathbb{R}^{m_q}}{\text{argmin}} \|A\mathbf{z} - \mathbf{b}\|_2 , \quad A \in \mathbb{R}^{|S_\ell| \times m_q} , \quad \mathbf{b} \in \mathbb{R}^{|S_\ell|} .$$

Each row of A is defined by the values of the basis functions $\lambda_{l,m,n}$ evaluated at the corresponding sampling point. The choice $S_\ell = \mathcal{M}_\ell$ quickly becomes quite expensive, due to the fast growth of n_ℓ with ℓ . Thus we also consider reduced sampling sets $S_\ell := \mathcal{M}_{m(\ell,j)}$ with

$$m(\ell, j) := \min\{\ell, \max\{1, j\}\} \tag{8}$$

where j may depend on ℓ . The min-max definition guarantees that we neither exceed the largest possible set, i.e., \mathcal{M}_ℓ on level ℓ , nor go below \mathcal{M}_1 . For simplicity of notation, we also denote $\mathcal{M}_{m(\ell,j)}$ by \mathcal{M}_j if there is no ambiguity.

We note that for $q \leq 4$ and any choice of $S_\ell = \mathcal{M}_k$ with $k \in \{1, \dots, \ell\}$ we have, by comparing (5) and (7), that $n_\ell \geq m_q$. Moreover, the spatial distribution of the nodes in the mesh guarantees full rank of A and consequently the least-squares problem has a unique solution.

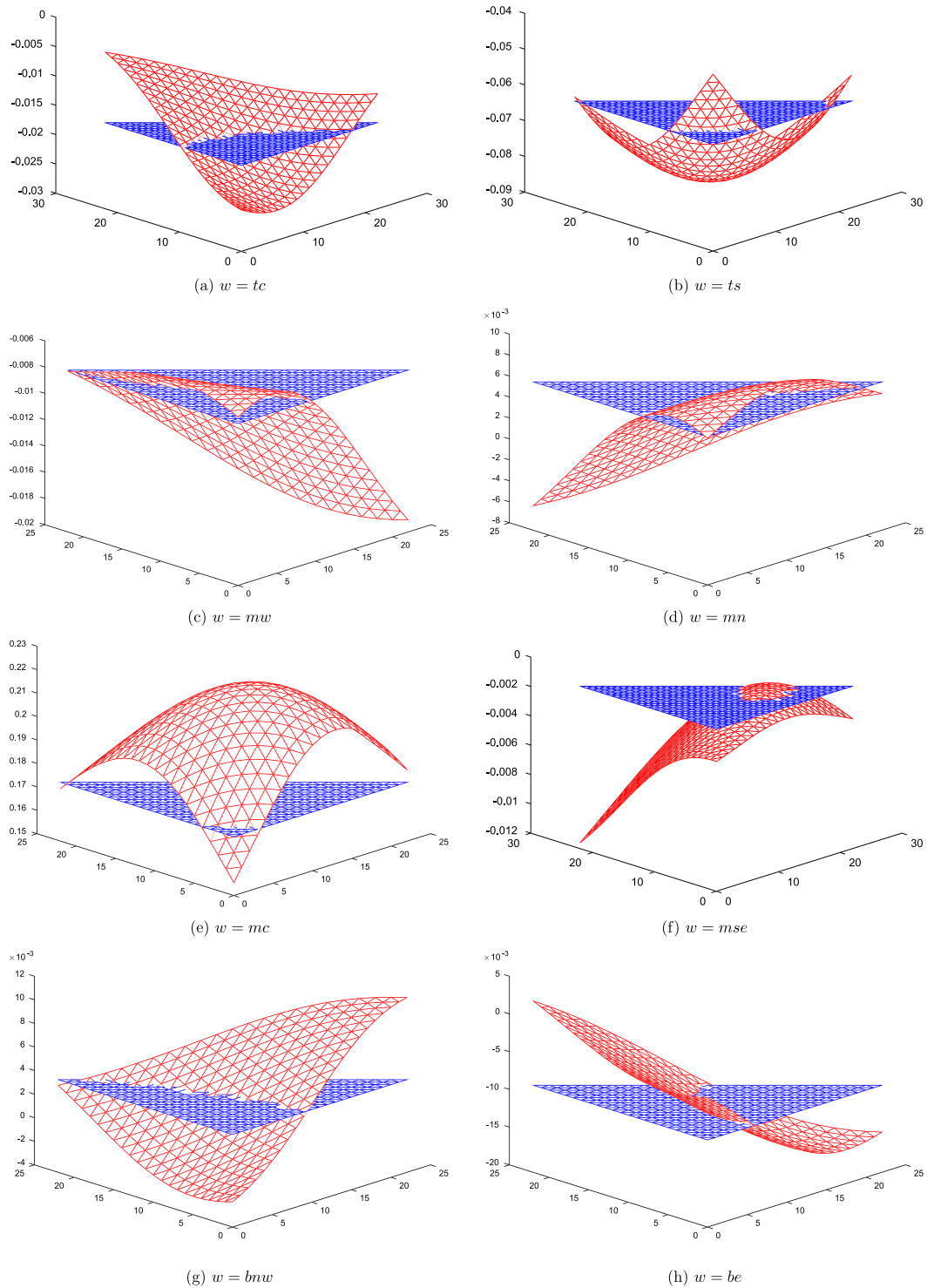


Fig. 2.6. Eight out of the 15 stencil values associated with the uniformly refined mesh series \mathcal{T} (blue) and the projected series $\tilde{\mathcal{T}}$ (red). Each subplot shows the stencil values of one $w \in \mathcal{W}$ for all fine grid nodes in the plane $k = 7$ of a macro element at refinement level $\ell = 3$. The horizontal axes give the i and j indices of the nodes, the vertical axis the stencil value. The fixed k index is chosen such that it contains the median node.

3.3. IPOLY versus LSQP for quadratic surrogate polynomials

In a first test, we compare the difference between the least-squares technique for $S_\ell = \mathcal{M}_{\ell-1}$ and the interpolation based approach, see Fig. 3.1. Note that, as in Fig. 2.6, the points visualized are chosen to be in the plane with $k = 7$. Hence the displayed corners are no sampling points for IPOLY. While the values of IPOLY are more accurate close to the boundary, the values of LSQP have a uniformly better fit.

In order to provide a more quantitative measure for the approximation accuracy, we introduce a metric for the overall fit. For $w \in \mathcal{W}$ we set

$$e_w := \left(\frac{\sum_{(i,j,k) \in \mathcal{I}_\ell} [s_w(i, j, k) - \mathcal{P}^\alpha s_w(i, j, k)]^2}{\sum_{(i,j,k) \in \mathcal{I}_\ell} s_w(i, j, k)^2} \right)^{1/2},$$

where $\mathcal{P}^\alpha s_w$, $\alpha \in \{\text{LSQP}, \text{IPOLY}\}$, denotes one of the proposed surrogate polynomials associated with the cardinal direction $w \in \mathcal{W}$. Obviously this metric corresponds to the discrete relative L^2 -norm of the error in approximating the w -th component of the stencil function s .

In Fig. 3.2, we illustrate the influence of the choice of the sampling set on LSQP and compare it to the IPOLY approach. Already for the relatively coarse macro mesh we observe a maximal deviation of no more than 12 percent. When the macro mesh size is halved, then also the error in the approximation is decreased by about a factor of two. For the finer macro mesh we observe two peaks. These result from the following. The blending function Φ_T depends on the location of the macro element T within the spherical shell and different directions are distorted differently. The precise Φ_T determines which $w \in \mathcal{W}$ are affected most strongly by the approximation.

Comparing LSQP and IPOLY we find that the relative error is smaller for LSQP for all sampling sets. Already for the smallest choice \mathcal{M}_1 , i.e., $n_1 = 35$, it is significantly better than IPOLY. This comes as no surprise, as the least-squares approach constructs a minimizer in the Euclidean norm. We note that \mathcal{M}_2 , i.e., $n_2 = 455$, is, in general, a good compromise between quality and problem size of the stencil approximation, in particular if the number of macro elements is already large.

3.4. Definition of the surrogate operator

We define our new two-scale finite element approximation as the solution of

$$\mathcal{L}_\ell^{H,\alpha,q} \mathbf{u}_\ell^{H,\alpha,q} = \mathbf{f}_\ell. \tag{9}$$

Here, the superscripts mark the dependencies on the macro element size H , the degree q of the surrogate polynomial, and the selected approach $\alpha \in \{\text{IPOLY}, \text{LSQP}\}$. For ease of notation, we replace the superscript triple by a tilde, i.e., we set $\tilde{\mathcal{L}}_\ell := \mathcal{L}_\ell^{H,\alpha,q}$ and similarly for $\mathbf{u}_\ell^{H,\alpha,q}$.

As is the case with \mathcal{L}_ℓ , each row of $\tilde{\mathcal{L}}_\ell$ is associated with a node. In our approach, we replace for all nodes in a volume primitive the exact finite element stencil on level ℓ by its surrogate polynomial approximation, i.e., instead of computing $s_w(i, j, k)$ on-the-fly, we evaluate the surrogate $\mathcal{P}^\alpha s_w(i, j, k)$. As already mentioned, the stencil entries associated with nodes on lower dimensional primitives are assembled on-the-fly in standard fashion.

Remark 2. We point out that, if $\Phi_T = \text{Id}$, $T \in \mathcal{T}_{-2}$, then both our approaches reduces to the standard one. Also, they can be applied selectively and switched on or off macro element-wise depending on the location of T within the domain or its shape. Additionally the choice of the polynomial degree can depend on T , and we can even use anisotropic orders reflecting the properties of the mapping Φ_T .

Since we have one stencil weight for each of the 15 couplings $w \in \mathcal{W}$, we use 15 different second order polynomials, described by 15 different sets of coefficients \mathbf{a}_w . The latter are computed in a setup phase. This, firstly, requires for each macro element and each level $\ell \geq 1$ the evaluation of m_q (IPOLY) or $|\mathcal{S}_\ell|$ (LSQP) stencils and then the solution of a linear or least-squares system.

Once the setup is completed, whenever the stencil for node I is to be applied, e.g. as part of a smoothing step in a multigrid method, or a residual computation, 15 polynomials of type (6) must be evaluated at this node to provide the surrogate stencil weights. At first glance, this does not seem to be any faster than a standard implementation. Recall, however, that within a typical matrix-free framework, the on-the-fly evaluation of the stencil involves costly operations such as the (re-)computation of local element matrices or the direct evaluation of weak forms via quadrature rules. The advantage of our approach is that the setup needs to be performed only once. The operator assembly itself reduces to polynomial evaluations. This also implies that the cost for the actual stencil application is the same for both, IPOLY and LSQP. Only in the setup phase is LSQP more expensive as more sampling points need to be evaluated and the minimization problem includes a larger system matrix.

Compared to a full stencil pre-calculation and storage scheme also the amount of required memory can be neglected. We only need $15m_q L$ coefficients per macro element to fully describe the approximated stencils. Such a small number of coefficients can easily be stored even when the refinement depth L is large.

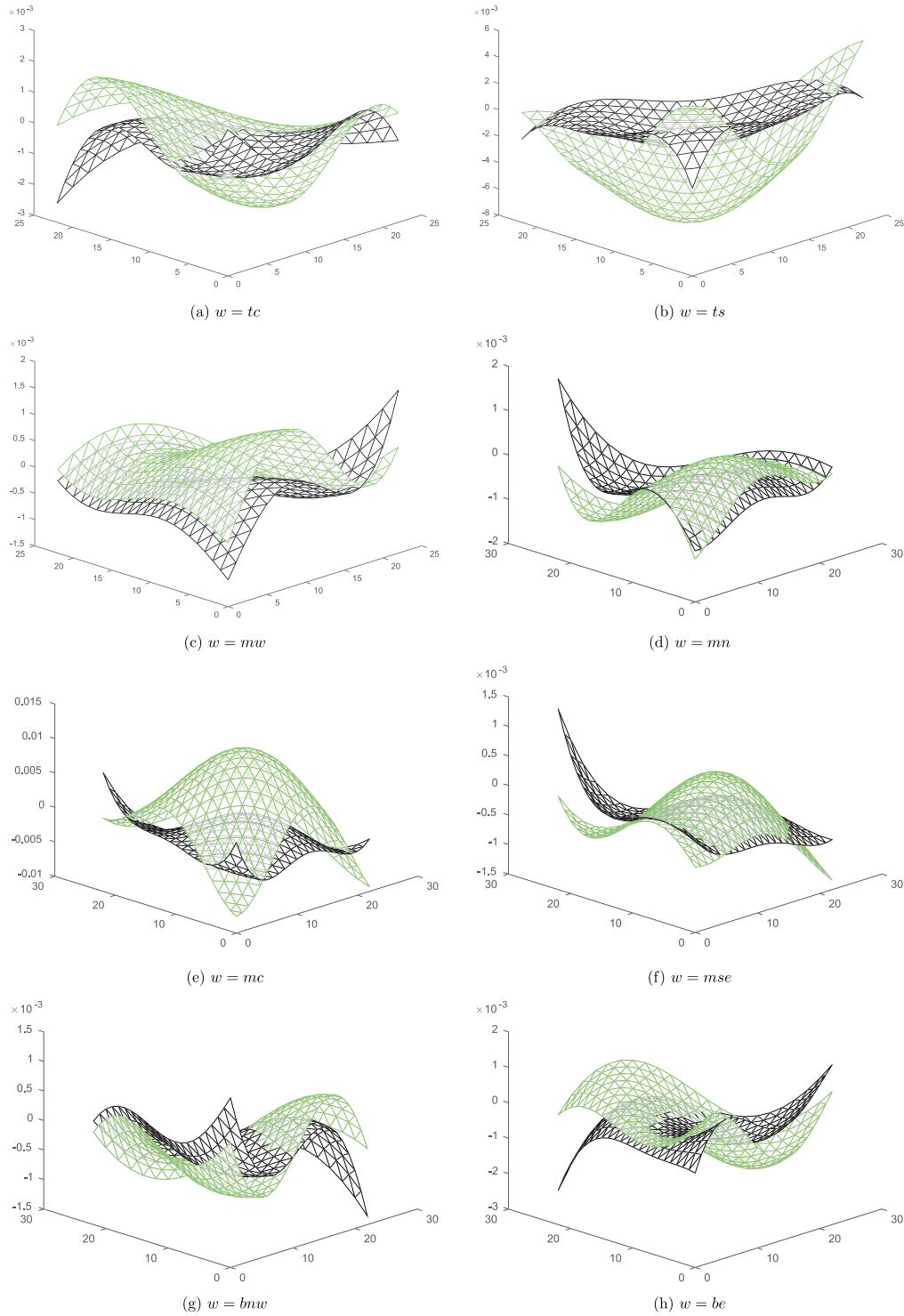


Fig. 3.1. Difference between IFEM and IPOLY (green) or LSQP (black) for the same setting than in Fig. 2.6.

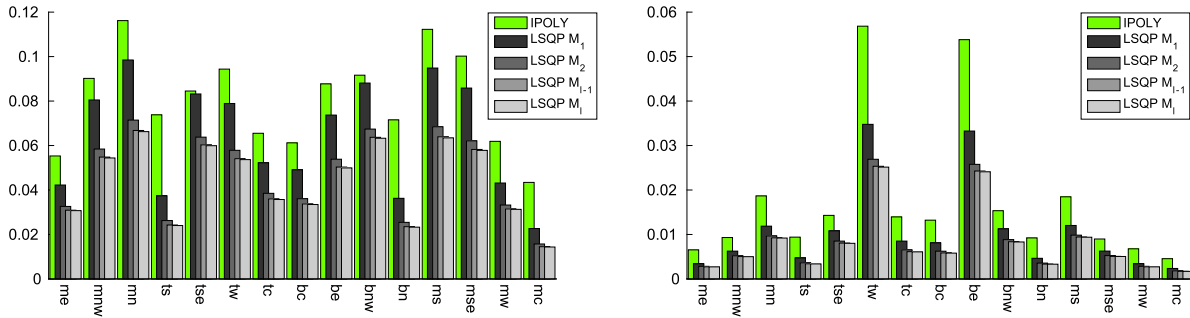


Fig. 3.2. e_w for IPOLY and LSQP with different choices for the sampling set. Left plot is for a macro mesh that consists of 60 elements, while in the right plot the macro mesh size is halved, resulting in 480 elements. For both we set $L = 4$.

Remark 3. Alternatively, one can think of using the same fine level surrogate polynomial to evaluate the stencils on the coarser meshes. A study of this idea and its effect on the multigrid convergence rate is beyond the scope of the paper.

We want to point out that in the application of the surrogate operator we loop over the indices (i, j, k) in a line-wise fashion. This implies that we can always fix two of the three indices and, consequently, only need to evaluate a 1D and not a 3D polynomial. This reduces the cost significantly as the 1D polynomial has fewer terms. Evaluation of the 1D polynomial itself can be accomplished very efficiently by using a hierarchical recursion formula. More precisely starting from a single direct evaluation of $\hat{p}(i) := \mathcal{P}^\alpha s_w(i, j_0, k_0)$, we can evaluate the polynomial incrementally in terms of q basic operations. In computer graphics this technique is known as forward differencing [34,32].

Apart from its cost, the accuracy of a numerical scheme is, of course, of major importance. In this context we provide the following lemma related to the consistency of the discretization expressed by $\tilde{\mathcal{L}}_\ell$.

Lemma 1. *The row sum property of the stencil*

$$\sum_{w \in \mathcal{W}} s_w^l = 0$$

is preserved by IPOLY and LSQP, when using identical polynomial order and sampling sets for all stencil weights.

Proof. We recall that the coefficients of both our approaches can formally be obtained by solving a system of normal equations. As A has full rank its solution is unique and

$$\mathbf{a}_{mc} = \left(A^\top A \right)^{-1} A^\top \mathbf{b}_{mc} = - \left(A^\top A \right)^{-1} A^\top \sum_{\substack{w \in \mathcal{W} \\ w \neq mc}} \mathbf{b}_w = - \sum_{\substack{w \in \mathcal{W} \\ w \neq mc}} \mathbf{a}_w .$$

Thus, the row sum condition holds for the coefficients of the surrogate polynomials and, consequently, also for the stencil entries approximated by an evaluation of the polynomials at each node position. \square

Finally, we recall that the matrix \mathcal{L}_ℓ is symmetric, as $a_\ell(\phi_I, \phi_J) = a_\ell(\phi_J, \phi_I)$. This property is not preserved by our approach, due to the fact that the polynomials are evaluated at the nodes. Let I be a node, and J be the neighboring node which is reached, if we move from I in cardinal direction w . By w^0 , we denote the opposite cardinal direction, i.e., if we move from J into the direction of w^0 then we recover I . In our approach, we find

$$\begin{aligned} (\tilde{\mathcal{L}}_\ell)_{IJ} &= \mathcal{P}^\alpha s_w(i_I, j_I, k_I) \\ (\tilde{\mathcal{L}}_\ell)_{JI} &= \mathcal{P}^\alpha s_{w^0}(i_J, j_J, k_J) . \end{aligned}$$

To quantify this loss of symmetry, we measure it in the relative Frobenius norm. We consider

$$\frac{\|(\tilde{\mathcal{L}}_\ell|_T)^\top - \tilde{\mathcal{L}}_\ell|_T\|_F}{\|\mathcal{L}_\ell|_T\|_F} .$$

Here $\tilde{\mathcal{L}}_\ell|_T$ is the restriction of the global system matrix to one macro element. As expected, for both IPOLY and LSQP we find that the relative non-symmetry is in $\mathcal{O}(h_\ell)$.

Remark 4. A symmetric matrix can also be recovered. One possibility is to define only seven cardinal directions, associated with the seven edge directions of a macro element. For this we identify the cardinal directions w and w^0 and evaluate the

surrogate polynomial at the center of the edge connecting two nodes. Note that in that case, we need to define the central stencil weight via the row sum condition. As we will see in Sec. 5, a direct evaluation of the row sum is more expensive than the evaluation of a surrogate polynomial of moderate order. Therefore we will not further consider this option.

3.5. Some remarks on a priori estimates

In this section, we will discuss the IPOLY and LSQP discretization errors in the L^2 -norm. Let \mathbf{u}_ℓ and $\tilde{\mathbf{u}}_\ell$ be the solution of (4) and (9), respectively, and u the exact solution of (1). Since \mathcal{L}_ℓ is a positive definite operator, we can rewrite the surrogate operator $\tilde{\mathcal{L}}_\ell$ as

$$\tilde{\mathcal{L}}_\ell = \mathcal{L}_\ell \left(\text{Id} + H^{q+1} \tilde{\mathbf{B}}_\ell \right).$$

For our heuristic considerations we assume that $\tilde{\mathbf{B}}_\ell$ satisfies

$$\|\tilde{\mathbf{B}}_\ell\| \leq C_{q,\alpha} < \infty, \quad (10)$$

where $\|\cdot\|$ denotes the Euclidean norm. Our assumption is motivated by the presumed smoothness of the mapping Φ described in Sec. 2.2, in combination with best approximation and interpolation results. We point out that the stencil values can be interpreted as a function depending on $D\Phi D\Phi^\top / |\det D\Phi|$ where $D\Phi$ is the Jacobian of Φ . Thus, an approximation with a polynomial of degree at most q results in an error of $\mathcal{O}(H^{q+1})$. Moreover by construction we preserve the row sum property, and consequently the kernel of \mathcal{L}_ℓ and $\tilde{\mathcal{L}}_\ell$ is locally the same. The notation \lesssim is used for $\leq C$ with an ℓ, H -independent constant $C < \infty$.

Furthermore, we assume that H is chosen small enough such that $H^{q+1} \|\tilde{\mathbf{B}}_\ell\| < 1$. By employing the properties of the Neumann series, we then obtain

$$\begin{aligned} \tilde{\mathcal{L}}_\ell^{-1} &= \left(\text{Id} + H^{q+1} \tilde{\mathbf{B}}_\ell \right)^{-1} \mathcal{L}_\ell^{-1} = \left[\sum_{k=0}^{\infty} \left(-H^{q+1} \tilde{\mathbf{B}}_\ell \right)^k \right] \mathcal{L}_\ell^{-1} \\ &= \left(\text{Id} - H^{q+1} \tilde{\mathbf{B}}_\ell \right) \mathcal{L}_\ell^{-1} + \mathcal{O}\left(H^{2(q+1)}\right). \end{aligned}$$

Neglecting the higher order terms, we arrive at

$$\begin{aligned} \|\tilde{\mathbf{u}}_\ell - u_\ell\|_0^2 &\lesssim h^3 \|\tilde{\mathbf{u}}_\ell - \mathbf{u}_\ell\|^2 = h^3 \|\mathcal{L}_\ell^{-1} \mathbf{f}_\ell - \tilde{\mathcal{L}}_\ell^{-1} \mathbf{f}_\ell\|^2 \\ &\lesssim h^3 \|H^{q+1} \tilde{\mathbf{B}}_\ell \mathcal{L}_\ell^{-1} \mathbf{f}_\ell\|^2 \lesssim h^3 H^{2(q+1)} \|\tilde{\mathbf{B}}_\ell\|^2 \|\mathbf{u}_\ell\|^2 \\ &\lesssim H^{2(q+1)} \|\tilde{\mathbf{B}}_\ell\|^2 \|u_\ell\|_0^2 \lesssim H^{2(q+1)} \|\tilde{\mathbf{B}}_\ell\|^2 \|f\|_0^2. \end{aligned}$$

It is well known that under suitable regularity assumptions $\|u - u_\ell\|_0 \lesssim h_\ell^2 \|f\|_0$. Then the triangle inequality yields

$$\|u - \tilde{\mathbf{u}}_\ell\|_0 \lesssim \left(h_\ell^2 + H^{q+1} \right) \|f\|_0 \lesssim h_\ell^2 \left(1 + 4^{l+2} H^{q-1} \right) \|f\|_0. \quad (11)$$

We note that the generic constants do depend on q , higher order derivatives of the mapping function Φ and α , the type of selected polynomial approximation.

Remark 5. Keeping H fixed and increasing ℓ gives an asymptotically constant error not equal to zero. However, if a given pairing (h_ℓ, H) results in an error not dominated by the second term in (11), then this is also correct for any pairing of the form $(h_{\ell+2k}, H/2^k)$ in the case of $q=3$. Moreover for $\ell \leq L$ and L fixed and for a macro mesh size $H < H_0$, we also obtain optimal order results in h_ℓ , if $q \geq 2$. We note that H_0 depends on L and q , but also on the data.

4. Evaluation of accuracy and cost

For our numerical studies, we employ model problem (1) with an exact solution given by

$$\tilde{u}(x, y, z) = (r - r_1)(r - r_2) \sin(10x) \sin(4y) \sin(7z)$$

where $r = r(x, y, z) = (x^2 + y^2 + z^2)^{1/2}$. The right-hand side is set accordingly, and the homogeneous Dirichlet boundary conditions are automatically satisfied by \tilde{u} . For the model domain that represents the Earth mantle we set $r_1 = 0.546$ and $r_2 = 1$. An illustration is given in the left plot of Fig. 4.1. The right plot shows the finite element error associated with the mesh sequence \mathcal{T} . It is obviously dominated by the inaccurate boundary approximation of Ω_H and is of second order in H .

To solve the discrete problem, we employ a stand-alone geometric multigrid solver. It uses a V(3, 3)-cycle with a lexicographic hybrid Gauss–Seidel (GS) smoother, which also serves as coarse grid solver. The smoother is hybrid in the sense that, due to the HHG communication patterns between primitives, some dependencies are neglected and low-order primitives can have some points that are only updated in a Jacobi-like fashion, see [9] for details. However, the majority of points is

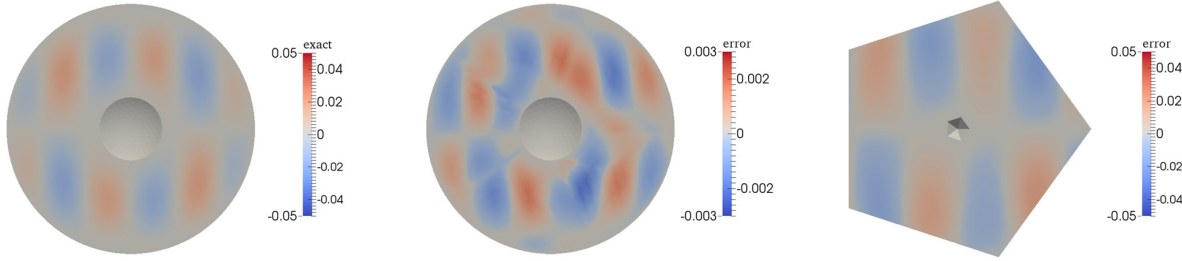


Fig. 4.1. Cut parallel to x, y -plane at $z = -0.5$. Exact solution \bar{u} (left), error $\bar{u} - u_{\text{IFEM}}$ (middle) and $\bar{u} - u_{\text{cons}}$ (right). On the boundary the error w.r.t. the solution u_{cons} of the unprojected approach is equal to the exact \bar{u} as we impose zero Dirichlet boundary conditions. The maximal error of u_{IFEM} is one order of magnitude smaller than that of u_{cons} . The slight asymmetry visible in the middle plot results from the unstructuredness of the macro grid.

updated as in a standard GS scheme. Experiments with different numbers of smoothing steps have shown qualitatively the same results. Although new vertices on level ℓ are not necessarily located at the midpoint of the edges on level $\ell - 1$, the node connectivity is not changed. Thus we apply the standard prolongation and restriction operators of a conforming low order finite element setting. The iteration is started with a random initial guess.

A standard geometric multigrid solver contains two operations which involve the application of the stencils. These are *smoothing* and *residual computation* for the coarse grid correction. Instead of applying either the stencil for \mathcal{L}_ℓ or $\tilde{\mathcal{L}}_\ell$ in both operations, it is also possible to apply a mixed formulation. This is a classical technique in the multigrid literature [12, Chapter 10.2] known as *double discretization* (DD). It consists in a combination of defect correction and multigrid [21]. Different from conventional defect correction, the higher order discretization is only used when computing the residuals for the coarse grid correction while the smoother is employed with the low order discretization. This has the effect that the multigrid algorithm combines two competing iterations having distinct fixed points and the combined iteration generally converges to neither of the two original limits. With this, the multigrid method becomes an integrated solver for the continuous problem that blends the properties of several discretizations. In particular it can be shown that under suitable conditions [21,12] the order of accuracy delivered by multigrid with double discretization is determined by the order of the discretization that is employed for the coarse grid correction. The discretization used in smoothing has only a mild effect. Therefore this technique can be used to design extremely efficient higher order solvers, since the complexity of a higher order discretization will affect the cost of each iteration of the linear multigrid solver only moderately.

We include this technique in our tests. For this we apply the less expensive surrogate operator $\tilde{\mathcal{L}}_\ell$ for the relaxation sweeps, as those require less accuracy, and the exact one \mathcal{L}_ℓ for residual computation. Using two different operators leads to two competing components in the multigrid method, as the smoother and the coarse grid correction aim to drive the iterate towards the algebraic solution of their respective operators. Hence in the double discretization approach, the algebraic convergence will typically suffer, and residual-based stopping criteria are inappropriate. Instead one should, e.g., check the sequence of updates for convergence to zero [12].

4.1. Accuracy of IPOLY and LSQP

For our numerical study, we will compare the standard Galerkin formulation with projected coordinates, termed IFEM, against the IPOLY and LSQP approaches. For the latter, we will also test the aforementioned double discretization technique. The resulting iterative schemes are denoted as IPOLY DD and LSQP DD. Besides this we will show finite element results for an unprojected approach (CONS), i.e. with a fixed Ω_H . In order to further stress out the quality of the double discretization approach, we test it using this unprojected operator for smoothing, giving us CONS DD. If not explicitly mentioned otherwise, all IPOLY and LSQP based results use quadratic polynomials. For LSQP we set $j = 2$ in $m(\ell, j)$, i.e. $S_\ell = M_2$. Using computationally more expensive choices of S_ℓ did not yield any significant differences.

We start our tests with an initial mesh \mathcal{T}_{-2} consisting of 60 macro elements and set $L \in \{1, \dots, 6\}$. To measure the discretization error, we introduce the discrete L^2 -norm

$$e := h_\ell^{3/2} \|I_\ell(\bar{u}) - \mathbf{v}_\ell\|_2,$$

where I_ℓ denotes the nodal interpolation operator, and \mathbf{v}_ℓ represents either the actual multigrid approximation or the final iterate of the employed scheme. This error is equivalent to the L^2 -error of the associated finite element functions. In order to be sure to have reached the asymptotic regime, we perform 10 multigrid cycles. Results for $L = 5$ are given in the left part of Fig. 4.2.

One can see that the discretization error for CONS clearly suffers from the insufficient resolution of the spherical geometry. The errors for IPOLY and LSQP are roughly a factor of 10 and 50 smaller, respectively. We observe that LSQP performs better than IPOLY. This supports the observation from Sec. 3.2 that the stencils are uniformly better fitted. Note that, as was to be expected for a rather coarse input mesh (large H) and multiple refinements (large L), the error is dominated by the approximation error in the stencil s introduced by replacing its values by those of the surrogate polynomials. Using the

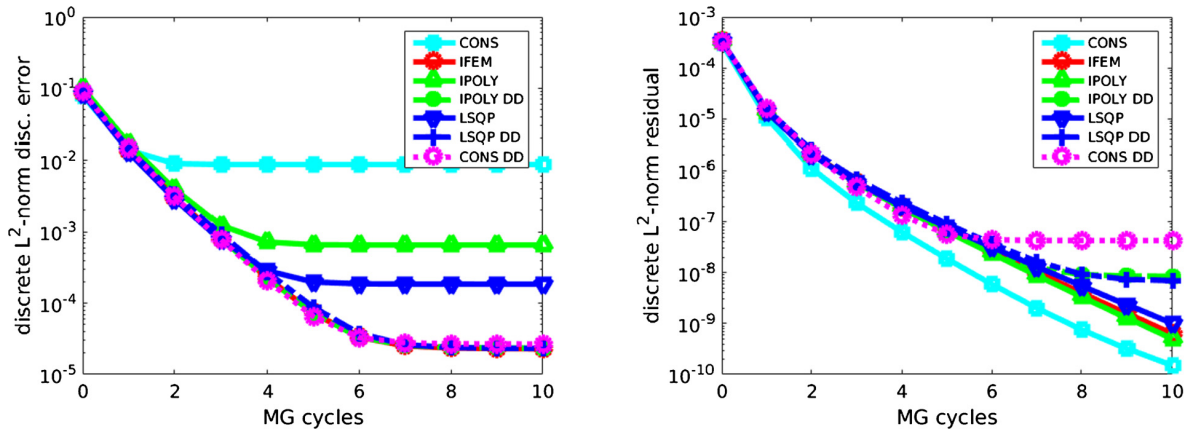


Fig. 4.2. Development of the error (left) and residual (right) in the iterates in discrete L^2 -norm for an input mesh of 60 macro tetrahedra and refinement level $L = 5$.

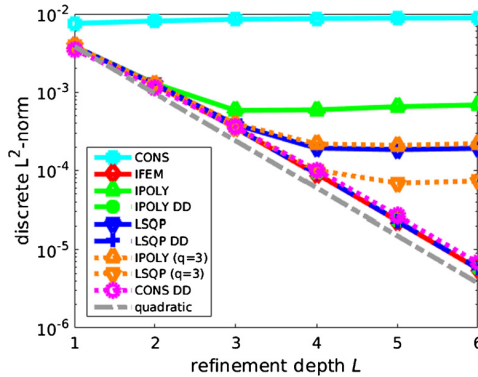


Fig. 4.3. Discretization error in discrete L^2 -norm for a sequence of refinement levels L for an input mesh of 60 macro tetrahedra.

double discretization scheme, however, allows to (almost) recover the discretization error of the IFEM approach. Here, both IPOLY DD and LSQP DD perform equally well and even CONS DD achieves very good results.

In the right part of Fig. 4.2, we provide the discrete L^2 -norms of the residuals. For IPOLY and LSQP these are based on the respective surrogate matrix. We observe that the convergence for the DD variants breaks down after a few iterations. As already indicated, this was to be expected due to the inconsistency in smoothing and residual computation. Whereas the residuals of all non-DD-scenarios show similar asymptotic convergence rates.

In Fig. 4.3, we show the discrete L^2 -error for increasing refinement levels. As expected, IFEM shows quadratic order convergence.

This is also satisfied by the DD approaches. But, as already mentioned before, this does not hold for IPOLY and LSQP. Nevertheless, we observe in the pre-asymptotic regime a second order error decay and only asymptotically, the error is dominated by the term in H^{q+1} , see also (11). For LSQP we observe second order for $L \leq 3$, while for IPOLY this is only the case for $L \leq 2$.

While DD shows excellent convergence, it also requires the application of the expensive operator \mathcal{L}_ℓ . Although it is used only every seventh operator call, in case of a $V(3, 3)$ cycle, it still dominates the time to solution. This can be seen in the scaling results in Sec. 6.

Motivated by the a priori estimates from Sec. 3.5, we further investigate the dependence of LSQP and IPOLY on the refinement level L , the macro mesh size H and the polynomial degree q . The above test setting describes the coarsest macro mesh with 60 elements. Further refinement in radial and tangential direction results in input meshes \mathcal{T}_{-2} with 480, 3,840 and 30,720 macro elements, respectively. The discretization error for IFEM, LSQP and IPOLY after 10 multigrid cycles is displayed in Table 4.1 for different numbers of macro elements and different polynomial degrees, $q = 1, 2, 3$.

As expected, we see in Table 4.1 a quadratic convergence for IFEM. For the other variants we observe that for a finer input mesh and for a higher polynomial degree, the accuracy obtained increases. Combinations for which e_{LSQP} or e_{IPOLY} differ relatively by more than 10 percent from e_{IFEM} are highlighted by italic table entries.

We find that for IPOLY and $q = 1$ the maximal feasible refinement level L is independent of H . While for $q = 2$ two refinements in H allow one more refinement in L . And for $q = 3$ those values form a diagonal, meaning that for each finer

Table 4.1
Discretization error for IFEM and IPOLY/LSQP (linear, quadratic and cubic) for different combinations of h_ℓ and H . Italic entries have a relative deviation from the reference e_{IFEM} of more than 10 percent.

#mac el	60	480	3840	30720
	e_{IFEM}			
$L = 2$	1.2e-03	3.5e-04	8.9e-05	2.2e-05
$L = 3$	3.4e-04	9.4e-05	2.2e-05	5.6e-06
$L = 4$	9.0e-05	2.3e-05	5.7e-06	1.4e-06
$L = 5$	2.2e-05	5.9e-06	1.4e-06	3.5e-07
$L = 6$	5.7e-06	1.5e-06	3.5e-07	8.9e-08
	$e_{\text{IPOLY}} (q = 1)$			
$L = 2$	1.5e-03	4.1e-04	8.9e-05	2.2e-05
$L = 3$	2.1e-03	5.6e-04	1.1e-04	2.8e-05
$L = 4$	2.7e-03	7.3e-04	1.5e-04	3.8e-05
$L = 5$	3.1e-03	8.2e-04	1.8e-04	4.4e-05
$L = 6$	3.3e-03	8.7e-04	1.9e-04	4.7e-05
	$e_{\text{IPOLY}} (q = 2)$			
$L = 2$	1.3e-03	3.6e-04	9.0e-05	2.3e-05
$L = 3$	5.9e-04	1.0e-04	2.3e-05	5.7e-06
$L = 4$	5.9e-04	6.3e-05	6.5e-06	1.5e-06
$L = 5$	6.5e-04	6.7e-05	3.4e-06	4.0e-07
$L = 6$	6.9e-04	7.1e-05	3.1e-06	1.7e-07
	$e_{\text{IPOLY}} (q = 3)$			
$L = 2$	1.3e-03	3.6e-04	9.0e-05	2.3e-05
$L = 3$	4.0e-04	9.6e-05	2.3e-05	5.7e-06
$L = 4$	2.2e-04	2.7e-05	5.9e-06	1.4e-06
$L = 5$	2.1e-04	1.2e-05	1.6e-06	3.7e-07
$L = 6$	2.2e-04	9.9e-06	6.4e-07	1.0e-07
	$e_{\text{LSQP}} (q = 1)$			
$L = 2$	1.3e-03	3.7e-04	9.0e-05	2.3e-05
$L = 3$	7.7e-04	1.4e-04	2.7e-05	6.4e-06
$L = 4$	8.0e-04	1.1e-04	1.5e-05	3.1e-06
$L = 5$	8.5e-04	1.2e-04	1.4e-05	2.8e-06
$L = 6$	8.8e-04	1.2e-04	1.5e-05	2.9e-06
	$e_{\text{LSQP}} (q = 2)$			
$L = 2$	1.3e-03	3.6e-04	8.9e-05	2.3e-05
$L = 3$	3.9e-04	9.4e-05	2.3e-05	5.7e-06
$L = 4$	1.9e-04	2.5e-05	5.6e-06	1.4e-06
$L = 5$	1.8e-04	1.3e-05	1.4e-06	3.5e-07
$L = 6$	1.9e-04	1.3e-05	3.8e-07	8.2e-08
	$e_{\text{LSQP}} (q = 3)$			
$L = 2$	1.3e-03	3.6e-04	8.9e-05	2.3e-05
$L = 3$	3.5e-04	9.4e-05	2.3e-05	5.7e-06
$L = 4$	1.0e-04	2.4e-05	5.7e-06	1.4e-06
$L = 5$	6.9e-05	5.8e-06	1.4e-06	3.5e-07
$L = 6$	7.3e-05	1.9e-06	3.3e-07	8.7e-08

H also L can be increased by one. For LSQP the results look even better. As a rule of thumb we observe that for LSQP the italic entries are shifted by at least one level down compared to IPOLY.

These observations confirm our theory that the discretization error is a combination of two parts: one fine scale part h_ℓ^2 and one depending on the coarse mesh size H , but being of higher order depending on the selected polynomial degree.

Real geophysical applications will typically be executed on HPC systems with $\mathcal{O}(10^5)$ cores. Each core will then manage at least one, but often also more than one macro element, imposing a small upper bound on H and thus a reasonable large bound on L . Hence, in these application we will be automatically in an (L, H) -parameter regime that does not exceed the critical L . Consequently our two-scale approach is perfectly suited for this kind of applications.

4.2. Cost considerations

Besides the quality of the approximation, we also have to consider its cost. We split our considerations into two parts, the cost for applying and the cost for setting up the surrogate operator. While our two approaches IPOLY and LSQP have different setup costs, the cost for the evaluation of the surrogate stencil operator is the same in both surrogate approaches. Moreover once the stencil weights are evaluated, the application of the actual stencil is independent of the chosen approach.

Table 4.2

Number of FLOPs for a single stencil application in a Gauss–Seidel relaxation step for a scalar operator in HHG. For fair comparison DD cost is averaged over cheap smoothing and expensive residual computation.

Constant	29
IFEM (direct)	1353
IFEM (row sum)	1343
IPOLY/LSQP (naïve, $q = 2$)	378
IPOLY/LSQP (increment)	$29 + 15q$
CONS DD	$\frac{1343+29\nu}{\nu+1}$

4.2.1. Cost for operator evaluation

To give a quantitative idea Table 4.2 provides the number of FLOPs for a single Gauss–Seidel relaxation step, which involves the equivalent of one stencil application, see (13) below. We base our considerations, w.r.t. polynomial evaluation, on the incremental approach. Note that the traditional FLOPs measure can only give an impression on the arithmetic load of the different variants. The actual execution times depend on various implementation details, see also Sec. 5 where we discuss how to optimize the polynomial evaluation with respect to run-time.

For the constant case, we get in Table 4.2 the minimal value of 29 FLOPs, as the same pre-computed stencil can be used at all nodes. As mentioned above this cost for the stencil application remains the same in the other approaches. However, the extra cost for computing the proper finite element or surrogate stencil entries occur. As a small detail note that in the constant case HHG employs a multiplication with the reciprocal of the central stencil weight in (13) which gets pre-computed in the setup phase. For the other approaches we need to perform a floating point division, which is more costly.

Setting up the stencil in the IFEM approach with projected coordinates involves firstly computing for the current node the element matrices of its surrounding 24 elements. For this we use a C++ code that was auto-generated and optimized using the FEniCS Form Compiler (FFC), see [29], which results in 53 FLOPs per element matrix. Stencil weights are obtained by summing up the associated entries in the element matrices. In our 15-point-stencil, we have 6 weights belonging to edges attached to 4 neighboring elements and 8 weights belonging to edges attached to 6 neighboring elements, see Fig. 2.3. Hence, this results in $6 \cdot 3 + 8 \cdot 5 = 58$ additions for computing the non-central stencil weights. The central weight can either be assembled in the same fashion (23 FLOPs) or via the row sum property (13 FLOPs). Altogether this leads to the cost of 1353 resp. 1343 FLOPs given in Table 4.2. Thus, this approach is nearly a factor of 50 more costly than the constant one. Though, to be fair, we remark that our implementation does not take into account the fact that each element matrix is shared by four nodes. However, exploiting this is not straightforward as the point-smoother we are considering here requires an assembled central stencil weight and re-using element matrices for different nodes introduces additional overhead to manage the related dependency information.

We now turn to our proposed surrogate stencils. Here we need to add on top of the cost for applying the stencil the evaluation of the polynomials to obtain its weights. If we employ the naïve approach, i.e., a straightforward evaluation with respect to its monomial basis representation, we end up with 378 FLOPs for a tri-variate quadratic polynomial. Here, we assumed the sum rule is employed for the central weight. This already gains a factor of 3.5 compared to IFEM, but still requires over ten times more FLOPs than in the constant case. Now, for the incremental approach, asymptotically, only two operations are needed for the evaluation of each polynomial. Thus, the total number of FLOPs decreases to 59, which is only about twice the optimal value of the constant setting. Note that in this case, evaluation of the central entry via its polynomial is, of course, cheaper than the sum rule. Using higher order polynomials of degree q , we asymptotically need q operations and an additional cost factor of

$$C(q) = 1 + \frac{15q}{29} \approx 1 + \frac{q}{2}. \quad (12)$$

So the cubic IPOLY/LSQP approach, overall, is a factor of 1.25 more expensive than the quadratic one.

To quantify the cost of a CONST double discretization approach in comparison with a pure IPOLY/LSQP approach ($q = 2$), we take the average over one residual computation and a total of ν pre- and post-smoothing steps. For a typical choice of ν ranging between 2 and 8 the resulting cost factor is roughly between 8 and 3. This demonstrates that DD is an option that should seriously be considered, but overall is still dominated by the expensive residual computation.

In order to reach for a fixed h an acceptable accuracy H and q have to be selected properly. If one is outside the accepted tolerance one can either decrease H or increase the polynomial degree. For smaller runs on workstations or on mid-sized clusters the latter may be the preferred choice in order to keep the number of macro elements as low as possible. Whereas for large scale simulation on high performance computers, quadratic polynomials seem most suitable. Here, H is decreased almost naturally as the number of cores that are to be used is coupled to the number of macro elements.

4.2.2. Cost for setup

We continue by considering the setup phase. This consists in the computation of the sampling values and the solution of the resulting linear systems for IPOLY or least-squares problems for LSQP. For the IPOLY approach with its small number of

Table 4.3

Number of sampling points $|S_\ell|$ for a single macro element on finest level ($L = 5$), sum of sampling points over all levels, time t_{sample} for stencil computation at sampling points, t_{linalg} for computation of polynomial coefficients, t_{setup} for total setup, t_{cycle} for a single $V(3, 3)$ -cycle, p_{cycle} percentage of t_{setup} w.r.t. t_{cycle} ; for IFEM p_{cycle} relates an IFEM to a LSQP cycle; for LSQP values depend on strategy $m(\ell, j)$ for choosing S_ℓ ; times are given in seconds.

	j	$ S_\ell $	$\sum_{\ell=1}^L S_\ell $	t_{sample}	t_{linalg}	t_{setup}	t_{cycle}	p_{cycle}
IPOLY	–	10	50	0.01	0.001	0.01	16.8	0.1%
	2	455	1,855	0.52	0.02	0.56	16.8	3%
LSQP	$\ell - 1$	39,711	44,731	11.06	0.37	11.53	16.8	67%
	ℓ	333,375	378,071	96.22	4.10	102.66	16.8	611%
IFEM	–	333,375	378,071	–	–	–	872.8	5195%

sampling points and very low dimensional linear system we expect the cost to be negligible. In the LSQP approach, however, the situation might be different as problem size strongly varies with the choice of the sampling set S_ℓ .

To quantify this setup cost we report in Table 4.3 on run-times related to the setup phase and compare them to the execution time for a single $V(3, 3)$ -cycle with our surrogate operator. For this we restrict ourselves to the case of quadratic polynomials and deliberately consider the worst case scenario of a monomial basis. Measurements were performed for a serial run on a single node on the SuperMUC cluster, cf. Sec. 6 for technical details. The code was compiled with the Intel Compiler version 15.0. Linear systems for IPOLY were solved using the DGESV routine of LAPACK, while for the least-squares problems in the LSQP case we used DGELS. In both cases the LAPACK implementation from the Intel Math Kernel Library (MKL) in version 11.3 was employed. The macro mesh consisted of 60 elements, and we used $L = 5$ levels of refinement. Reported values are the minimal ones over a collection of test runs.

Table 4.3 lists the size of the sampling set for the finest level and the total number of sampling points over levels 1 to L for IPOLY and LSQP with different choices for $m(\ell, j)$, see (8). The value t_{sample} represents the time spent with computing stencil weights at the sampling points from the finite element formulation with projected coordinates. This was performed as described in Sec. 4.2.1 for the IFEM approach. The values t_{linalg} and t_{setup} give the time required for the solution of the linear systems resp. least-squares problems and the total time of the setup phase for computing the polynomial coefficients. The last two columns provide information to relate the cost of the setup to those for the evaluation and application of our surrogate operator. t_{cycle} represents the run-time for a single $V(3, 3)$ -cycle using, in the case of IPOLY and LSQP, the surrogate operator and p_{cycle} gives the percentage of t_{setup} with respect to t_{cycle} .

For comparison the last row shows for the IFEM approach the number of sampling points, the cost for a $V(3, 3)$ -cycle using this operator and percentage of these cost compared to t_{cycle} of the surrogate operator. Remember that in this approach there is no setup phase, but instead the stencils are re-computed on-the-fly for each stencil application. This must be done for every node which corresponds to LSQP using $j = \ell$.

Examining the results we see that, as expected, the cost for IPOLY is marginal, only 0.1% of the time required for a single $V(3, 3)$ -cycle. This changes dramatically, when we employ for LSQP all available points \mathcal{M}_ℓ on level ℓ as sampling points, i.e., we choose $j = \ell$ for $m(\ell, j)$. However, already for $j = (\ell - 1)$ this reduces to 67%, i.e., two thirds of one multigrid cycle. For $m(\ell, j) \equiv 2$ this reduces further to only 3% and this choice seems to be the optimal trade-off between accuracy and cost. We point out that the relative cost is measured with respect to one MG-cycle based on the surrogate operator evaluation.

Note that for all choices in the LSQP case the majority, between 90 and 95%, of the time for the setup phase is spent in the evaluation of the stencil weights at the sampling points and only the remaining fraction is required for solving the least-squares problems.

Our previous accuracy experiments had shown that even using a sampling set S_ℓ as coarse as M_2 can give satisfactory results and thus the relative setup cost for our surrogate approaches is quite small compared to the total cost of the solver.

5. Kernel optimization

In Sec. 4, the run-time t_{cycle} of a single V-cycle was measured in a serial setting. Comparing the measured 16.8 seconds to the 5.0 seconds required for a V-cycle with a constant stencil application shows that our straightforward implementation based on incremental polynomial evaluation underperforms. We exceed the theoretically expected cost factor given in (12) by roughly a factor of 1.68.

Thus we now focus on a node-level performance analysis and a more sophisticated re-implementation of the surrogate polynomial application. As we are mainly interested in extreme scale simulations, this will be done for the quadratic case.

In the following, we study the optimization of the GS smoother in detail. Updating the value of the current iterative solution at node I can be formulated as

$$\mathbf{u}_I = \frac{1}{s_{mc}^I} \left(\mathbf{f}_I - \sum_{w \in \mathcal{W} \setminus \{mc\}} s_w^I \mathbf{u}_{I_w} \right) \tag{13}$$

and, naturally, involves all 15 stencil weights s_w^I . Here I_w stands for the node which is the closest one to I when we move in direction w . We will refer to this as a *stencil-based update (SUP)* below.

Our Gauss–Seidel smoother uses a lexicographic ordering. Within HHG this implies that the function values \mathbf{u}_I are updated in the following ordering, as shown in Fig. 2.4. We traverse the nodes of a volume primitive from west to east, going line by line from south to north and plane by plane from bottom to top. Hence, when updating the value at node I , the values at the neighboring nodes with indices bc , be , bnw , bn , ms , mse , and mw have already been updated during the current sweep, whereas values at nodes with indices me , mnw , mn , ts , tse , tw , and tc still need to be updated. Note that due to the hybrid nature of the smoother this is true for the majority of, but not for all nodes, as those near the boundary couple to lower dimensional primitives. Consequently, the only data dependency which must be considered during the update inside a line is $\mathbf{u}_{I_{mw}}$. This value is only available for updating \mathbf{u}_I after the update at the previous node, i.e., I_{mw} , was computed. We further remark that due to the lexicographic ordering, the tc value is the only one that has not been touched in a previous update. Hence, it has not been loaded into the cache so far.

Our analysis revealed that the recursive dependency in (13) dominates the execution time for the update. Furthermore, it hinders certain optimizations, like vectorization, and thereby prohibits exploiting the full floating point performance of the core.

To circumvent this problem, the update formula (13) for all nodes within one line of the volume primitive is re-arranged. For this, we split it into a non-recursive and a recursive part $\mathbf{u}_I = \tau + \rho$ with

$$\tau = \frac{1}{s_{mc}^I} \left(\mathbf{f}_I - \sum_{\substack{w \in \mathcal{W} \\ w \notin \{mc, mw\}}} s_w^I \mathbf{u}_{I_w} \right) \quad (14)$$

$$\rho = -\frac{1}{s_{mc}^I} s_{mw}^I \mathbf{u}_{I_{mw}}. \quad (15)$$

Firstly, the non-recursive part τ is pre-computed and stored in a temporary array, `tmp` in Algorithm 1 below, large enough to hold the results for a complete line. As an incremental polynomial evaluation does not allow complete vectorization, we instead represent the stencil weights s_w^d for $d \in \mathbb{N}_0$ as

$$s_w^d = \hat{p}_w(0) + d \delta \hat{p}_w(0) + \frac{(d-1)d}{2} \delta k_w \quad (16)$$

with $\hat{p}_w(0) := \mathcal{P}^\alpha s_w(0, j_0, k_0)$ and

$$\delta \hat{p}_w(0) := \left. \frac{d \mathcal{P}^\alpha s_w(i, j_0, k_0)}{di} \right|_{i=0} + \frac{\delta k_w}{2},$$

$$\delta k_w := \left. \frac{d^2 \mathcal{P}^\alpha s_w(i, j_0, k_0)}{di^2} \right|_{i=0}.$$

Here, the index d denotes the position of a node inside the current line, starting at 0. For the very first line, the values $\hat{p}_w(0)$, $\delta \hat{p}_w(0)$ and δk_w have to be initialized. For all further lines even these values are obtained by updates analogue to (16) in j or k direction, respectively.

For performance reasons, (14) is further split into two loops, where in the first all stencil elements in \mathcal{W}_1 are treated and then those in \mathcal{W}_2 . The two sets are defined as

$$\mathcal{W}_1 = \{me, mnw, mn, ts, tse, tw\},$$

$$\mathcal{W}_2 = \{tc, bc, be, bnw, bn, ms, mse\}.$$

On the tested architecture, see below, this splitting avoids problems with register spilling, see e.g. [13]. The distribution of the sets is based on hardware dependent performance investigations as detailed in Sec. 5.1 and has no geometric meaning.

Now in a second step, the recursive part ρ , see (15), is computed and the stencil update completed by combining it with the pre-computed value τ stored in the temporary array. Unrolling this last loop by a factor of four or eight increases the performance on the evaluated hardware architecture.

The full pseudo code for the optimized GS smoother is displayed in Algorithm 1.

5.1. Single core performance

In the following, we will conduct a detailed performance investigation. This will be based on performance modeling and compared to actual measurements. Note that we do not include the initialization in line 2 of the algorithm and that the setup phase for computing the polynomials representing the weights of our surrogate operator is considered neither in the model nor in the measurements. The latter is justifiable as for reasonable choices of $m(\ell, j)$ the cost is negligible.

Algorithm 1 Optimized GS smoother.

```

1: for line  $(k, j, 0 : n - 1) \in \text{TET}$  do
2:   initialize/update  $s_w^0, \delta \hat{p}_w^0$ 
3:   // setup stencils along line and store in tmp array tmp[d], tmp_mw[d], and tmp_mc[d], except recursive part mw.
4:   for  $d = 0, \dots, n - 1$  do // loop 1: vectorized
5:     tmp[d] =  $f^d$ ; tmp_mc[d] = 0
6:     for  $w_1 \in \mathcal{W}_1$  do // completely unrolled
7:       compute stencil weight  $s_{w_1}$  // evaluate (16)
8:       tmp[d] -=  $s_{w_1}^d \times u_{w_1}^d$ , tmp_mc[d] -=  $s_{w_1}^d$ 
9:     end for
10:  end for
11:  for  $d = 0, \dots, n - 1$  do // loop 2: vectorized
12:    for  $w_2 \in \mathcal{W}_2$  do // completely unrolled
13:      compute stencil weight  $s_{w_2}^d$  // evaluate (16)
14:      tmp[d] -=  $s_{w_2}^d \times u_{w_2}^d$ , tmp_mc[d] -=  $s_{w_2}^d$ 
15:    end for
16:    compute stencil weight  $s_{mw}^d$  // evaluate (16)
17:    tmp_mc[d] -=  $s_{mw}^d$ ,  $s_0 = 1 / \text{tmp\_mc}[d]$ 
18:    tmp_mw[d] =  $s_0 \times s_{mw}^d$ , tmp[d] = tmp[d]  $\times$   $s_0$ 
19:  end for
20:  // compute recursive part to finish application of the stencil(s)
21:  for  $d = 0, \dots, n - 1$  do // loop 3: 4- or 8-way unrolled
22:     $u_{mc}^d = \text{tmp}[d] - \text{tmp\_mw}[d] \times u_{mw}^d$ 
23:  end for
24: end for

```

Table 5.1

Cycle numbers reported by IACA for the three loops normalized to eight stencil-based updates. Given are the maximum values for each execution unit of a certain category. The reported duration of 40 cy from IACA for loop 3 is higher than the occupancy of the execution ports. This is due to the data dependencies, which hinder efficient instruction pipelining.

	max	FP division	FP mul	FP add	load	store
loop 1	22	0	20	20	22	2
loop 2	56	56	32	34	36	4
loop 3	40	0	4	4	12	8

Benchmarked system The execution is modeled for and evaluated on the Haswell based Intel Xeon E5-2695 processor with 14 cores, which was configured in cluster on die mode, i.e. seven cores per NUMA domain. This is the same system as used in the SuperMUC Phase 2 cluster, on which the scaling experiments of Sec. 6 were conducted. The only difference is that the processor here was only clocked at 2.3 GHz instead of 2.6 GHz on the SuperMUC.

One core of the Intel Xeon comprises several execution ports, which can perform: floating point (FP) multiply & FP fused multiply add (FMA), FP addition & FP FMA, FP division, load, and store. Transferring a cache line between L1/L2 cache or L2/L3 cache takes 2 cycles (cy) on Haswell, respectively. As the system sustains 24.2 GiB/s memory bandwidth from one NUMA domain, it takes 5.7 cy at 2.3 GHz to transfer one cache line between L3 cache and memory.

Performance modeling Our performance analysis is based on the Execution–Cache–Memory (ECM) model [22]. In contrast to the simpler Roofline model [39], which only takes into account the (number of) floating point operations performed for a certain amount of data transferred between core and memory, the ECM model considers the complete path of the memory hierarchy including caches. Furthermore the complete execution inside the core is examined, which, besides floating point operations, includes also, e.g., address generation and register spilling. As in the actual memory hierarchy, data is transferred in the granularity of cache lines, the modeling is based on this unit. On the target architecture a cache line comprises 64 bytes, which is for the stencil considered equivalent to eight updates. In the following, the modeling of the execution and the data transfers inside the memory hierarchy are outlined.

Modeling of execution For modeling the execution of the optimized implementation inside the core, the *Intel Architecture Code Analyzer* (IACA) is used, see [24]. The tool generates a scheduling of the instructions over the available execution ports on a certain micro-architecture for a given binary. This was also used to derive the optimal distribution of the two sets \mathcal{W}_1 and \mathcal{W}_2 . Here it is assumed that all data can be fetched from L1 data cache. IACA can operate in two modes: throughput and latency. In throughput mode it is assumed that the considered loop iterations are independent and can overlap, while in latency mode, it is assumed that operations between iterations cannot overlap. Analyzing the three loops in our implementation with throughput mode resulted in the closest agreement of prediction and measurement.

The values reported by IACA for eight stencil-based updates, our base unit from above, on the Haswell micro-architecture are listed in Table 5.1 for the two vectorized loops (loop 1 and 2) as well as the loop containing the recursive component (loop 3). To ease the modeling process the initialization and setup of the stencil weights in line 2 of Algorithm 1 are not considered. Whereas loop 1 and 2 are limited by the occupancy of the execution ports, IACA reports for loop 3 as bottleneck

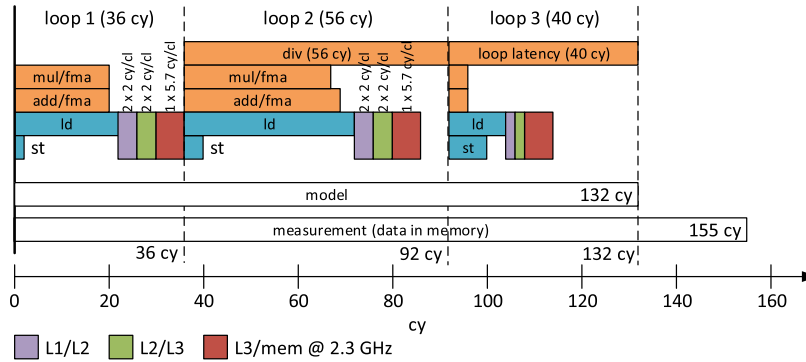


Fig. 5.1. Duration of the different phases involved during eight stencil-based updates as obtained by the ECM model. Shown are the durations of executing loop 1, 2, and 3, as well as the time spend with cache line transfers inside the memory hierarchy. For the loops the occupancy of the different execution ports are shown: multiplication/fused multiply add (mul/fma), addition/fused multiply add (add/fma), division (div), loads (ld), stores (st). Considered in neither model nor measurement is the setup of the initial stencil weights.

the *loop latency* with 40 cy. This indicates that due to data dependencies resulting from the recursion in loop 3 instructions cannot be fully pipelined and stall until their corresponding operands become available. A more detailed overview is presented in Fig. 5.1 which displays the occupancy of the execution units.

Modeling of cache and memory transfers The amount and duration of cache line transfers inside the memory hierarchy for eight stencil-based updates beyond L1 data cache are modeled manually. Those numbers depend on the refinement level ℓ of the macro tetrahedron, as it determines how many data items can be re-used from cache. If two planes of the tetrahedron fit concurrently into a cache level, i.e., the so called *layer condition* [37] is fulfilled, then only the leading stencil neighbor u_{tc}^l must be loaded from a higher cache level, the last stencil neighbor u_{bc}^l gets evicted, and the remaining neighbors can be fetched from this cache level. In our case, we focus on $\ell = 5$ and for simplicity consider the case where the layer condition is only fulfilled in L3 cache. For L1 and L2 cache we assume, that in each cache level only six rows of the tetrahedron, i.e., two rows from top, middle, and bottom plane each, can be kept concurrently.

With this assumptions between L1/L2 cache and L2/L3 cache five cache lines are transferred, respectively: one cache line from each plane containing u_{tc}^l , u_{mn}^l , and u_{bn}^l is loaded (three loads), one cache line containing the corresponding element from the f array (one load), and the modified cache line containing u_{mc}^l gets evicted (one store). As in L3 cache the layer condition is fulfilled, instead of three cache lines only the one for the top plane containing u_{tc}^l must be loaded (one load). Everything else stays the same, which results in three cache line transfers. Concerning the arrays used for vectorization to store the temporary results, they are small enough to fit with the other data into L1 cache. Further the *least recently used* cache policy, commonly used on current architectures, ensures that they get not evicted.

Results The timings resulting from the IACA analysis combined with the manual modeling of the data transfers in the memory hierarchy are summarized in Fig. 5.1. Note that this does not show an exact scheduling of the different execution phases and only sums up the duration of the different parts, which are involved during execution. The predicted duration of 132 cy for eight stencil-based updates by the ECM model is $\approx 15\%$ off from the measurement of the executed kernel.

5.2. Socket performance

When increasing the core count the ECM model assumes that only shared resources can become a bottleneck. In the case of the benchmarked Haswell system this concerns the segmented L3 cache and the memory interface. However, the bandwidth of the L3 cache scales linearly with the number of cores. Thus, only the memory interface could limit performance. Consequently the performance should increase linearly with the number of cores up to the point where the memory interface becomes continuously occupied. The model prediction and the corresponding measurement are both shown in Fig. 5.2. The code does not saturate the memory bandwidth (green line with triangles) completely. This upper limit is given as ratio of the achievable memory bandwidth of 24.2 GiB/s with the bytes transferred between L3 and memory required for one stencil-based update. If the layer condition is fulfilled in L3 cache one stencil-based update requires 24 bytes: load of u_{tc}^l , load of one element from f array, and eviction of updated u_{mc}^l . Utilizing also the SMT thread of each core (single filled red square) increases performance slightly. This has two reasons. Firstly the memory interface is not continuously utilized and secondly still execution resources on each core are available. The resources are available because of pipeline bubbles, i.e., time slots with no operation in the execution pipeline, due to the division in loop 2 and the recursion in loop 3.

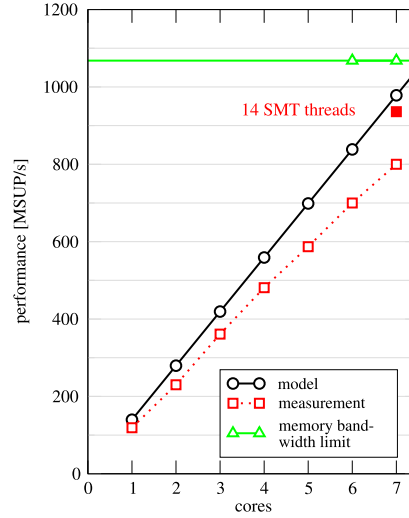


Fig. 5.2. Performance of the optimized code (in MSUP/s, mega stencil-based updates per second) compared with the ECM model prediction.

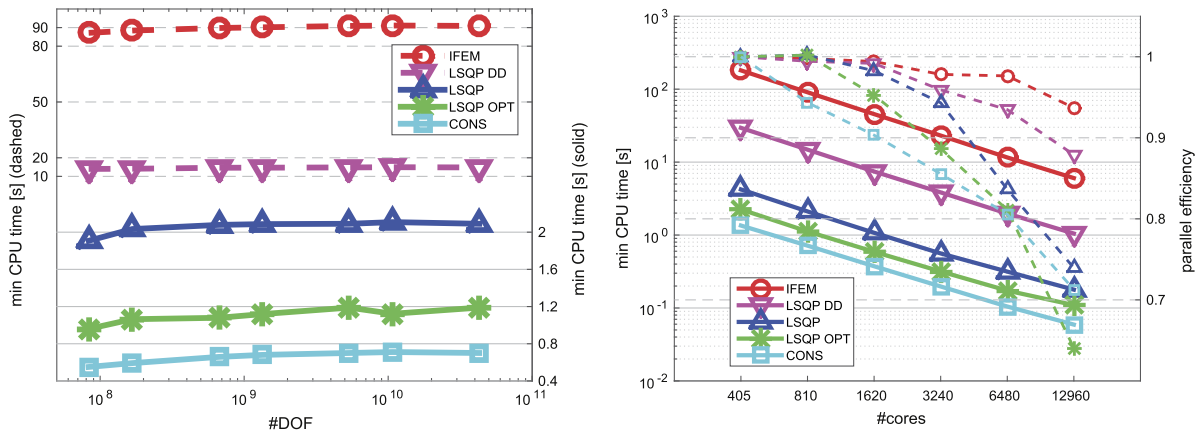


Fig. 6.1. Run-times of one $V(3, 3)$ -cycle. (Left) Weak scaling on SuperMUC. 16 macro elements are assigned to one core. The finest grid is fixed to $L = 5$, i.e. there are $3.3 \cdot 10^5$ DOFs per volume primitive. (Right) Strong scaling (solid) and parallel efficiency (dashed). The problem size is fixed to 12,960 macro elements and $L = 5$. In total there are $4.5 \cdot 10^9$ DOFs.

6. Weak and strong scaling

To demonstrate the performance and scalability of our approach, we perform weak and strong scaling runs on the supercomputer SuperMUC (Phase 2) that is ranked number 28 in the June 2016 TOP500 list. It is a Lenovo NeXTScale system with a theoretical peak performance of 3.58 PFLOPs/s. Each compute node consists of two Haswell Xeon E5-2697 v3 processors where each processor is equipped with 14 cores running at 2.6GHz. Per core 2.3 GiB of memory are provided, but typically only 2.1 GiB are available for applications. The nodes are connected via an Infiniband (FDR14) non-blocking tree.

For our runs we utilize the Intel C/C++ Compiler 15.0 with flags `-O3 -CORE-AVX2 -fma` and the Intel MPI library 5.0.

As mentioned before degree two polynomials for the IPOLY/LSQP approach provide the most efficient compromise between accuracy and computational cost in the HPC context. In Fig. 6.1 we present the run-times of one $V(3, 3)$ -cycle for the different implementations. For reproducibility we take the minimum value out of ten cycles reflecting the best exploitation of the machine’s capabilities. We note that these minimal values are robust. For each of the 65 runs performed, 35 for weak scaling and 30 for strong scaling, at least six of the ten V -cycle run-times, and overall more than 90% are within a 10% deviation from the respective minimum value.

The CONS setting serves as minimal cost reference. IPOLY and LSQP differ only in the setup phase, so only LSQP is considered here. For this approach, we compare the straightforward implementation of the method with the optimized version from Sec. 5 termed LSQP OPT. To demonstrate the cost of a conventional assembly of the stencils also the IFEM case is shown. Furthermore the run-times for double discretization are given. Here the non-optimized LSQP implementation is

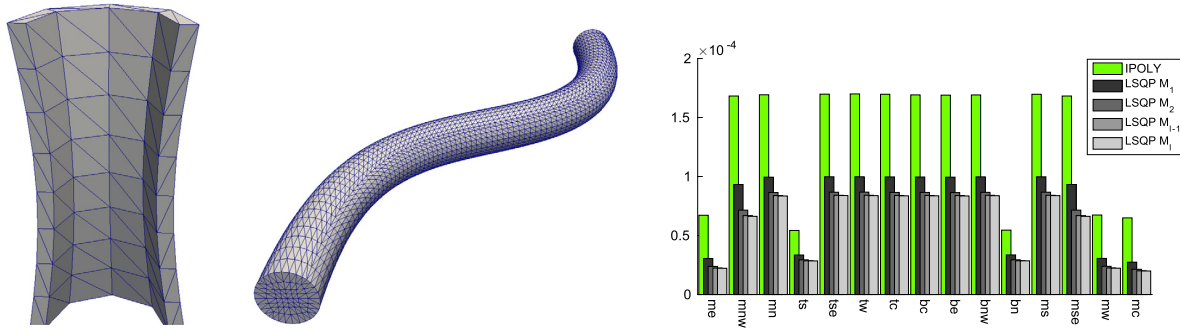


Fig. 7.1. Macro mesh for the half cylinder mantle (left) and the curved pipe (center). On the right the relative mean error e_w for one macro element for the cylinder mantle is given for $L = 4$.

chosen for the smoother and the conventional stencil assembly for residual. As this is dominated by the expensive residual, employing the optimized smoother does not lead to a significant improvement.

For the weak scaling, Fig. 6.1(left), we assign 16 macro elements to one core and fix the refinement level to $L = 5$ giving us $3.3 \cdot 10^5$ DOFs per volume primitive. The number of macro elements and cores varies from 240 to 122,880 and from 15 to 7,680, respectively.

We observe that the IFEM method is even more expensive than the theoretical prediction from Table 4.2 indicated. This is due to additional memory accesses and function calls which are not considered in the flop count. The double discretization is about a factor of 7 slower than the non-optimized LSQP method. This is in line with the flop count for a $V(3, 3)$ -cycle taking into account the aforementioned additional cost for the residual. The values for the straightforward implementation of the LSQP method reflect the serial run-time factor of $(1.68 \cdot C(2))$ mentioned in Sec. 5. Our optimized version outperforms the cost factor $C(2) \approx 2$, and the run-time increases by only a factor of 1.6 compared to the CONS case. Also it shows excellent weak scaling behavior.

For the strong scaling case, Fig. 6.1(right), we fix the problem size to 12,960 macro elements and set $L = 5$. In total this results in $4.5 \cdot 10^9$ DOFs. This also gives an upper bound for the number of cores that can be used. The lower bound is determined by the available memory of 2.1 GiB per core where we need about 100 bytes per DOF in our implementation.

Not surprisingly, the IFEM implementation shows the best parallel efficiency of above 90% even for the largest number of cores. Here, most of the time is spent with the cost intense on-the-fly stencil computation. Thus, increasing the amount of communication by increasing the number of cores does hardly affect the performance that scales almost linearly with the number of cores. This is also the case for the DD approach. Even though the computationally expensive residual occurs a factor of 7 less often than in the IFEM case, it is still the dominating factor. Whereas for the CONS and for the LSQP implementations where the stencil is either already known or computed by an inexpensive evaluation of the surrogate polynomials, we clearly see the influence of communication for the runs with larger number of cores. But nevertheless the run-time even for the largest case of the optimized LSQP method is a factor of 10 or 55 faster than LSQP DD or IFEM, respectively, and only 1.8 times slower than the CONS reference. We point out that for the DD approaches, the expensive operator need only be used in the final few iterations and thus its cost within the solver can be further reduced.

7. Alternative curved geometries

To demonstrate the flexibility and robustness of our approach, we apply it to two alternative curved geometries. The first is a half cylinder mantle with inner radius $r_1 = 0.8$ and outer radius $r_2 = 1.0$, height $z_1 = 4.0$ and with an angular coordinate between 0 and π . In axial direction the cylinder mantle is warped inwards by $w(z) = 0.2 \sin(z\pi/z_1)$. The second geometry is a curved pipe with radius one and an overall length of about 28.

For both settings it is easier to define the input mesh \mathcal{T}_{-2} not on the domain itself. Instead we mesh a fitting hexahedron and then apply a global mapping Φ to this mesh, see Appendix A for details. \mathcal{T}_{-2} is then regularly refined and the mapping applied to the new nodes.

Fig. 7.1 depicts the two coarse meshes after application of Φ , i.e. we show the mesh constructed from mapping the vertices of \mathcal{T}_{-2} . Note that these meshes involve thin and stretched macro tetrahedra. Since our multigrid method combines a point-smoother with full coarsening, its algebraic convergence rate will deteriorate. However, this is independent of whether we use IFEM or LSQP, as will be shown below, and could be fixed by standard multigrid techniques, see e.g. [38].

On the right of Fig. 7.1 we show the relative mean errors in the stencil weights for the case of the cylinder mantle and a macro element which was refined six times. Compared to the spherical shell scenario the error is much smaller. This can be attributed to the moderate curvature in the geometry.

For our experiments with the cylinder we employ an input mesh that gives us one hexahedral block of macro elements in radial direction, four in tangential and eight in axial direction, where each of these blocks is decomposed into six tetrahedral macro elements, see the left part of Fig. 7.1. This leads to macro elements of size $H_r = 0.2$, $H_t = 0.79$ and $H_z = 0.5$. For

Table 7.1

Discretization error in the L^2 -norm for IFEM and LSQP ($q = 2$) and ratio of the asymptotic multigrid convergence rates for half cylinder mantle.

Level	e_{IFEM}	e_{LSQP}	$\rho_{\text{IFEM}}/\rho_{\text{LSQP}}$
$L = 1$	7.125e-02	7.125e-02	1.00
$L = 2$	1.749e-02	1.749e-02	1.00
$L = 3$	4.305e-03	4.304e-03	1.00
$L = 4$	1.066e-03	1.066e-03	1.00
$L = 5$	2.653e-04	2.649e-04	1.00
$L = 6$	6.614e-05	6.719e-05	1.00

Table 7.2

Discretization error in the L^2 -norm for IFEM and LSQP ($q = 2$ and $q = 3$) and ratio of the asymptotic multigrid convergence rates for curved pipe.

Level	e_{IFEM}	$e_{\text{LSQP}} (q = 2)$	$\rho_{\text{IFEM}}/\rho_{\text{LSQP}} (q = 2)$	$e_{\text{LSQP}} (q = 3)$	$\rho_{\text{IFEM}}/\rho_{\text{LSQP}} (q = 3)$
$L = 1$	7.50e-05	7.50e-05	1.00	7.50e-05	1.00
$L = 2$	1.86e-05	1.86e-05	1.00	1.86e-05	1.00
$L = 3$	4.64e-06	4.67e-06	1.00	4.64e-06	1.00
$L = 4$	1.16e-06	1.41e-06	1.00	1.16e-06	1.00
$L = 5$	2.89e-07	9.87e-07	1.00	3.10e-07	1.00

testing the accuracy of the finite element solution based on the surrogate operator we use the following analytic solution, which is zero at the inner and outer radial boundary

$$u(x, y, z) = \sin\left(\frac{\sqrt{x^2 + y^2} + w(z) - r_1}{r_2 - r_1}\pi\right) \cos\left(4 \arccos\left(\frac{x}{\sqrt{x^2 + y^2}}\right)\right) \exp(z/2) . \tag{17}$$

For the pipe \mathcal{T}_{-2} is chosen such that along its course the pipe is split into 80 pieces. Each of these resulting disks is further divided into 8×8 blocks. Again, each block consists of 6 tetrahedral macro elements. In total this gives 30,720 macro elements. For the circular shape we use in the interior a squared block of length 0.66 that is composed of $4 \cdot 4 \cdot 6$ macro elements and in the outer region we use two circular layers. In order to get a smooth transition from the inner to outer region the curvature is continuously increased with radius. Note that this setting is comparable to the second column of Table 4.1 where we also used two radial layers to discretize the spherical shell. But this time we have a thicker circular layer with largest radial $H_r \approx 0.37$ which makes this example more challenging. So we expect that the LSQP approach with quadratic polynomials will allow about 3 to 4 levels of refinement. To test this geometry we set the analytic solution

$$u(x, y, z) = \sin(x) \sin(y) \sin(z). \tag{18}$$

with non-homogeneous Dirichlet boundary conditions.

As described in Sec. 4, we use a geometric multigrid as iterative solver in combination with a residual based stopping criterion given by a reduction factor of 10^{-8} . Table 7.1 shows the ratio of ρ_{IFEM} and ρ_{LSQP} , i.e. the convergence rates of the multigrid solver for both approaches. This was measured as $\rho = (r^{(10)}/r^{(5)})^{1/5}$ where $r^{(j)}$ denotes the norm of residual at iteration j . We see that, indeed, the convergence rate is not affected by choosing the surrogate operator. This is also reflected by the fact that for both discretizations the multigrid solver always requires the same number of V-cycle steps to satisfy the stopping criterion. Here, as in Sec. 6, the run-time required for LSQP was much smaller than that for IFEM.

The discretization error is evaluated with respect to the exact solution in the L^2 -norm for both discretizations and geometries. In Table 7.1, we compare the discretization errors for IFEM and LSQP. Even for the rather coarse mesh of the half cylinder mantle with anisotropic elements we observe no loss of accuracy at all for the first five levels. On level 6 the accuracy loss is smaller than 2%.

The second geometry is more challenging. On the one hand, it has a stronger curvature along its course. On the other the change of curvature within a mapped macro element is much stronger and increases in radial direction. To account for these additional complexities we had to increase the sampling set to $S_\ell = M_\ell$. Indeed, as predicted above, we find that our approach is exact almost up to $L = 4$ where the accuracy loss is about 20%, see Table 7.2. Even for cases $L \geq 4$ where the quadratic surrogate operator converges to a slightly different solution, the algebraic convergence of the multigrid solver is not affected. When we increase the polynomial degree of the surrogate operator, we can restore the exact discretization error with a relative deviation of less than 8% even for $L = 5$.

8. Conclusion and outlook

We introduced a novel two-scale approach for handling non-polyhedral domains in large scale simulations with matrix-free finite elements. It is based on replacing the cost intensive on-the-fly computations of the node stencil by the evaluation

of a surrogate approximation polynomial of low order. Typically for large system size, second order polynomials already guarantee high enough accuracy and thus reduce the flop counts significantly. This technique can be combined for a further improvement of the numerical accuracy with the classical double discretization approach. Here the smoother in the multi-grid algorithm employs cheap lower order stencils while the expensive high accuracy operators are only used to compute the residual for the coarse grid correction.

The accuracy of our approach was first examined theoretically by an a priori estimate and secondly verified by numerical experiments. The implementation was further optimized on the node-level and systematically analyzed by the ECM model. Excellent performance and scalability was demonstrated on the supercomputer SuperMUC. The approach is completely local with respect to the macro elements and open to general blending functions and geometries. Future work will include different partial differential equations such as the Stokes system, variable coefficients in the PDE and alternative stencil approximations preserving symmetry for the \mathcal{L} operator.

Acknowledgements

This work was partly supported by the German Research Foundation through the Priority Programme 1648 “Software for Exascale Computing” (SPPEXA) and by WO671/11-1. The authors gratefully acknowledge the Gauss Centre for Supercomputing (GCS) for providing computing time on the supercomputer SuperMUC at Leibniz-Rechenzentrum (LRZ).

Appendix A. Blending functions

In this appendix we give details of the initial macro mesh and blending functions Φ used for our three test geometries.

A.1. Spherical shell

The initial macro mesh \mathcal{T}_{-2} is generated using an icosahedral approach well-known in meteorology and geodynamics. It is based on an icosahedral triangulation of the unit sphere, see e.g. [6,36], that is then mapped onto corresponding radial sub-layers of the thick spherical shell. This results in a mesh composed of prismatic elements consisting of two parallel planar triangles with vertices on neighboring spherical layers of the mesh. Their vertices are connected by radial grid lines \mathcal{R} . For our purposes we split each prism into three tetrahedral macro elements. All three macro elements T generated from the same prism \mathcal{P} use an identical blending function Φ_T . The mapping itself is given by

$$\Phi_T(\underline{x}) = \frac{\underline{x}}{|\underline{x}|} R(\underline{x}) . \quad (\text{A.1})$$

The corresponding scaling factor $R(\underline{x})$ is fixed in the following fashion. Each point \underline{x} lies in a triangle parallel to the top and bottom one of \mathcal{P} . The intersection of this triangle with the radial lines \mathcal{R} determines $R(\underline{x})$.

A.2. Half cylinder mantle

For this geometry we start with an initial triangulation \mathcal{T}_{-2} of the hexahedron $[r_1, r_2] \times [0, 1/2] \times [0, z_1]$. Note that these are the (scaled) cylindrical coordinates of the cylinder mantle. In this setting Φ can be defined globally by

$$\Phi(x, y, z) = \begin{pmatrix} (x - w(z)) \cos(2\pi y) \\ (x - w(z)) \sin(2\pi y) \\ z \end{pmatrix} \quad (\text{A.2})$$

with $w(z) = 0.2 \sin(\pi z/z_1)$ prescribing the curvature along z-direction.

A.3. Curved pipe

Here we start with an initial triangulation \mathcal{T}_{-2} of the hexahedron given by $[-\sqrt{2}/2, \sqrt{2}/2] \times [-\sqrt{2}/2, \sqrt{2}/2] \times [0, z_1]$ with $z_1 = 20$. In a first step this domain is mapped onto a cylinder by

$$\varphi(x, y, z) = \begin{pmatrix} \varphi^x(x, y) \\ \varphi^y(x, y) \\ z \end{pmatrix} =: \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} \quad (\text{A.3})$$

To this end, we divide the square $[-\sqrt{2}/2, \sqrt{2}/2]^2$ into an interior square Ω_1 of size $[-1/3, 1/3]^2$ and four outer trapezoidal regions Ω_2 to Ω_5 . The top trapezoid is Ω_2 and the others are numbered clock-wise. Then, we define on each region the mapping $\varphi_i(x, y)$, $1 \leq i \leq 5$, as:

$$\varphi_1(x, y) = \begin{pmatrix} x \\ y \end{pmatrix} \quad (\text{A.4})$$

$$\varphi_2(x, y) = \frac{y - \sqrt{2}/2}{1/3 - \sqrt{2}/2} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{y - 1/3}{\sqrt{2}/2 - 1/3} \frac{\sqrt{2}y}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix} \quad (\text{A.5})$$

$$\varphi_3(x, y) = \frac{x - \sqrt{2}/2}{1/3 - \sqrt{2}/2} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{x - 1/3}{\sqrt{2}/2 - 1/3} \frac{\sqrt{2}x}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix} \quad (\text{A.6})$$

$$\varphi_4(x, y) = \frac{y + \sqrt{2}/2}{-1/3 + \sqrt{2}/2} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{y + 1/3}{-\sqrt{2}/2 + 1/3} \frac{-\sqrt{2}y}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix} \quad (\text{A.7})$$

$$\varphi_5(x, y) = \frac{x + \sqrt{2}/2}{-1/3 + \sqrt{2}/2} \begin{pmatrix} x \\ y \end{pmatrix} + \frac{x + 1/3}{-\sqrt{2}/2 + 1/3} \frac{-\sqrt{2}x}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix} \quad (\text{A.8})$$

This gives a smooth transition from the inner block Ω_1 where no transformation is done, to the outer blocks where the curvature is increased with $|y|$ for $\Omega_{2,4}$ or $|x|$ for $\Omega_{3,5}$, respectively.

In a second step we add the curvature along the pipe. The shape of the curved pipe is prescribed by the translation $T(\hat{z}) = (f(\hat{z}), 0, 0)^T$ where $f(\hat{z}) = (2\hat{z} - z_1)^3/1000$. Additionally, for each \hat{z} the corresponding disk of the cylinder in x, y -plane is rotated such that it is perpendicular to the plane with normal vector $1/\mu (f'(\hat{z}), 0, 1)^T$, with the scaling factor $\mu = 1/\sqrt{f'(\hat{z})^2 + 1}$. This can be described by $M(\hat{z}) (\hat{x}, \hat{y}, 0)^T + (0, 0, \hat{z})^T$ with a rotation matrix

$$M(\hat{z}) = \frac{1}{\mu} \begin{bmatrix} 1 & 0 & f'(\hat{z}) \\ 0 & \mu & 0 \\ -f'(\hat{z}) & 0 & 1 \end{bmatrix}$$

Finally, the global mapping Φ is given by

$$\Phi(x, y, z) = M(z)\varphi(x, y, 0) + T(z) + (0, 0, z)^T. \quad (\text{A.9})$$

References

- [1] P. Arbenz, G.H. van Lenthe, U. Mennel, R. Müller, M. Sala, A scalable multi-level preconditioner for matrix-free μ -finite element analysis of human bone structures, *Int. J. Numer. Methods Eng.* 73 (2008) 927–947.
- [2] A. Baker, R. Falgout, T. Kolev, U.M. Yang, *High-Performance Scientific Computing – Algorithms and Applications*, Springer, 2012, pp. 261–279.
- [3] A.H. Baker, A. Klawonn, T. Kolev, M. Lanser, O. Rheinbach, U.M. Yang, Scalability of classical algebraic multigrid for elasticity to half a million parallel tasks, in: *Software for Exascale Computing – SPPEXA 2013–2015*, in: *Lecture Notes in Computational Science and Engineering*, vol. 113, Springer, 2016, pp. 113–140.
- [4] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J.J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, 1994.
- [5] P. Bastian, C. Engwer, D. Göddeke, O. Iliev, O. Ippisch, M. Ohlberger, S. Turek, J. Fahlke, S. Kaulmann, S. Müthing, D. Ribbrock, Exa-dune: flexible PDE solvers, methods and applications, in: *Euro-Par 2014: Parallel Processing Workshops: Euro-Par 2014 International Workshops*, Porto, Portugal, August 25–26, 2014, Springer International Publishing, Cham, 2014, pp. 530–541, *Revised Selected Papers, Part II*.
- [6] J.R. Baumgardner, P.O. Frederickson, Icosahedral discretization of the two-sphere, *SIAM J. Numer. Anal.* 22 (1985) 1107–1115.
- [7] Y. Bazilevs, K. Takizawa, T. Tezduyar, *Computational Fluid-Structure Interaction: Methods and Applications*, John Wiley & Sons, Ltd., 2013.
- [8] B. Bergen, *Hierarchical Hybrid Grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*, Ph.D. thesis, Technische Fakultät der Friedrich-Alexander-Universität Erlangen, Nürnberg, 2005.
- [9] B. Bergen, F. Hülsemann, Hierarchical hybrid grids: data structures and core algorithms for multigrid, *Numer. Linear Algebra Appl.* 11 (2004) 279–291.
- [10] J. Bey, Tetrahedral grid refinement, *Computing* 55 (1995) 355–378.
- [11] A. Bienz, R. Falgout, W. Gropp, L. Olson, J. Schroder, Reducing parallel communication in algebraic multigrid through sparsification, *SIAM J. Sci. Comput.* 38 (2016) S332–S357.
- [12] A. Brandt, O. Livne, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, revised edition, *Classics in Applied Mathematics*, Society for Industrial and Applied Mathematics, 2011.
- [13] D.E. Comer, *Essentials of Computer Architecture*, Pearson Prentice Hall, New Jersey, 2005.
- [14] J. Cottrell, T. Hughes, Y. Bazilevs, *Isogeometric Analysis: Towards Integration of CAD and FEA*, John Wiley & Sons, Ltd., 2009.
- [15] S. Dalton, L. Olson, N. Bell, Optimizing sparse matrix–matrix multiplication for the GPU, *ACM Trans. Math. Softw.* 41 (2015) 25:1–25:20.
- [16] C.C. Douglas, J. Hu, M. Kowarschik, U. Rüde, C. Weiß, Cache optimization for structured and unstructured grid multigrid, *Electron. Trans. Numer. Anal.* 10 (2000) 21–40.
- [17] R. Falgout, U. Meier-Yang, hypre: a library of high performance preconditioners, *Comput. Sci.-ICCS 2002 (2002)* 632–641.
- [18] B. Flemisch, J.M. Melenk, B.I. Wohlmuth, Mortar methods with curved interfaces, *Appl. Numer. Math.* 54 (2005) 339–361.
- [19] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, B. Wohlmuth, Performance and scalability of hierarchical hybrid multigrid solvers for Stokes systems, *SIAM J. Sci. Comput.* 37 (2015) C143–C168.
- [20] D. Guo, W. Gropp, L.N. Olson, A hybrid format for better performance of sparse matrix–vector multiplication on a GPU, *Int. J. High Perform. Comput. Appl.* 30 (2016) 103–120.
- [21] W. Hackbusch, On multi-grid iterations with defect correction, in: W. Hackbusch, U. Trottenberg (Eds.), *Multigrid Methods: Proceedings of the Conference Held at Köln–Porz*, in: *Lecture Notes in Mathematics*, vol. 960, Springer, 1982, pp. 461–473.
- [22] G. Hager, J. Treibig, J. Habich, G. Wellein, Exploring performance and power properties of modern multicore chips via simple machine models, *Concurr. Comput.: Practice Exp.* (2014).
- [23] H. Igel, *Computational Seismology: A Practical Introduction*, Oxford University Press, 2016.

- [24] Intel Corp., Intel architecture code analyzer, <http://software.intel.com/en-us/articles/intel-architecture-code-analyzer>, 2012, Version: 2.1.
- [25] B.L.N. Kennett, H.P. Bunge, *Geophysical Continua*, Cambridge University Press, 2008.
- [26] M. Kowarschik, U. Rude, C. Wei, Data layout optimizations for variable coefficient multigrid, in: *International Conference on Computational Science*, in: *Lecture Notes in Computer Science*, vol. 2331, Springer, 2002, pp. 642–651.
- [27] M. Kreutzer, G. Hager, G. Wellein, H. Fehske, A. Bishop, A unified sparse matrix data format for efficient general sparse matrix–vector multiplication on modern processors with wide SIMD units, *SIAM J. Sci. Comput.* 36 (2014) C401–C423.
- [28] M. Kronbichler, K. Kormann, A generic interface for parallel cell-based finite element operator application, *Comput. Fluids* 63 (2012) 135–147.
- [29] A. Logg, K.B. Ølgaard, M.E. Rognes, G.N. Wells, FFC: the FEniCS form compiler, in: A. Logg, K.A. Mardal, G.N. Wells (Eds.), *Automated Solution of Differential Equations by the Finite Element Method*, in: *Lecture Notes in Computational Science and Engineering*, vol. 84, Springer, 2012, pp. 227–238.
- [30] D.A. May, J. Brown, L.L. Pourhiet, A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow, *Comput. Methods Appl. Mech. Eng.* 290 (2015) 496–523.
- [31] Y. Notay, A. Napov, A massively parallel solver for discrete Poisson-like problems, *J. Comput. Phys.* 281 (2015) 237–250.
- [32] A. Rappoport, Rendering curves and surfaces with hybrid subdivision and forward differencing, *ACM Trans. Graph.* 10 (1991) 323–341.
- [33] B. van Rietbergen, H. Weinans, R. Huiskes, B. Polman, Computational strategies for iterative solutions of large fem applications employing voxel data, *Int. J. Numer. Methods Eng.* (1996) 2743–2767.
- [34] A.P. Rockwood, Generalized scanning technique for display of parametrically defined surfaces, *IEEE Comput. Graph. Appl.* 7 (1987) 15–26.
- [35] J. Rudi, A.C.I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P.W.J. Staar, Y. Ineichen, C. Bekas, A. Curioni, O. Ghattas, An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in Earth’s mantle, in: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC ’15*, ACM, New York, NY, USA, 2015, pp. 5:1–5:12.
- [36] A. Staniforth, J. Thuburn, Horizontal grids for global weather and climate prediction models: a review, *Q. J. R. Meteorol. Soc.* 138 (2012) 1–26.
- [37] H. Stengel, J. Treibig, G. Hager, G. Wellein, Quantifying performance bottlenecks of stencil computations using the execution–cache–memory model, in: *Proceedings of the 29th International Conference on Supercomputing, ICS ’15*, ACM, New York, NY, USA, 2015, pp. 207–216.
- [38] U. Trottenberg, C. Oosterlee, A. Schller, *Multigrid*, ISBN 0-12-701070-X, Academic Press, 2001.
- [39] S. Williams, A. Waterman, D. Patterson, Roofline: an insightful visual performance model for multicore architectures, *Commun. ACM* 52 (2009) 65–76.

Chapter 4

A new matrix-free approach for large-scale geodynamic simulations and its performance

This chapter was published in the book series *Lecture Notes in Computer Science* in 2018 and reprinted by permission from Springer Customer Service Centre GmbH:

Springer Nature, Lecture Notes in Computer Science, vol 10861

A new matrix-free approach for large-scale geodynamic simulations and its performance, Simon Bauer, Markus Huber, Marcus Mohr, Ulrich Rüde, Barbara Wohlmuth.


© Springer International Publishing AG, part of Springer Nature, 2018

https://doi.org/10.1007/978-3-319-93701-4_2

In this chapter the LSQP method from chapter 3 will be extended to the full Stokes system including variable viscosity coefficients. The method will be applied to a large scale geophysical application with a global resolution of 1.7 km.



A New Matrix-Free Approach for Large-Scale Geodynamic Simulations and its Performance

Simon Bauer¹, Markus Huber², Marcus Mohr¹ , Ulrich Rüde^{3,4},
and Barbara Wohlmuth²

¹ Department of Earth and Environmental Sciences,
Ludwig-Maximilians-Universität München, Munich, Germany
{simon.bauer,marcus.mohr}@lmu.de

² Institute for Numerical Mathematics (M2), Technische Universität München,
Munich, Germany

³ Department of Computer Science 10, FAU Erlangen-Nürnberg,
Erlangen, Germany

⁴ Parallel Algorithms Project, CERFACS, Toulouse, France

Abstract. We report on a two-scale approach for efficient matrix-free finite element simulations. The proposed method is based on surrogate element matrices constructed by low-order polynomial approximations. It is applied to a Stokes-type PDE system with variable viscosity as is a key component in mantle convection models. We set the ground for a rigorous performance analysis inspired by the concept of parallel textbook multigrid efficiency and study the weak scaling behavior on SuperMUC, a peta-scale supercomputer system. For a complex geodynamical model, we achieve a parallel efficiency of 95% on up to 47 250 compute cores. Our largest simulation uses a trillion ($\mathcal{O}(10^{12})$) degrees of freedom for a global mesh resolution of 1.7 km.

Keywords: Two-scale PDE discretization
Massively parallel multigrid · Matrix-free on-the-fly assembly
Large scale geophysical application

1 Introduction

The surface of our planet is shaped by processes deep beneath our feet. Phenomena like earthquakes, plate tectonics, crustal evolution up to the geodynamo are governed by forces in the Earth's mantle that transport heat from the interior of our planet to the surface in a planetwide solid-state convection. For this reason, the study of the dynamics of the mantle is critical to our understanding of how the entire planet works.

There is a constant demand for ever more realistic models. In the case of mantle convection models (MCMs), this includes, e.g., compressible flow formulations, strongly non-linear rheologies, i.e., models in which the fluid viscosity

depends not only on pressure and temperature, but also on the flow velocity, the inclusion of phase transitions or the tracking of chemical composition. A discussion of current challenges is, e.g., given in [15]. Another trend is the growing use of MCMs to perform inverse computations via adjoint techniques in order to link uncertain geodynamic modeling parameters to geologic observables and, thus, improve our understanding of mantle processes, see e.g. [7]. These advanced models require efficient software frameworks that allow for high spatial resolutions and combine sophisticated numerical algorithms with excellent parallel efficiency on supercomputers to provide fast time-to-solution. See [11, 15, 21] for recent developments.

We will focus here on the most compute-intensive part of any MCM, which is the solution of the generalized Stokes problem, where \mathbf{f} represents the buoyancy forces, \mathbf{u} velocity, p pressure, T temperature and $\nu(\mathbf{u}, T)$ is the viscosity of the mantle.

$$-\operatorname{div} \left[\frac{1}{2} \nu (\nabla \mathbf{u} + (\nabla \mathbf{u})^\top) \right] + \nabla p = \mathbf{f}, \quad \operatorname{div} \mathbf{u} = 0. \quad (1)$$

Problem (1) needs to be solved repeatedly as part of the time-stepping and/or as part of a non-linear iteration, if ν depends on \mathbf{u} . Note that in (1) we assume an incompressible fluid, as the best way to treat the compressibility of the mantle is an open question, [15], outside the scope of this contribution.

Most current global convection codes are based on finite element (FE) discretizations, cf. [8, 15, 21]. While traditional FE implementations are based on the assembly of a global system matrix, there is a trend to employ matrix-free techniques, [2, 4, 17, 19]. This is motivated by the fact that storing the global matrix increases the memory consumption by an order of magnitude or more even when sparse matrix formats are used. This limits the resolution and results in a much increased memory traffic when the sparse matrix must be re-read from memory repeatedly. Since the cost for data movement has become a limiting factor for all high performance supercomputer architectures both in terms of compute time and energy consumption, techniques for reducing memory footprint and traffic must receive increased attention in the design of modern numerical methods.

In this contribution, we report on the prototype of a new mantle convection framework that is implemented based on Hierarchical Hybrid Grids (HHG) [1, 4, 11, 14]. HHG employs an unstructured mesh for geometry resolution which is then refined in a regular fashion. The resulting mesh hierarchy is well suited to implement matrix-free geometric multigrid methods. Multigrid techniques play an important role in any large-scale Stokes solver, most commonly as preconditioner for the momentum operator in a Krylov solver, or as inner solver in a Schur complement approach. We employ a geometric Uzawa-type multigrid solver that treats the full Stokes system all-at-once [12]. We present a new approach that allows to assemble the resulting FE stencils in the case of curved geometries and variable viscosity on-the-fly as a core component of matrix-free multigrid solvers. It is based on a polynomial approximation of the local element matrices, extending our work in [2].

We will carry out a systematic performance analysis of our HHG-based implementation and investigate parallel performance with respect to run-time, memory consumption and parallel efficiency of this new numerical approach for a real-world geophysical application. It will be investigated and tuned on the SuperMUC peta-scale system of the Leibniz Supercomputing Center (LRZ).

2 Software Framework and Discretization

Here we consider the thick spherical shell $\Omega = \{\mathbf{x} \in \mathbb{R}^3 : r_{\text{cmb}} < \|\mathbf{x}\|_2 < r_{\text{srf}}\}$, where r_{cmb} and r_{srf} correspond to the inner and outer mantle boundary, and $\|\cdot\|_2$ denotes the Euclidean norm of a vector. By taking the Earth radius as reference unit, we set $r_{\text{cmb}} = 0.55$ and $r_{\text{srf}} = 1$. We discretize Ω by an initial tetrahedral mesh \mathcal{T}_0 using a standard icosahedral meshing approach for spherical shells, see e.g. [8]. From this we construct a family of semistructured meshes $\mathcal{T} := \{\mathcal{T}_\ell, \ell = 0, \dots, L\}$ by uniform refinement up to level $L \in \mathbb{N}_0$. For the finite element discretization of the Stokes system (1), we employ standard conforming linear finite element spaces for velocity and pressure on \mathcal{T} . While this P^1 – P^1 pairing is of computational interest, it is known to be unstable. We use the pressure stabilization Petrov-Galerkin (PSPG) method [6] as stabilization technique. Using standard nodal basis functions for the finite element spaces, we obtain on each level ℓ of the hierarchy a linear system of algebraic equations

$$\mathcal{L}_\ell \begin{pmatrix} \mathbf{u}_\ell \\ \mathbf{p}_\ell \end{pmatrix} := \begin{pmatrix} A_\ell & G_\ell \\ D_\ell & -C_\ell \end{pmatrix} \begin{pmatrix} \mathbf{u}_\ell \\ \mathbf{p}_\ell \end{pmatrix} = \begin{pmatrix} \mathbf{f}_\ell \\ \mathbf{g}_\ell \end{pmatrix}, \quad \ell = 0, \dots, L, \quad (2)$$

where $\mathbf{u}_\ell \in \mathbb{R}^{n_{\mathbf{u};\ell}}$ and $\mathbf{p}_\ell \in \mathbb{R}^{n_{\mathbf{p};\ell}}$. The dimensions of the velocity and the pressure space are denoted by $n_{\mathbf{u};\ell}$ and $n_{\mathbf{p};\ell}$. For our considerations below, it is advantageous to re-write (2) by sorting the vector of unknowns with respect to the different types of degrees of freedom to expose the scalar building blocks of (2)

$$\mathcal{L}_\ell = \begin{pmatrix} A_\ell^{11} & A_\ell^{12} & A_\ell^{13} & G_\ell^1 \\ A_\ell^{21} & A_\ell^{22} & A_\ell^{23} & G_\ell^2 \\ A_\ell^{31} & A_\ell^{32} & A_\ell^{33} & G_\ell^3 \\ D_\ell^1 & D_\ell^2 & D_\ell^3 & -C_\ell \end{pmatrix}, \quad \begin{pmatrix} \mathbf{u}_\ell \\ \mathbf{p}_\ell \end{pmatrix} = \begin{pmatrix} \mathbf{u}_\ell^1 \\ \mathbf{u}_\ell^2 \\ \mathbf{u}_\ell^3 \\ \mathbf{p}_\ell \end{pmatrix}. \quad (3)$$

In this representation, the upper left 3×3 substructure of blocks corresponds to A_ℓ and is related to the divergence of the strain tensor in (1). The submatrix D_ℓ , resulting from the discretization of the divergence operator in the continuity equation, has a 1×3 block-structure, while G_ℓ , coming from the pressure gradient in (1), has a 3×1 block-structure and our discretization yields $D_\ell = G_\ell^\top$. The stabilization C_ℓ term acts only on the pressure and, therefore, gives a 1×1 block. It can be viewed as a discrete Laplacian operator acting on the pressure with Neumann boundary condition. Note that, while it is obvious that A_ℓ depends on the viscosity ν , it is also necessary to include ν^{-1} in the stabilization C_ℓ .

The mesh hierarchy \mathcal{T} allows to construct an efficient geometric all-at-once Uzawa multigrid method [12]. For solving the linear system (2), we apply multigrid V-cycles with three pre- and post-smoothing steps on level L and on each

coarser level two extra smoothing steps are added. Using a Uzawa type smoother then guarantees mesh-independent convergence, and we denote this type of multigrid as $V_{\text{var}}(3, 3)$. As the multigrid method acts both on velocity and pressure, the problem that needs to be solved on the bottom of the V-cycle is also of the form (2). For this, we employ the preconditioned minimal residual method (PMINRES). Our preconditioner has a block structure, where we apply a Jacobi preconditioned conjugate gradient method to the velocity part and perform a lumped mass matrix scaling on the pressure.

The HHG framework is a carefully designed and implemented high performance finite element multigrid software package [3, 12] which has already demonstrated its usability for geodynamical simulations [1, 22]. Conceptually, refinement of the input mesh \mathcal{T}_0 , which we call *macro mesh*, generates new nodes on edges, faces and within the volume of the tetrahedra of the input mesh. In HHG, these nodal values are organized by their geometric classification into a system of container data-structures called *primitives*. The nodal values in the interior of each macro tetrahedron are stored in a volume primitive, and similarly the values on macro edges, faces and vertices in their respective primitives. In this way, each nodal value is uniquely assigned to one primitive. Note that, only starting with refinement level two, we get nodes to store in the volume primitives. We use \mathcal{T}_2 as coarsest level in our multigrid solver. HHG's approach of splitting nodes between primitives of different geometric dimensionality naturally integrates with distributed-memory parallelism. Primitives are enriched by the nodal values of neighboring primitives in the form of ghost layer data-structures and kept up-to-date by MPI-communication in case of off-process dependencies, [3, 4].

The structured refinement of the input mesh, employed in HHG, results in the same types of tetrahedra being adjacent to each node within a certain primitive type and, thus, identical coupling patterns for these nodes. For constant ν on each macro tetrahedron, the discretization results also in the weights of these coupling being constant when proceeding from one node of a primitive to the next. This allows to use a constant stencil for all nodes in each volume primitive in a matrix-free approach, resulting in a significantly improved performance of computationally-intensive matrix-vector multiplications. In view of the system matrix in (3), we can identify the non-zero entries of each row of each block by a stencil and denote it by

$$s_{ij}^{A;m,n} = (A_\ell^{mn})_{ij}, \quad s_{ij}^{D;m} = (D_\ell^m)_{ij}, \quad s_{ij}^{G;m} = (G_\ell^m)_{ij}, \quad s_{ij}^C = (C_\ell)_{ij},$$

for row index i and column index j and $m, n \in \{1, 2, 3\}$. Within each volume primitive each stencil reduces to 15 non-zero entries. In the following, we will denote a stencil weight by s_{ij} , if there is no ambiguity. The full 15pt stencil at node i will be written as $s_{i,:}$.

3 Efficient On-the-Fly Stencil Assembly

While the hybrid approach of HHG exhibits superior performance, its geometry approximation on curved domains such as the spherical shell, is limited in

the sense that no refined nodes reside on the actual boundary. To account for this, in our implementation the fine grid nodes can be projected outwards onto the spherical surface. Also all interior nodes are projected to form concentric spherical layers. In a matrix-free framework, this comes at the cost that the FE stencils have to be repeatedly re-assembled on-the-fly.

We briefly describe the assembly procedure. For brevity, we show this only for A_ℓ^{11} from (3); the other entries are computed analogously. For linear FE the stencil weight s_{ij} can be computed by

$$s_{ij} = \sum_{t \in \mathcal{N}(i,j)} J_t^{-\top} \nabla \hat{\phi}_{i_{\text{loc}}} \cdot J_t^{-\top} \nabla \hat{\phi}_{j_{\text{loc}}} |\det(J_t)| \int_t \nu d\mathbf{x} = \sum_{t \in \mathcal{N}(i,j)} E_{i_{\text{loc}}, j_{\text{loc}}}^t \bar{\nu}_t \quad (4)$$

where J_t is the Jacobian of the mapping from the reference element \hat{t} , $\mathcal{N}(i, j)$ the set of elements with common nodes i and j , $E^t \in \mathbb{R}^{4 \times 4}$ the local element matrix on t , i_{loc} the element local index of the global node i , and $\hat{\phi}_{i_{\text{loc}}}$ the associated shape function. We can use a vertex based quadrature rule for the integral over ν by summing over the four vertices of t with weights $1/4$. This fits naturally to the HHG memory layout where the coefficients ν_i are stored point-wise. Also techniques for elimination of common sub-expressions can be employed, see [14].

A traditional matrix-free implementation requires to repeatedly evaluate (4) on-the-fly. For the full 15pt stencil $s_{i,:}$, this involves the computation of E^t on each of the 24 elements adjacent to node i . Even though we use optimized code generated by the FEniCS Form Compiler [18] for this task, it constitutes the most expensive part in the stencil assembly procedure and severely reduces overall performance. We term this approach IFEM and it will serve as our baseline for comparison. We remark that our implementation is node- and not element-centric. A benefit of this is, e.g., that the central stencil weight, essential for point-smoothers, is directly available. A disadvantage is that it performs redundant operations as it does not take into account the fact that each element matrix is shared by four nodes. We could slightly reduce the operation count by computing only the i -th row of the matrix when dealing with node i . However, this still involves the Jacobian of the reference mapping which gives the largest contribution to the number of operations.

In order to recover the performance of the original HHG implementation also on curved domains we recently proposed an alternative approach in [2] for block-wise constant ν . It replaces the expensive evaluation of (4) with approximating the values of s_{ij} by a low-order polynomial. The polynomial coefficients are computed via a least-squares fit in a setup phase and stored. Hence we denote the technique as LSQP. Later, whenever the stencil $s_{i,:}$ is needed, one has to evaluate 15 polynomials at node i , one for each stencil weight. In [2] quadratic polynomials gave the best compromise between accuracy and runtime performance provided that the coarse scale mesh was fine enough. Furthermore, we showed that this approximation does not violate the optimal approximation order of the L^2 -discretization error for linear finite elements, provided that the pairing of refinement depth L and macro mesh size H is selected carefully. Results for the Laplace operator [2, Table 4.1] indicated that for eight levels of refinement

the converted macro resolution of the spherical shell should be at least around 800 km. For the experiments carried out in Sect. 5, this is satisfied except for the smallest run, though even there we find good results, see Table 2.

For our PDE problem (2), we have to deal with two additional challenges. Firstly, instead of a scalar PDE operator as used in [2] we have a system of PDEs. Secondly, we have to incorporate the non-constant viscosity in the elliptic operators A_ℓ and C_ℓ . Conceptually, our discrete PDE system (3) consists of 4×4 operator blocks coupling the three velocity components and the pressure. Our implementation allows to individually replace any of 16 suboperators by a LSQP approximation. Here, we only report on the most compute time saving approach, which is to replace all of the suboperators by the surrogates. We do this on all levels \mathcal{T}_ℓ , apart from the coarsest one $\ell = 2$. We remark that the polynomials are evaluated at the nodal centers which leads to a small asymmetry in the operators. In [2] we found this relative asymmetry to be in $\mathcal{O}(h)$. This does not impact the algebraic convergence of the multigrid solver. However, it leads to a small issue on the coarsest level. There LSQP uses the same matrix \mathcal{L}_2 as IFEM. That matrix is symmetric positive semi-definite with a trivial kernel. Due to the asymmetry in our LSQP approach the restricted residual can include contributions from that kernel, which we fix by a simple projection of the right-hand side onto $\text{Im}(\mathcal{L}_2)$ to avoid problems with our PMINRES solver.

How to accommodate variable viscosity is a more intricate problem. In addition to the geometry variation, which can be approximated by quadratic polynomials as shown in [2], we also get variations due to the non-constant viscosity. If these are smooth enough, LSQP still yields good results. For more complex viscosity models, like in Sect. 5, with strong lateral variations a low order polynomial approximation may lead to poor results. Also in time-dependent and/or non-linear simulations where viscosity changes together with temperature and/or velocity, we would need to regularly recompute the polynomial coefficients. We, therefore, choose another approach. Recall that the most expensive part in (4) is the computation of the 24 element matrices. Instead of directly approximating s_{ij} , one can also approximate the contributions of E^t by quadratic polynomials. That is we substitute the expensive $E_{i_{\text{loc}}, j_{\text{loc}}}^t$ by an inexpensive polynomial approximation $\tilde{E}_{i_{\text{loc}}, j_{\text{loc}}}^t$ in (4). The polynomial approximation then solely depends on the geometry and is independent of the coefficients. Thus, it works for all kinds of coefficients. To distinguish between the two variants, we denote the original one as LSQP_S and the new modified one as LSQP_E. Note that due to the linearity of the least-squares fit w.r.t. the input data, LSQP_E yields the same stencil weights as LSQP_S in case of blockwise constant coefficients.

Each element matrix E^t contributes four values to one stencil $s_{i,:}$. Thus, in total the LSQP_E version requires to define $4 \cdot 24$ quadratic polynomials per macro element. For the full system (2) with general ν , we approximate the stencils of A_ℓ and C_ℓ via LSQP_E, while for G_ℓ and G_ℓ^\top the faster LSQP_S version is used.

4 Towards a Rigorous Performance Analysis

The LSQP_S approach was shown in [2] to be significantly faster than the traditional IFEM implementation. A more fundamental performance study must employ an absolute metric that does not rely on just quantifying the speed-up with respect to an arbitrary baseline implementation. To account for the real algorithmic efficiency and scalability of the implementation in relation to the relevant hardware limitations, we follow [14] where the notion of *textbook multigrid efficiency* [5] was extended to analyze massively parallel implementations. This metric is known as *parallel textbook multigrid efficiency* (parTME) and relies on detailed hardware performance models. While this goes beyond the scope of our current contribution, this section will provide first results and lay the foundation for further investigations.

The parTME metric is based on an architecture-aware characterization of a work unit (WU), where one WU is defined as one operator application of the full system. Here, we restrict ourselves to one scalar suboperator of (3). Conceptually, the extension to the full system is straightforward. The operator application can be expressed in terms of stencil based nodal updates $u_i \leftarrow \sum_{j=1}^{15} s_{ij}u_j$. The number of such updates performed per unit time is measured as *lattice updates per second* (Lup/s). This quantifies the primary performance capability of a given computer system with respect to a discretized system. A careful quantification of the Lup/s with an analytic white box performance model will often exhibit significant code optimization potential, as shown in [14]. Equally important, it provides absolute numbers of what performance can be expected from given hardware. This is crucial for a systematic performance engineering methodology. Our target micro-architecture is the eight-core Intel Sandy Bridge (SNB) Xeon E5-2680 processor with clock frequency 2.7 GHz as used in SuperMUC Phase 1. This processor delivers a peak performance of 21.6 double precision GFlops per core, and 172.8 GFlops per chip. However, this is under the assumptions that the code vectorizes perfectly for the Sandy Bridge AVX architecture, that the multiply-add instructions can be exploited optimally, and that no delays occur due to slow access to data in the different layers of the memory hierarchy.

We start with a classic cost count per update to derive an upper bound for the maximal achievable Lup/s. Here, we will compare the versions IFEM, LSQP_S and LSQP_E that are extensions of (CC) and (VC) for domains with curved boundaries.

First, we briefly recapitulate the cost for (CC) and (VC) and refer to [14] for details. On a blockwise regular mesh with constant coefficients, also the stencils are blockwise constant. Thus, for (CC) only one single 15pt stencil is required per block. This can be easily stored and loaded without overhead. Therefore, the cost for one stencil based update is 14 add/15 mult. For variable coefficients, the stencils have to be assembled on-the-fly. This requires the additional evaluation of (4). In the (VC) implementation, one can exploit the fact that on a polyhedral domain there exist only six different congruency classes of local elements. Thus, again per block its contributions to (4) can be pre-computed.

Table 1. Maximal and measured performance on one Intel SNB core

Kernel	Domain	Coefficients	Add/Mult	$P_{\text{core}}^{\text{max}}$	Measured
CC	Polyhedral	Blockwise constant	14/15	720 MLup/s	176 MLup/s
VC	Polyhedral	Variable	136/111	79.4 MLup/s	39.5 MLup/s
IFEM	Curved	Variable	1480/1911	5.7 MLup/s	0.7 MLup/s
LSQP _S	Curved	Moderately variable	44/45	245 MLup/s	71.7 MLup/s
LSQP _E	Curved	Variable	328/303	33.0 MLup/s	11.3 MLup/s

Now, we turn to curved domains. The LSQP_S approach is the extension of (CC) with the additional cost of 15 evaluations of a quadratic polynomial, one for each stencil component. For the evaluation, we use the scheme described in [2] that allows to evaluate a quadratic polynomial with 2 multiply-add operations. We note that LSQP_S can also be seen as an extension of (VC) for moderately variable coefficients. For problems with strongly variable coefficients, we propose either to use IFEM or the LSQP_E approach. Different from (VC), the contributions of the 24 neighboring element matrices must be re-computed on-the-fly. For IFEM, we count 56 additions and 75 multiplications per element matrix. The advantage of LSQP_E is obvious, since only 4 polynomial evaluations, one for each of the four contributions are required per element matrix. Again, this can be achieved with 8 multiply-add operations. In Table 1, we report the total number of operations for the different algorithms. Based on the operation count, the processor peak performance provides an upper limit on the achievable performance. In Table 1 we show these upper bounds as well as the measured values. For (CC) and (VC) the values are taken from [14]. For the measurements, we employed the Intel C/C++ Compiler 17.0 with flags `-O3 -march = native -xHost`.

Table 1 clearly shows that the peak rates are far from being obtained. For the simpler kernels (CC) and (VC), we carefully analyzed the performance discrepancy using the roofline and Execution-Cache-Memory models, see [14] and the references therein. Reasons why the peak rates are not achieved, are the limitations in bandwidth, but also bottlenecks that occur in the instruction stream and CPU-internal memory transfers between the cache layers. A full analysis for the advanced kernels is outside the scope of this contribution, but will be essential in the future to exhibit the possible optimization potential. But even the simple Flop count and the measured throughput values indicate the success of LSQP_S and LSQP_E in terms of reducing operation count as compared to a conventional implementation, such as IFEM. Similarly, the MLup/s show a substantial improvement. Both together, and the comparison with (CC) and (VC) indicate that there may be further room for improvement.

5 Accuracy and Weak Scaling Results

In this section, we analyze the accuracy and scaling behavior of our implementation for a geophysical application. Our largest simulation run will be with a global resolution of the Earth’s mantle of ~ 1.7 km.

System: We run our simulations on SuperMUC Phase1, a TOP500 machine at the LRZ, Garching, Germany. It is an IBM iDataPlex DX360M4 system equipped with eight-core SNB processors, cf. Sect. 4. Per core around 1.5 GB of memory are available to applications. Two sockets or 16 cores form one compute node, and 512 nodes are grouped into one island. The nodes are connected via an Infiniband FDR10 network. In total, there are 147 456 cores distributed on 18 islands with a total peak performance of 3.2 PFlop/s. We used the Intel compiler with options as in Sect. 4 and the Intel 2017.0 MPI library.

Setup: The icosahedral meshing approach for the spherical shell does not allow for an arbitrary number of macro elements in the initial mesh and the smallest feasible number of macros would be 60 already. Also we are interested in the scaling behavior from typical to large scale scenarios. Thus, we perform experiments starting on one island and scaling up to eight islands. We try to get as close as possible to using the full number of nodes on each island, while keeping the tangential to radial aspect ratio of the macro elements close to 1:1.

Inside a node, we assign two macro elements to each MPI process running on a single core. As the memory consumption of our application is on average about 1.7 GB per core, we utilize only 12 of the 16 available cores per node. These 12 cores are equally distributed on the two sockets by setting $L_MPI_PIN_PROCESSOR_LIST = 0-5,8-13$. A deep hierarchy with 8 levels of refinement is used. This yields problem sizes with $1.3 \cdot 10^{11}$ DoFs on 5 580 cores (one island), $2.7 \cdot 10^{11}$ DoFs on 12 000 cores (two islands), $4.8 \cdot 10^{11}$ DoFs on 21 600 cores (four islands) and $1.1 \cdot 10^{12}$ DoFs on 47 250 cores (eight islands).

Geophysical Model: In order to have a realistic Stokes-type problem (1) as it appears in applications, we consider the following model. On the top of the mantle we prescribe non-homogeneous Dirichlet boundary conditions, composed of a no-outflow component and tangential components given by present day plate velocity data from [20]. On the core-mantle boundary vanishing tangential shear stress resulting in a free-slip condition is enforced.

In terms of viscosity, we employ a similar model as used in [9]. The viscosity is the product of a smooth function depending on the temperature and the radial position and a discontinuous function reflecting a viscosity jump in radial direction due to an asthenospheric layer, a mechanically weak zone where the viscosity is several orders of magnitude smaller than in the lower mantle. The concrete thickness of the asthenosphere is unknown and subject to active research, see e.g. [22]. Here, we choose the model from [22] with a thickness of 660 km as this depth is one of two transition zones of seismic wave velocities. The viscosity model in non-dimensional form is given by

$$\nu(\mathbf{x}, T) = \exp\left(2.99 \frac{1 - \|\mathbf{x}\|_2}{1 - r_{\text{cmb}}} - 4.61T\right) \begin{cases} 1/10 \cdot 6.371^3 d_a^3 & \text{for } \|\mathbf{x}\|_2 > 1 - d_a, \\ 1 & \text{else.} \end{cases}$$

where $d_a = 660/R$ with the Earth radius $R = 6371$ (km). Finally, we used present day temperature and density fields to compute the buoyancy term \mathbf{f} and the viscosity, see [7].

Table 2. Results for one island scenario with $1.3 \cdot 10^{11}$ degrees of freedom: differences in the velocities inside the mantle obtained with IFEM and LSQP for different refinement levels (left); characteristic velocities in cm/a for level 8 (right).

level	discr. L_2	max-norm	charac. velocities	IFEM	LSQP	difference
4	$2.81 \cdot 10^{-4}$	$2.58 \cdot 10^{-2}$	avg. (whole mantle)	5.92	5.92	$5.60 \cdot 10^{-5}$
5	$4.05 \cdot 10^{-4}$	$4.84 \cdot 10^{-2}$	avg. (asthenosphere)	10.23	10.23	$1.10 \cdot 10^{-4}$
6	$5.19 \cdot 10^{-4}$	$6.70 \cdot 10^{-2}$	avg. (lower mantle)	4.48	4.48	$1.12 \cdot 10^{-4}$
7	$5.75 \cdot 10^{-4}$	$7.89 \cdot 10^{-2}$	max. (asthenosphere)	55.49	55.49	$2.61 \cdot 10^{-4}$
8	$6.83 \cdot 10^{-4}$	$8.58 \cdot 10^{-2}$	max. (lower mantle)	27.46	27.46	$6.33 \cdot 10^{-4}$

Accuracy: Before considering the run-time and scaling behavior of our new LSQP approach, we demonstrate its applicability by providing in Table 2 a comparison to results obtained with IFEM. We observe that the differences are sufficiently small in relation to typical mantle velocities and the uncertainties in the parameters that enter the model. The fact that the differences slightly grow with level reflects the two-scale nature of LSQP, as the finite element error decreases with mesh size h of the finest level, while the matrix approximation error is fixed by the mesh size H of the coarsest level, see also [2].

Memory Consumption: One important aspect in large scale simulations is memory consumption. Ideally, it should stay constant in weak scaling runs, as the number of DoFs per process remains the same. However, this is not always the case, especially in large scale simulations, due to buffer sizes that scale with the number of MPI ranks, see [10] for some examples.

To determine how strongly this affects our application, we measure the memory consumption per MPI process using the Intel MPI Performance Snapshot (mps) tool [16]. In Fig. 1 (left), we report the mean and maximum memory usage over all MPI processes. For each process, we assigned two volume primitives. The difference between the mean and maximum value comes from the different numbers of lower dimensional primitives attached to one process.

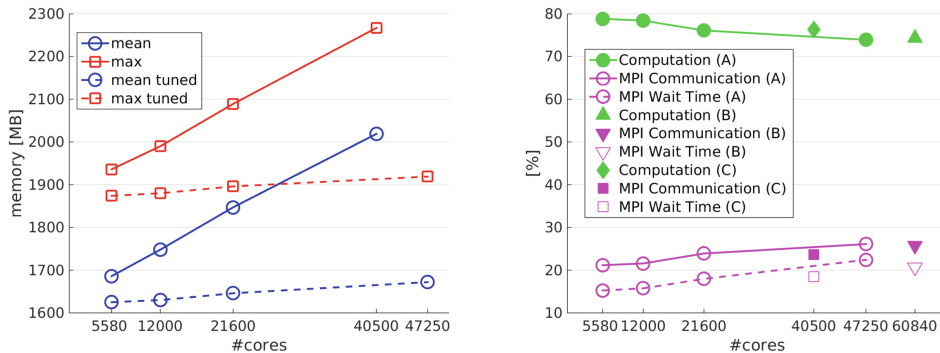


Fig. 1. Left: mean and max memory usage over all MPI processes. Right: percentage of computation versus communication (non-overlapping).

Table 3. Default and tuned Intel MPI DAPL settings (p = total no. of MPI processes.)

Environment variable	Default	Tuned
I_MPI_DAPL_UD_SEND_BUFFER_NUM	$16 + 4p$	8208
I_MPI_DAPL_UD_RECV_BUFFER_NUM	$16 + 4p$	8208
I_MPI_DAPL_UD_ACK_SEND_POOL_SIZE	256	8704
I_MPI_DAPL_UD_ACK_RECV_POOL_SIZE	$512 + 4p$	8704
I_MPI_DAPL_UD_RNDV_EP_NUM	4	2

For the default MPI buffer settings, we observe a significant linear increase in the memory usage caused by MPI. As a result the eight islands case runs out of memory. We therefore reduced the number of cores per node for this run to 10 resulting in configuration (B) (Table 4). Alternatively, one could decrease the number of MPI ranks for the same problem size and core count by using hybrid MPI/OpenMPI parallelism as done in [11]. This does, however, also not attack the root of the problem. For this, we need to deal with the MPI library instead.

On an Infiniband cluster the Intel MPI library uses the Shared Memory (SHM) transport mechanism for intra-node communication, while for inter-node communication it uses the Direct Access Programming Library (DAPL). While the UD (User Datagram) version of DAPL is already much more memory conservative than the RC (Reliable Connection) version, the default buffer pool sizes still scale with the number of MPI processes, [10]. This can be seen from the default configuration values in Table 3. As suggested in [10], we set the internal DAPL UD buffer sizes to the fixed values given in Table 3, leading to a significant decrease of the memory consumption. The latter, now, shows almost perfect weak scalability and allows to go to extreme scales. Compared to the all-to-all communication scenarios shown in [10], we even see a much better scaling behavior up to 47 250 MPI ranks. We also do not notice any performance loss.

Computation vs. Communication: Current supercomputers provide tremendous computing capacities. This makes computations relatively cheap compared to communication that gets more expensive, the more processes are used. So, often communication is the bottleneck in high-performance codes.

To investigate the ratio of both, we again employ the Intel mps tool to measure the time for computation, i.e., mean time per process spent in the application code versus time for MPI communication. The latter is the time spent

Table 4. Configurations used in our experiments; default is to use configuration (A).

Configuration	Macro elements per core	Cores per node	# Cores (8 islands)	# DoFs (8 islands)
A	2	12	47 250	$1.1 \cdot 10^{12}$
B	2	10	40 500	$9.1 \cdot 10^{11}$
C	1	16	60 840	$6.8 \cdot 10^{11}$

inside the MPI library. This tool also reports the MPI imbalance, i.e., the mean unproductive wait time per process spent in the MPI library calls, when a process is waiting for data. This time is part of the reported MPI communication time. Here, a high percentage of computation is favorable, while the MPI imbalance should be small. Note that we do not overlap computation and communication. Using overlapping communication does not improve the performance significantly [13].

Besides our default configuration (A) and configuration (B), we consider a third case (C) for the eight islands run. Here, we increase the number of cores per node to the maximum of 16. This increases the total number of MPI processes to 60 840. To make this feasible, we assign one single macro element per rank. This can be seen as the most critical run in terms of communication as it involves the largest number of MPI processes.

The results are shown in Fig. 1 (right), where all initialization times are excluded. We find only a slight increase of communication during weak scaling. And even for the extreme cases the amount of communication is only about 25%. However, we also observe a relatively high MPI imbalance of around 20%. This is partly due to the imbalance of lower dimensional primitives and could be improved by a load balancing scheme that takes the cost of face primitives into account. Changing the number of macro elements per MPI process (C), or varying the number of cores per node (A, B) does hardly affect the results.

Parallel Efficiency: Finally, we report in Table 5 the time-to-solution. For these runs, we switch off any profiling. The iteration is stopped when the residual is reduced by 10^5 starting with a zero initial guess. For our geophysical application such a stopping criterion is more than sufficient. The high viscosity jump in our application makes the problem particularly difficult for the coarse grid (c.g.) solver. Choosing the right stopping criterion is essential for the Uzawa multigrid (UMG) convergence rate, while tuning it becomes quite tricky. It turned out that a criterion based on a maximal iteration count is favorable compared to a tolerance based criterion. In Table 5, we also report the best values we came up with. We remark that for the two islands case we could not find an acceptable number of c.g. iterations that reduced the UMG V-cycles below 10. For this run,

Table 5. Weak scaling results for geophysical application: Runtime w/ and w/o coarse grid solver (c.g.) and no. of UMG iterations. Values in brackets show no. of c.g. iterations (preconditioner/Minres). Parallel efficiency is shown for timings w/ and w/o c.g. *Timings and parallel efficiency are scaled to 7 UMG iterations.

Islands	Cores	DoFs	Global resolution	UMG V-cycles	Time-to-solution	Time-to-sol. w/o c.g.	Parallel efficiency
1	5 580	$1.3 \cdot 10^{11}$	3.4 km	7 (50/150)	1347 s	1151 s	1.00/1.00
2	12 000	$2.7 \cdot 10^{11}$	2.8 km	10* (100/150)	1493 s	1183 s	0.90/0.97
4	21 600	$4.8 \cdot 10^{11}$	2.3 km	7 (50/250)	1468 s	1201 s	0.92/0.96
8	47 250	$1.1 \cdot 10^{12}$	1.7 km	8* (50/350)	1609 s	1209 s	0.83/0.95

the element aspect ratio deviates most from 1:1. For all other simulations, the UMG iterations are stable around 7. Note that for the largest simulation the residual reduction was $9.9 \cdot 10^4$ after 7 iterations, so the stopping criterion was only slightly missed. For a fair comparison of runtimes, we scaled all timings to 7 iterations. On up to eight islands, we find a parallel efficiency of 83%. Taking into account that it includes the c.g. solver with its non-optimal complexity, this is an excellent value. Examining the time-to-solution with the c.g. solver excluded, we find an almost perfect parallel efficiency on up to 47 250 cores of 95%. Compared to the IFEM reference implementation, we observe for the smallest run a speed-up of a factor larger than 20. In order to save core-h, and thus energy, we did not perform such a comparison for the larger scenarios.

6 Outlook

We extended our LSQP approach to systems of PDEs with variable coefficients and demonstrated that it is suitable for large scale geophysical applications. A systematic performance analysis demonstrates the new matrix-free techniques lead to substantial improvements compared to conventional implementations and they indicate that there is potential for further improvement. In future work, we will expand our study by detailed performance models for a rigorous performance classification and optimization.

Acknowledgments. This work was partly supported by the German Research Foundation through the Priority Programme 1648 “Software for Exascale Computing” (SPPEXA) and WO671/11-1. The authors gratefully acknowledge the Gauss Centre for Supercomputing (GCS) for providing computing time on the supercomputer SuperMUC at LRZ. Special thanks go to the members of LRZ for the organization and their assistance at the “LRZ scaling workshop: Emergent applications”. Most scaling results were obtained during this workshop.

References

1. Bauer, S., et al.: Hybrid parallel multigrid methods for geodynamical simulations. In: Bungartz, H.-J., Neumann, P., Nagel, W.E. (eds.) Software for Exascale Computing - SPPEXA 2013–2015. LNCSE, vol. 113, pp. 211–235. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40528-5_10
2. Bauer, S., Mohr, M., Rde, U., Weismller, J., Wittmann, M., Wohlmuth, B.: A two-scale approach for efficient on-the-fly operator assembly in massively parallel high performance multigrid codes. *Appl. Numer. Math.* **122**, 14–38 (2017)
3. Bergen, B., Gradl, T., Rde, U., Hlsemann, F.: A massively parallel multigrid method for finite elements. *Comput. Sci. Eng.* **8**(6), 56–62 (2006)
4. Bergen, B., Hlsemann, F.: Hierarchical hybrid grids: data structures and core algorithms for multigrid. *Numer. Linear Algebra Appl.* **11**, 279–291 (2004)
5. Brandt, A.: Barriers to achieving textbook multigrid efficiency (TME) in CFD. Institute for Computer Applications in Science and Engineering, NASA Langley Research Center (1998)

30 S. Bauer et al.

6. Brezzi, F., Douglas, J.: Stabilized mixed methods for the Stokes problem. *Numer. Math.* **53**(1), 225–235 (1988)
7. Colli, L., Ghelichkhan, S., Bunge, H.P., Oeser, J.: Retrodictions of Mid Paleogene mantle flow and dynamic topography in the Atlantic region from compressible high resolution adjoint mantle convection models: sensitivity to deep mantle viscosity and tomographic input model. *Gondwana Res.* **53**, 252–272 (2018)
8. Davies, D.R., Davies, J.H., Bollada, P.C., Hassan, O., Morgan, K., Nithiarasu, P.: A hierarchical mesh refinement technique for global 3-D spherical mantle convection modelling. *Geosci. Model Dev.* **6**(4), 1095–1107 (2013)
9. Davies, D.R., Goes, S., Davies, J., Schuberth, B., Bunge, H.P., Ritsema, J.: Reconciling dynamic and seismic models of earth’s lower mantle: the dominant role of thermal heterogeneity. *Earth Planet. Sci. Lett.* **353–354**, 253–269 (2012)
10. Durnov, D., Steyer, M.: Intel MPI Memory Consumption. *The Parallel Universe* 21 (2015)
11. Gmeiner, B., Rüde, U., Stengel, H., Waluga, C., Wohlmuth, B.: Performance and scalability of hierarchical hybrid multigrid solvers for Stokes systems. *SIAM J. Sci. Comput.* **37**(2), C143–C168 (2015)
12. Gmeiner, B., Huber, M., John, L., Rüde, U., Wohlmuth, B.: A quantitative performance study for Stokes solvers at the extreme scale. *J. Comput. Sci.* **17**(Part 3), 509–521 (2016)
13. Gmeiner, B., Köstler, H., Stürmer, M., Rüde, U.: Parallel multigrid on hierarchical hybrid grids: a performance study on current high performance computing clusters. *Concurr. Comput.: Pract. Exp.* **26**(1), 217–240 (2014)
14. Gmeiner, B., Rüde, U., Stengel, H., Waluga, C., Wohlmuth, B.: Towards textbook efficiency for parallel multigrid. *Numer. Math. Theor. Meth. Appl.* **8**(01), 22–46 (2015)
15. Heister, T., Dannberg, J., Gassmöller, R., Bangerth, W.: High accuracy mantle convection simulation through modern numerical methods - II: realistic models and problems. *Geophys. J. Int.* **210**(2), 833–851 (2017)
16. Intel Corp.: MPI Performance Snapshot, version: 2017.0.4 (2017). <https://software.intel.com/en-us/node/701419>
17. Kronbichler, M., Kormann, K.: A generic interface for parallel cell-based finite element operator application. *Comput. Fluids* **63**, 135–147 (2012)
18. Logg, A., Ølgaard, K.B., Rognes, M.E., Wells, G.N.: FFC: the FEniCS form compiler. In: Logg, A., Mardal, K.A., Wells, G. (eds.) Automated solution of differential equations by the finite element method. LNCSE, vol. 84, pp. 227–238. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-23099-8_11
19. May, D.A., Brown, J., Pourhiet, L.L.: A scalable, matrix-free multigrid preconditioner for finite element discretizations of heterogeneous Stokes flow. *Comput. Methods Appl. Mech. Eng.* **290**, 496–523 (2015)
20. Müller, R.D., Sdrolias, M., Gaina, C., Roest, W.R.: Age, spreading rates, and spreading asymmetry of the world’s ocean crust. *Geochem. Geophys. Geosyst.* **9**(4), 1525–2027 (2008)
21. Rudi, J., Malossi, A.C.I., Isaac, T., Stadler, G., Gurnis, M., Staar, P.W.J., Ineichen, Y., Bekas, C., Curioni, A., Ghattas, O.: An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth’s mantle. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, pp. 5:1–5:12. ACM (2015)
22. Weismüller, J., Gmeiner, B., Ghelichkhan, S., Huber, M., John, L., Wohlmuth, B., Rüde, U., Bunge, H.P.: Fast asthenosphere motion in high-resolution global mantle flow models. *Geophys. Res. Lett.* **42**(18), 7429–7435 (2015). <https://doi.org/10.1002/2015GL063727>

Chapter 5

Dynamic topography

This chapter is part of a manuscript that was invited to the special issue of *Journal of Computational Science* in 2019 [1]

Large-scale Simulation of Mantle Convection Based on a New Matrix-Free Approach, S. Bauer, M. Huber, S. Ghelichkhan, M. Mohr, U. Rüde and B. Wohlmuth.

It is an extended version of the article from chapter 4. Here we present the novel parts of the manuscript.

Earth's topography is by no means constant. In fact, it is constantly changed by various processes, e.g., erosion and sedimentation, or global isostatic adjustment. The latter being the fact that on 'short' time-scales Earth behaves elastically when a load is added on or removed from its surface. The convection driven stresses from the mantle induce deflections on the surface and core-mantle boundary (CMB) too, referred to as *dynamic topography* (see [2] for a review).

Such surface deflections produce notable gravity anomalies [3]. Since early on, in lack of direct measurements for surface dynamic topography, Geodynamicists use the equipotential figure of the Earth to constrain rheological depth-dependence in the mantle [4, 5]. These models take advantage of the semi-analytical solutions with propagator matrices [6]. Despite the initial progress, the best fitting models are non-unique [7]. That is, a set of different rheological depth-dependencies can be found to fit the gravitational observations equally well.

Recently, there has been renewed efforts to quantify dynamic topography at present geological time [8], and its evolution as a proxy for mantle flow processes (see [9] and references therein). These efforts can potentially further constrain rheology in the Earth's mantle. However, the new results on dynamic topography have aroused a debate too. The focus of this debate is on the amplitudes and spectrum of surface dynamic topography. Therefore, it is important to explore different mechanisms that can potentially affect the spectrum of dynamic topography.

One of the often neglected parameters in global dynamic models is the lateral viscosity variations (LVVs) induced by temperature variations in the mantle. This has been mainly because of complexities arising from mode coupling in propagator matrix technique. Fortunately, our FE approach using LSQP_E allows including orders of magnitude viscosity variations in our models. In this chapter, we will use our approach to study the effects of LVVs in mantle convection, and in particular the dynamic topography.

The essential part of a model for dynamic topography is the normal component of the

stress tensor:

$$\sigma_{nn}^s = \mathbf{n}^T \left[-\nu(\nabla \mathbf{u} + (\nabla \mathbf{u})^\top) + p\mathbf{I} \right] \mathbf{n} \quad (5.1)$$

at the surface, denoted by s . In discrete form, this can be computed as

$$\sigma_{nn,\ell}^s = (M_\ell^s)^{-1} [(\mathbf{f}_\ell - A_\ell \mathbf{u}_\ell - B_\ell \mathbf{p}_\ell) \mathbf{n}_\ell]_s, \quad (5.2)$$

where M_ℓ^s is the lumped surface mass matrix. For the ease of notation, we will skip the index ℓ in the following. Let $\mathcal{V}_i = \cup_{t \in \mathcal{N}(i,i)} \bar{t}$, i.e., the closed union of all elements t sharing node i . Then, for each boundary node i , the entry of M_ℓ^s is given by

$$(M_\ell^s)_{ii} = \frac{1}{3} \int_{\partial\Omega \cap \mathcal{V}_i} 1 \, ds. \quad (5.3)$$

Similarly, we can also compute dynamic topography σ_{nn}^c at the CMB. For a more detailed derivation of (5.2), we refer to [10].

In the following section, we will first introduce a standard community benchmark for dynamic topography. This benchmark will then be used this to verify our implementation and demonstrate the LSQP approach yields trust-worthy results also in terms of normal stresses. Finally, in Sec. 5.4 we will compare and discuss the dynamic topography for a model with and without temperature dependent lateral viscosity variations at unprecedented global resolution.

5.1 Benchmark Setup

We repeat the benchmark described in [10, 11, 12] and add an additional variant for a layered viscosity profile with jump at 410 km depth.

In [11], results are shown with and without self-gravitation. As stated therein, self-gravitation can be added in a post-processing step. Thus, we only consider cases without self-gravitation here.

The buoyancy term, i.e., the right-hand side of the momentum part of (1.5), is given in spherical coordinates by

$$T(r, \phi, \theta) = \delta(r - r_0) Y_l^m(\phi, \theta).$$

Here, Y_l^m denotes a spherical harmonic function of degree l and order m given by

$$Y_l^m(\phi, \theta) = \cos(m\phi) p_{lm}(\theta),$$

where

$$p_{lm}(\theta) = \sqrt{\frac{(2l+1)(l-m)!}{2\pi(1+\delta_{m0})(l+m)!}} P_{lm}(\theta)$$

are the normalized associated Legendre functions. For the benchmark, free-slip boundary conditions are prescribed on the surface and the CMB. If a viscosity model is employed that only shows radial variations, then this setup allows to compute a semi-analytical solution based on a propagator matrix technique, see e.g. [13]. This can then be compared to the solutions of the numerical model.

The main output of this benchmark are so-called kernels for topography and characteristic velocity. These are generated the following way: for a fixed r_0 in (5.5) the buoyancy is

computed. That is, it is non-zero in exactly one radial layer. Then, for this buoyancy the normal stress on the surface and CMB is computed. This can be represented by a spherical harmonic expansion. Next, the spectral coefficients of this expansion for the chosen m, l are computed. This gives one value of the kernel at depth r_0 .

Additionally, also the characteristic horizontal velocity U at the surface and CMB is computed via

$$U^2 = \frac{1}{r^2} \frac{1}{l(l+1)} \int_A \|\mathbf{u}\|^2 \quad (5.4)$$

where the two dimensional domain A is either the surface, or the CMB area. Again, the discrete form can be computed via the lumped mass matrix M_ℓ^s . Note, (5.4) defines U modulo its sign. Thus, we follow the convention from [11] and set U to be negative on the surface, and positive on the CMB.

Finally, this procedure is successively repeated for a different r_0 , sampling the whole mantle.

5.2 Benchmark Implementation and Results

We are going to perform benchmark tests setting the order m to be equal to 0 and degrees to be $l \in \{2, 5, 8, 15\}$. For the setup of the right-hand side, we need to preserve the integral property of the δ -function, i.e., $\int_\Omega \delta dx = 1$. We set the nodal values of each layer r_k as

$$\delta(r_k - r_0) \approx \begin{cases} 1 & \text{for } r_k = r_0, \\ 0 & \text{otherwise,} \end{cases} \quad (5.5)$$

and integrate over the sphere with radius r_0 . More precisely, we apply to the values of layer r_0 the two dimensional lumped mass matrix $M_\ell^{r_0}$. For the implementation of $M_\ell^{r_0}$, we proceed on layer r_0 as on the surface in (5.3). We remark that our approach is different to the implementation of the δ -function of [10, 11], since there hexahedral meshes which are uniform in radial direction are used. Then a simple scaling by the meshes is sufficient such that the three dimensional mass matrix can be used for the setup.

For all test runs, the default configuration is to use 128 layers, which is twice as much as used in [11]. Furthermore, if not explicitly mentioned otherwise, we always use the LSQP approach.

In a first test, we set $\nu_1 \equiv 1$ and refer to this as Case 1. For the second test Case 2, we consider a layered viscosity profile with a jump at 410 km depth, which is the location of another seismic discontinuity, as opposed to the one at 660 km depth considered before

$$\nu_2(\mathbf{x}) = \begin{cases} 1/10 \cdot 6.371^3 d_b^3 & \text{for } \|\mathbf{x}\|_2 > 1 - d_b, \\ 1 & \text{else.} \end{cases}$$

Here, $d_b = 410/R$, which roughly yields a $150\times$ fold viscosity jump. As this is an even higher viscosity jump than for the example from Sec. 4.5 it becomes even more challenging for our solver.

Note, that for both cases we have block-wise constant viscosity values. Thus, LSQP_S and LSQP_E are equivalent and we choose the computationally more efficient LSQP_S variant here.

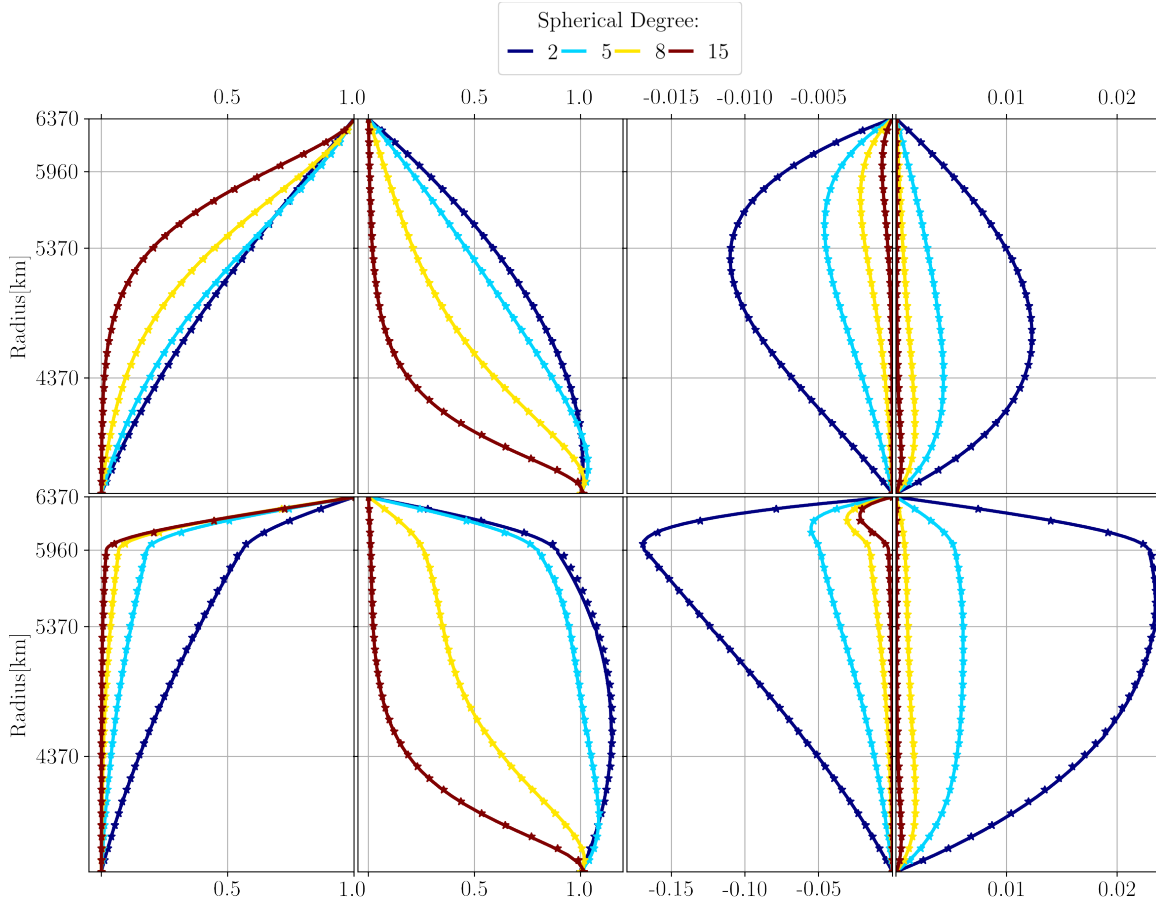


Figure 5.1: Benchmark results for Case 1 (top panel) and Case 2 (bottom panel): Spectral coefficients of surface and CMB topography, and characteristic horizontal velocities at surface and CMB (from left to right). Dotted values show semi-analytical solution and solid lines the solution obtained with HHG.

The resulting kernels are shown in Fig. 5.1 together with the semi-analytical solutions from the propagator matrix technique. For both test cases, we find very good agreement between our solver and the semi-analytical solutions.

Furthermore, we perform convergence tests with increasing refinement depth against the semi-analytical solution for a fixed radius r_0 , see Fig. 5.2. Here, we choose r_0 to represent the central layer of the mantle, i.e. $r_0 = r_{\text{srf}} + r_{\text{cmb}}/2$. A similar analysis is presented in e.g. [10, 11], whereas we use even finer resolutions. We find a quadratic convergence in all cases which is in agreement with the literature. In some cases, we observe that higher spherical degrees perform even better than expected. This can be seen, e.g., in the characteristic horizontal surface velocities, where degree 8 is even better than degrees 2 and 5.

The results of this test are twofold. First, they verify the overall accuracy of our solver, and secondly, they demonstrate that our LSQP approximation does not deteriorate its accuracy.

5.3 Scalability

The benchmark simulations were mostly carried out on SuperMUC Phase 1, but partially also on the Hazel Hen system at HLRS, Stuttgart, Germany. The largest simulations in Sec. 5.4

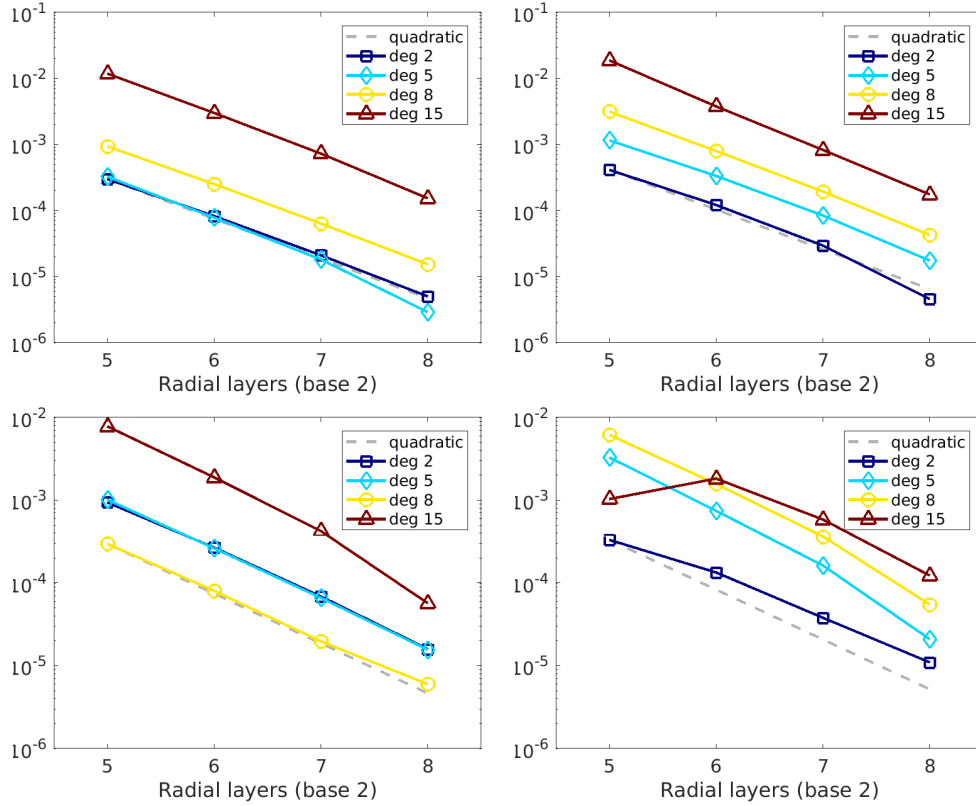


Figure 5.2: Relative errors with respect to semi-analytical solution at fixed depth $r_0 = \tau_{\text{srf}} + \tau_{\text{cmb}}/2$. (Top panel, left to right) Surface and CMB topography. (Bottom panel, left to right) Characteristic horizontal velocities at surface and CMB.

were performed on Hazel Hen on 75 810 compute cores. While the performance on SuperMUC was already extensively discussed, we will present in this section also scaling results for Hazel Hen together with a brief description of the hardware details.

Hazel Hen is a Cray XC40 system with Intel Xeon CPU E5-2680 v3 processors. The processor compute nodes consist of two sockets, each with 12 cores that are clocked at 2.5 GHz. Each node provides about 128 GB of memory and the nodes are connected via an Cray Aries network. Overall, the Hazel Hen system combines 7 712 nodes, or 185 088 compute cores respectively, within 41 cabinets and has a peak performance of 7.4 PFLOPs/s. Currently, it is ranked number 27 in the TOP500 list (June 2018).

For the scaling runs, we will use the benchmark setup from test Case 1 for a fixed r_0 . Furthermore, the refinement depth is set to $L = 8$ and one macro element is assigned per MPI rank. The UMG solver stops after a residual reduction of 10^6 . Even though the faster LSQP_S variant is applicable for this case, we use the LSQP_E approach for the scaling runs, as this variant will be used for the application scenarios in Sec. 5.4.

Results are shown in Tab. 5.1. The number of UMG cycles is asymptotically 7. For the smallest two runs, slightly more iterations are necessary. For a fair comparing of the time-to-solutions, all timings are rescaled, as before, to 7 iterations. We note that for this isoviscous scenario the coarse grid solver needs only a few iterations, and its influence on the overall time-to-solution is relatively small. Also on the Hazel Hen system the parallel efficiency of

Table 5.1: Weak scaling results on Hazel Hen. Timings and parallel efficiency are scaled to 7 UMG iterations.

cores	DoFs	UMG V-cycles	time-to- solution	parallel efficiency
480	$1.3 \cdot 10^9$	9	1 313 s	1.00
1 920	$5.4 \cdot 10^9$	8	1 343 s	0.98
3 840	$1.1 \cdot 10^{10}$	7	1 341 s	0.98
8 640	$2.4 \cdot 10^{10}$	7	1 329 s	0.99
15 360	$4.3 \cdot 10^{10}$	7	1 344 s	0.98

98% on up to 15 360 cores is excellent.

5.4 Dynamic topography with lateral viscosity variations

To compute the Earth’s surface dynamic topography, we use a-priori thermodynamic properties and viscosity structure for the Earth’s mantle as input to our numerical scheme. First, we convert a recent shear wave velocity tomography model [14], to temperature and density, using a pyrolitic mantle mineralogical model. That is, converting to temperature and density, we assume a purely thermal origin for seismic anomalies. As the isochemical assumption does not hold in continental regions, we remove the fast seismic anomalies in the uppermost 300 km. The mineralogical model is derived by combining the most recent elastic model of [15], with Q_5 , an adaptation of a mantle anelasticity model in [16]. To adapt the anelasticity model to lower-mantle conditions we update the solidus temperature in [16] to [17]. Further, we model three different viscosity scenarios for the Earth’s mantle:

Case A: This scenario has no lateral viscosity variations and serves as our reference case. The radial viscosity profile is composed of three layers, namely lithosphere, asthenosphere and the lower mantle. The lithosphere thickness is 100 km and that of the asthenosphere is set to 410 km. The viscosity jump to the lower mantle is set according to the Haskell constraint, $\mu_{\text{asth}} = 1/10 \cdot 6.371^3 d_b^3$, see e.g. [18]. This is the same viscosity jump as in the benchmark Case 2 from Sec. 5.2. For the ease of notation, we further introduce the scaled lithosphere radius $r_L := 1 - 100/R$ and asthenosphere radius $r_A := 1 - (100+410)/R$. Then, the reference viscosity profile reads as

$$\nu_A(\mathbf{x}) = \begin{cases} 1 & r_L \leq r, \\ \mu_{\text{asth}} & r_A \leq r < r_L, \\ 1 & \text{else,} \end{cases}$$

where $\|\mathbf{x}\|_2 = r$.

Case B: Next, we add lateral viscosity variations in the uppermost ~ 300 km. For this purpose, we employed the continental lithospheric thickness model, TC1 [19]. Oceanic lithosphere is then modeled by a half space cooling model. As demonstrated in Fig. 5.6, thickness of the lithosphere ranges from 20 km down to more than 350 km penetrating the asthenospheric layer. That is, while in Case A the lithosphere is a fixed channel, in Case B, its actual shape varies between oceans and continents. To account for this, we remove the lithosphere layer from our model in Case A, and prescribe its thickness via a 3D function, γ . That is, we

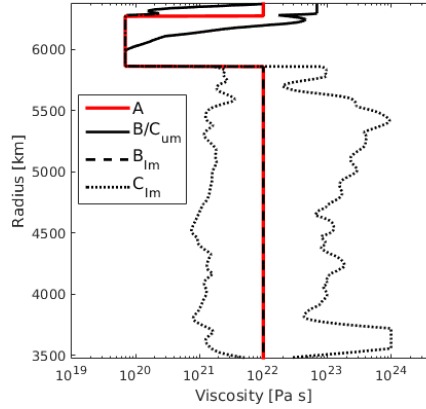


Figure 5.3: Radial viscosity profiles for Cases A (red), B (black-dashed), and C (black-dotted). The lateral variations in Cases B and C are visualized by the min/max values in each layer. In the upper mantle (um) the profiles of B and C are identical (solid lines) and in the lower mantle (lm) the dashed line shows the radial profile of Case B, and the dotted min/max curves the temperature dependent variations of Case C. Values are scaled with reference viscosity 10^{22} [Pa s].

define a scalar field γ ranging from 1 to 10^3 . We set $\gamma(\mathbf{x}) = 10^3$ for all points \mathbf{x} that are part of the lithosphere. Beneath the lithosphere the values drop from 10^3 to 1, smoothly as the result of a lateral spectral Gaussian filtering as shown in Fig. 5.6.

$$\nu_B(\mathbf{x}, T) = \gamma(\mathbf{x}) \begin{cases} \mu_{\text{asth}} & r_A \leq r, \\ 1 & \text{else.} \end{cases}$$

Case C: In our final test case, we also include lateral viscosity variations in the lower mantle. Here, in addition to the a-priori radial profile described in Case A, the lateral variations are inferred from an exponential temperature dependence, consistent with laboratory derived creep law [20]. In the upper mantle, the lateral variations are identical to case B. The viscosity model reads as:

$$\nu_C(\mathbf{x}, T) = \gamma(\mathbf{x}) \begin{cases} \mu_{\text{asth}} & r_A \leq r, \\ \exp\left(\Delta\mu_{\text{LM}}\left(\frac{1}{T} - \frac{1}{T_0}\right)\right) & \text{else,} \end{cases}$$

with radial reference temperature T_0 , and $\Delta\mu_{\text{LM}} = 5.75$. The value of $\Delta\mu_{\text{LM}}$ is chosen specifically to yield three orders of magnitude of variation in the lower mantle. Both, T_0 and T are scaled by the maximum value of T . The shape of the three viscosity models is illustrated in Fig. 5.3.

The simulations are carried out on Hazel Hen (Case A) and SuperMUC Phase 1 (B, C). For simulations on SuperMUC, the setup is as described in Sec. 4.5, namely $1.1 \cdot 10^{12}$ DoFs on 47 250 cores with a global resolution of ~ 1.7 km. On Hazel Hen, we had access to an even larger number of cores that allowed us to run simulations with an unprecedented global resolution of less than 1.5 km on 75 810 compute cores and $1.6 \cdot 10^{12}$ DoFs. For visualization and post-processing of the dynamic topographies, all results were interpolated to the same longitude-latitude grid. In all cases, free-slip boundary conditions are imposed on surface and CMB.

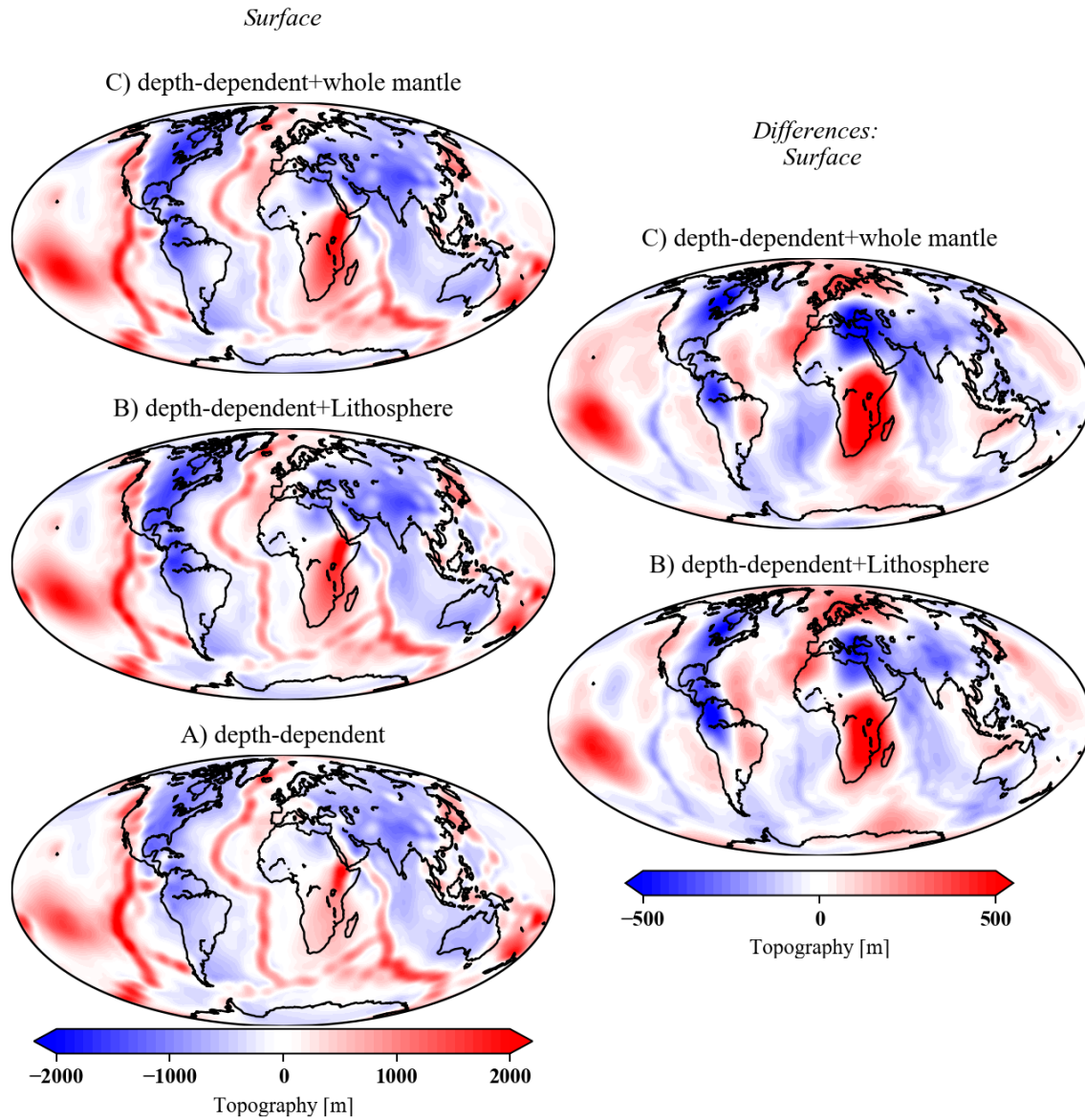


Figure 5.4: (Left column) Surface dynamic topography for Case: A - assuming an only radially varying viscosity. B - with the addition of viscosity variations due to varying thickness of the lithosphere. C - with a temperature-dependent viscosity structure, consistent with a creep law. (Right column) Differences in surface dynamic topography w.r.t Case A.

Fig. 5.4 (left column) shows the results. Fig. 5.4-A is the modeled dynamic topography in Case A, assuming only a depth-dependent viscosity. Mid-ocean ridges in the Indian, Pacific, and Atlantic oceans are marked with positive topography, where seismic tomography images asthenospheric warm upwellings. In the North-Atlantic realm, over Iceland, the surface is positively deflected, respectively by ~ 1000 m. The maximum positive dynamic topography is located over the Afar region in Africa, and Pacific super-plume, with an amplitude of ~ 2000 m. North and Central America are subsided by 1000 m, over the ancient Farallon subduction system. Fig. 5.4-B shows the modeled surface deflection in Case B, including lateral variations of viscosity due to lithospheric thicknesses. While the main features are

similar to Case B, details are different. The extent of positive topography over Afar is now extended southwards covering most of southeastern Africa, with amplitudes reaching up to 1000 m of change. The amplitudes are also raised over the Pacific Superswell. Northeastern Africa, as well as Scandinavia in case B, are uplifted by 500 m. Fig. 5.4-C is the computed dynamic topography in Case C, including an additional temperature-dependent viscosity in the lower-most mantle. The only minor difference with respect to Case B, are the slight increase of Pacific and African dynamic topography up to 2000 m.

Fig. 5.4 (right column) shows the difference in the modeled dynamic topography in Cases B and C, with respect to the reference Case A. In case B, there is an increase in the positive signal over southern Africa, reaching up to 700 m in western Zambia. In the North-Atlantic, there is an excess dynamic topography of 200 m around Scandinavia and north-western Africa. The same 200 m excess uplift is shown in the Pacific region. The subsidence over North-America and the Mediterranean is amplified by ~ 500 m, while the mid-ocean related uplift is 200 m less in amplitude. In case C however, the differences with respect to case B are minor. In case C, the uplift around Zimbabwe is amplified by 400 m w.r.t. case B. In the South-Atlantic, the mid-ocean ridge related uplift is now less by around 200 m.

5.5 Appendix: Supplementary Figures

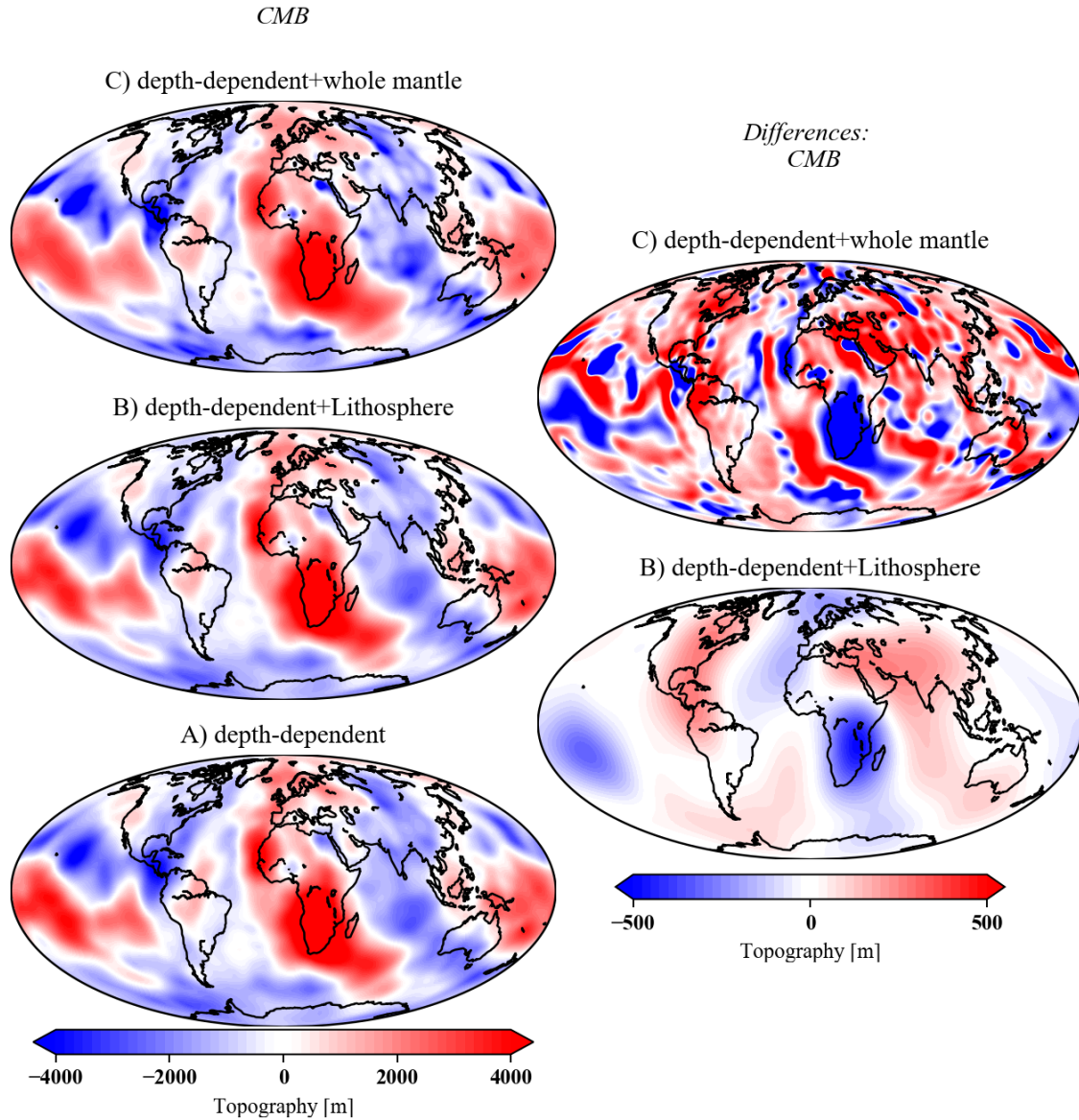


Figure 5.5: (Left column) CMB dynamic topography for Case: A - assuming an only radially varying viscosity. B - with the addition of viscosity variations due to varying thickness of the lithosphere. C - with a temperature-dependent viscosity structure, consistent with a creep law. (Right column) Differences in CMB dynamic topography w.r.t Case A.

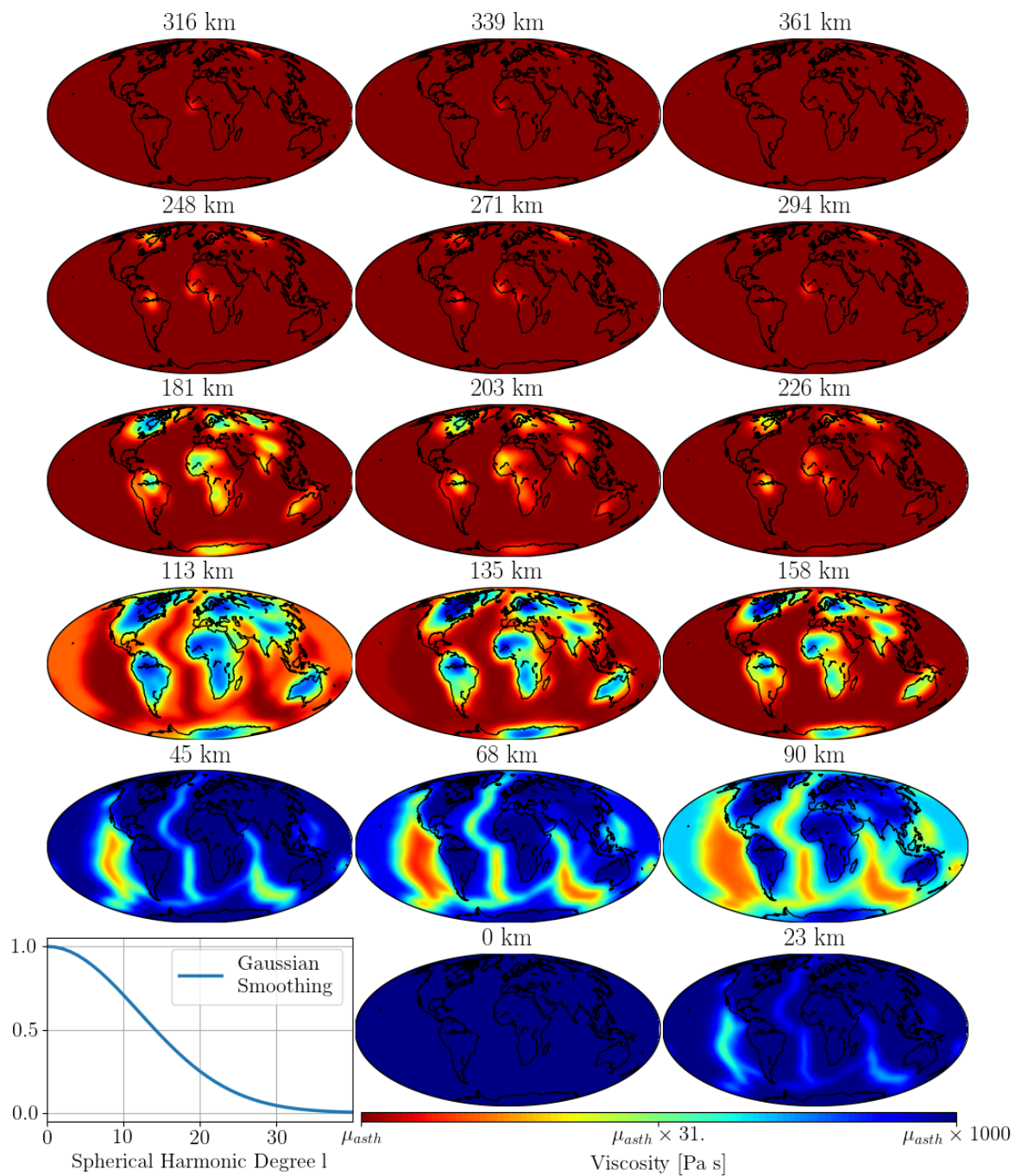


Figure 5.6: Lithospheric thicknesses

5.6 References

- [1] S. Bauer, M. Huber, S. Ghelichkhan, M. Mohr, U. Rde, and B. Wohlmuth, “Large-scale simulation of mantle convection based on a new matrix-free approach,” *Journal of Computational Science*, vol. 31, pp. 60 – 76, 2019.
- [2] J. Braun, “The many surface expressions of mantle dynamics,” *Nature Geoscience*, vol. 3, pp. 825–833, Nov 2010.
- [3] L. Colli, S. Ghelichkhan, and H.-P. Bunge, “On the ratio of dynamic topography and gravity anomalies in a dynamic Earth,” *Geophysical Research Letters*, vol. 43, no. 6, pp. 2510–2516, 2016.
- [4] B. H. Hager, R. W. Clayton, M. A. Richards, R. P. Comer, and A. M. Dziewonski, “Lower mantle heterogeneity, dynamic topography and the geoid,” *Nature*, vol. 313, pp. 541–545, feb 1985.
- [5] C. Thoraval and M. A. Richards, “The geoid constraint in global geodynamics: viscosity structure, mantle heterogeneity models and boundary conditions,” *Geophysical Journal International*, vol. 131, no. 1, pp. 1–8, 1997.
- [6] M. A. Richards and B. H. Hager, “Geoid anomalies in a dynamic Earth,” *Journal of Geophysical Research*, vol. 89, no. B7, p. 5987, 1984.
- [7] A. Paulson and M. A. Richards, “On the resolution of radial viscosity structure in modelling long-wavelength postglacial rebound data,” *Geophysical Journal International*, vol. 179, pp. 1516–1526, dec 2009.
- [8] M. J. Hoggard, J. Winterbourne, K. Czarnota, and N. White, “Oceanic residual depth measurements, the plate cooling model, and global dynamic topography,” *Journal of Geophysical Research: Solid Earth*, vol. 122, no. 3, pp. 2328–2372, 2017.
- [9] H. P. Bunge and U. A. Glasmacher, “Models and observations of vertical motion (MoveOn) associated with rifting to passive margins: Preface,” *Gondwana Research*, vol. 53, pp. 1–8, 2018.
- [10] C. Burstedde, G. Stadler, L. Alisic, L. C. Wilcox, E. Tan, M. Gurnis, and O. Ghattas, “Large-scale adaptive mantle convection simulation,” *Geophysical Journal International*, vol. 192, no. 3, pp. 889–906, 2013.
- [11] S. Zhong, A. McNamara, E. Tan, L. Moresi, and M. Gurnis, “A benchmark study on mantle convection in a 3-D spherical shell using CitcomS,” *Geochem. Geophys. Geosyst.*, vol. 9, p. Q10017, 2008.
- [12] C. Hüttig, N. Tosi, and W. B. Moore, “An improved formulation of the incompressible navier-stokes equations with variable viscosity,” *Physics of the Earth and Planetary Interiors*, vol. 220, pp. 11 – 18, 2013.
- [13] B. H. Hager and R. J. O’Connell, “A simple global model of plate dynamics and mantle convection,” *J. Geophys. Res.*, vol. 86, pp. 4843–4878, 1981.

-
- [14] N. A. Simmons, S. C. Myers, G. Johannesson, E. Matzel, and S. P. Grand, “Evidence for long-lived subduction of an ancient tectonic plate beneath the southern Indian Ocean,” *Geophysical Research Letters*, vol. 42, pp. 9270–9278, Nov 2015.
- [15] L. Stixrude and C. Lithgow-Bertelloni, “Thermodynamics of mantle minerals - II. Phase equilibria,” *Geophysical Journal International*, vol. 184, no. 3, pp. 1180–1213, 2011.
- [16] F. Cammarano, S. Goes, P. Vacher, and D. Giardini, “Inferring upper-mantle temperatures from seismic velocities,” *Physics of the Earth and Planetary Interiors*, vol. 138, no. 3, pp. 197 – 222, 2003.
- [17] D. Andrault, N. Bolfan-Casanova, G. L. Nigro, M. A. Bouhifd, G. Garbarino, and M. Mezouar, “Solidus and liquidus profiles of chondritic mantle: Implication for melting of the earth across its history,” *Earth and Planetary Science Letters*, vol. 304, no. 1, pp. 251 – 259, 2011.
- [18] N. A. Haskell, “The motion of a viscous fluid under a surface load,” *Physics*, vol. 6, no. 8, pp. 265–269, 1935.
- [19] I. M. Artemieva, “Global 11 thermal model tc1 for the continental lithosphere: Implications for lithosphere secular evolution,” *Tectonophysics*, vol. 416, no. 1, pp. 245 – 277, 2006. The Heterogeneous Mantle.
- [20] U. Christensen, “Convection with pressure- and temperature-dependent non-newtonian rheology,” *Geophysical Journal of the Royal Astronomical Society*, vol. 77, no. 2, pp. 343–384, 1984.

Chapter 6

Outlook

In this thesis we proposed two novel approaches to accelerate matrix-free finite element implementations. Both methods were implemented, profiled and optimized within the HHG framework. The first approach employs an appropriate scaling of the constant coefficient reference stencil, where the second one uses a polynomial surrogate operator. Originally developed to tackle two different problems, namely the inclusion of variable coefficients and the efficient assembly on curved domains, both approaches are similar in their nature. Both variants introduce a variational crime in the bilinear form, and if the coefficient is interpreted as blending function like in the example in section. 2.7.4, or vice versa, then both approaches can be transformed to each other. The difference is that the LSQP approach relies on a two-scale approximation, namely macro mesh size H and fine grid size h , the scaling approach is based on the fine scale h only.

For the scaling approach (chapter 2) a detailed mathematical analysis is presented and shows that the obtained solutions have asymptotically optimal order convergence in the H^1 - and L^2 -norm. The advantage of this approach compared to the original implementation is the $3\times$ reduction of floating-point operations while the memory traffic remains the same. Thus, this reduction directly transfers to $3\times$ faster time-to-solution. Initially proposed for a scalar operator, the next step is to extend this method to the Stokes system. Here additional complexities arise due to the cross-coupling terms that need special treatment.

The two-scale approach (chapter 3), also termed LSQP, replaces the expensive on-the-fly stencil assembly procedure by a low-cost polynomial evaluation. It was demonstrated that for a suitable choice of the macro scale H and fine scale h this approximation does not violate the L^2 approximation order of the discretization error. Furthermore, the implementation was optimized using advanced performance models. Weak and strong scaling results show that it ultimately recovers the performance of the original HHG implementation, but on a significantly more complex domain. We highlight that this approach is not limited to the spherical shell geometry, and can be further expanded by using higher polynomial orders to approximate even more complicated geometries.

The applicability of the LSQP method for real-world geophysical applications was demonstrated in chapters 4 and 5. First, the scheme was extended to the full Stokes system and its performance and scalability was extensively tested and profiled on up to $\sim 50\,000$ compute cores. In chapter 5 we used this approach to study the effect of lateral viscosity variations on dynamic topography. These simulations were performed at unprecedented global resolution close to 1 km.

To extend the scope of this study, one could also include non-linear rheologies in instantaneous models of the Earth's mantle. This requires an outer solver where the polynomial coefficients must be updated for each iteration. But as we have shown, this does not pose a challenge for our LSQP approach as the polynomial setup only takes a fraction of the time of one multigrid cycle. Another interesting aspect would be to use this approach for the computationally intense adjoint simulations which will reduce run-times significantly. In particular, our investigations of dynamic topography should be combined with retrodictions from adjoint simulations. We already found an increase of amplitudes of $\sim 25\%$ up to 50% , where the dynamics of a time-dependent model can further amplify these amplitudes. Furthermore, dynamic flow lines can significantly change the results, e.g. restored thermal anomalies may appear at different locations, and the sign of the retrodicted dynamic topography can be altered.

Ultimately, the examples in this thesis demonstrate that this approach is perfectly suited for the targeted kind of simulations and should be an integral part of the newly developed TerraNeo software.

Acknowledgments

There are so many people I am deeply grateful to and without their contribution this work would never have been possible. I tried my very best to mention all of them in the following - but please forgive me if I happened to forget a name. Furthermore I emphasize that the listing is without any particular order.

First I want to sincerely thank my supervisor Hans-Peter Bunge for your guidance and assistance throughout my doctorate and for showing me the beauty of applied geoscience. Also, I heartily thank my co-supervisor Marcus Mohr for your proficient advice and consecutive support and for always being around when necessary.

Moreover, I want to thank our TerraNeo partners from TU and FAU. In particular Barbara Wohlmuth for all the fruitful discussions and for always highlighting the details that make e.g. the difference between $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$ convergence. Also I thank Markus H. for the close collaboration and our (mostly) very productive coding sessions. And not to forget the times where we had a drink, or two, at certain conferences. Furthermore many thanks to Daniel, e.g. for our journey through the darkness of cache hierarchies.

Likewise I am very grateful to Ulrich Rude for your guidance in the world of HPC and for hosting me in the foreign land of Franconia for a couple of months. I also want to thank all the guys from LSS: Dominik B., Dominik T., Nils, Sebastian, Christoph, Julian, Christian, Iris, and all others. I had an incredible time in Erlangen, especially during *Berg*-time.

Not to forget our former project members Christian, Bjrn, Lorenz and, of course my former office mate Jens for your assistance in my early days as a PhD student, in particular with HHG. Also I thank Markus W. from RRZE for introducing me into the art of performance modeling.

Furthermore, I would like to thank all former and current members of our geodynamics group if I haven't mentioned them so far: Lorenzo, Bernhard, Rainer, and many others. Last but not least, Sia many thanks for being a perfect office mate and for answering all my stupid questions about geophysics. And of course for the many productive discussions about football.

There are some people that mostly work behind the scenes but to whom I am deeply thankful. Yvonne and Elena, many thanks for keeping track of my contracts and for your assistance with all the nifty details of bureaucracy. Jens and Gerald and all other guys from the IT team for providing and maintaining the IT infrastructure and for your help whenever I had some Linux problems with my laptop. But most importantly, Jens and Jana, many thanks for the delicious coffee.

Next, I want to express my gratitude to the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) of the DFG for providing most of the funding. Also I acknowledge the Gauss Centre for Supercomputing (GCS) for providing computing time on the supercomputers SuperMUC at LRZ and Hazel Hen at HLRS.

Finally, nothing of this would have been possible if it were not for Andrea and my family. They were the persons that initially motivated me to take on the challenge of a doctorate and kept on supporting me throughout the last couple of years.