

High accuracy mantle convection simulation through modern numerical methods

Martin Kronbichler,¹ Timo Heister² and Wolfgang Bangerth²

¹Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden.

²Department of Mathematics, Texas A&M University, College Station, TX 77843-3368. E-mail: bangerth@math.tamu.edu

Accepted 2012 July 11. Received 2012 July 10; in original form 2011 November 1

SUMMARY

Numerical simulation of the processes in the Earth's mantle is a key piece in understanding its dynamics, composition, history and interaction with the lithosphere and the Earth's core. However, doing so presents many practical difficulties related to the numerical methods that can accurately represent these processes at relevant scales. This paper presents an overview of the state of the art in algorithms for high-Rayleigh number flows such as those in the Earth's mantle, and discusses their implementation in the Open Source code ASPECT (Advanced Solver for Problems in Earth's ConvecTion). Specifically, we show how an interconnected set of methods for adaptive mesh refinement (AMR), higher order spatial and temporal discretizations, advection stabilization and efficient linear solvers can provide high accuracy at a numerical cost unachievable with traditional methods, and how these methods can be designed in a way so that they scale to large numbers of processors on compute clusters.

ASPECT relies on the numerical software packages DEAL.II and TRILINOS, enabling us to focus on high level code and keeping our implementation compact. We present results from validation tests using widely used benchmarks for our code, as well as scaling results from parallel runs.

Key words: Numerical solutions; Numerical approximations and analysis; Non-linear differential equations; Dynamics: convection currents, and mantle plumes.

1 INTRODUCTION

Computer simulation has been an important tool in studying the Earth's mantle owing to its inaccessibility to direct measurements. Consequently, deriving mathematical models and their numerical solution on computers has a long history dating back several decades. Comparing predictions from such models with indirect information about mantle properties (e.g. thermal fluxes, glacial rebound or the shape of the geoid) has provided an enormous amount of insight into the structure and mechanisms driving convection in the mantle. Similar computations have also been used to model other bodies in the solar system.

However, numerical predictions can only be as good as both the mathematical model and the numerical method used to solve it. To this end, more numerical methods have been proposed than we could attempt to summarize here, and a number of well-supported codes implementing the more successful methods have been published under licenses that have allowed their wide usage in the community. Citcom (Moresi *et al.* 1996) and Conman (King *et al.* 1990) are two examples of such codes, and both are now in fact at least in part maintained by the NSF-funded community initiative Computational Infrastructure in Geodynamics (CIG).

While highly successful, both of these codes as well as most others that are in use throughout the community have their roots

in numerical methods that were state of the art in the 1980s and early 1990s. For example, they use fixed meshes, low order finite elements, and—measured by today's standards—relatively simple solver and stabilization methods. Acknowledging the difficulty of retrofitting existing codes to new mathematical methods, and with support from CIG, we are therefore implementing a new code for mantle convection from scratch that incorporates the progress that has been made in numerical methods and computational science over the past 20 yr. Unlike other efforts that focus on a single part of a simulator (e.g. the solver, the advection scheme or the mesh), our intention in this work is to provide a code that uses current technology in every one of its components. This code, which we call ASPECT (short for *Advanced Solver for Problems in Earth's ConvecTion*) is intended as a modular program that can serve as the basis for both further method development and model refinements, as well as for easy modification to adjust for use in production simulations by the community at large. It is available under an Open Source license at <http://www.dealii.org/aspect/>.

In this paper, we summarize the current state of the art in numerical methods and computational science for problems of the kind that appear in the simulation of convection in the Earth's mantle, and give an overview of the methods implemented in ASPECT. Specifically, we address the following interconnected topics and the strategies used in our code.

(i) *Mesh adaptation.* Mantle convection problems are characterized by widely disparate length scales (from plate boundaries on the order of kilometres or even smaller, to the size of the entire Earth). Uniform meshes can not resolve the smallest length scale without producing an intractable number of unknowns. Fully adaptive meshes allow resolving local features of the flow field without the need to refine the mesh globally. Since the location of plumes that require high resolution change and move with time, meshes also need to be adapted every few time steps.

(ii) *Accurate discretizations.* The Boussinesq problem upon which most models for the Earth's mantle are based has a number of intricacies that make the choice of discretization non-trivial. In particular, the finite elements chosen for velocity and pressure need to satisfy the usual compatibility condition for saddle point problems. This can be worked around using pressure stabilization schemes for low-order discretizations, but high-order methods can yield better accuracy with fewer unknowns and offer more reliability. Equally important is the choice of a stabilization method for the highly advection-dominated temperature equation. We will choose a nonlinear artificial diffusion method for the latter.

(iii) *Efficient linear solvers.* The major obstacle in solving the Boussinesq system is the saddle-point nature of the Stokes equations. Simple linear solvers and preconditioners can not efficiently solve this system in the presence of strong heterogeneities or when the size of the system becomes very large. We will present an efficient solution strategy using a block triangular preconditioner based on an algebraic multigrid that provides optimal complexity even up to problems with hundreds of millions of unknowns.

(iv) *Parallelization of all of the steps above.* Global mantle convection problems frequently require extremely large numbers of unknowns for adequate resolution in 3-D simulations. The only realistic way to solve such problems lies in parallelizing computations over hundreds or thousands of processors. This is made more complicated by the use of dynamically changing meshes, and it needs to take into account that we want to retain the optimal complexity of linear solvers and all other operations in the program.

(v) *Modularity of the code.* A code that implements all of these methods from *scratch* will be unwieldy, unreadable and unusable as a community resource. To avoid this, we build our implementation on widely used and well tested libraries that can provide researchers interested in extending it with the support of a large user community. Specifically, we use the DEAL.II library (Bangerth *et al.* 2007, 2012) for meshes, finite elements and everything discretization related; the TRILINOS library (Heroux *et al.* 2005, 2012) for scalable and parallel linear algebra; and P4EST (Burstedde *et al.* 2011) for distributed, adaptive meshes. As a consequence, our code is freed of the mundane tasks of defining finite element shape functions or dealing with the data structures of linear algebra, can focus on the high-level description of what is supposed to happen, and remains relatively compact at currently only around 1400 lines for its core functionality. The code will also automatically benefit from improvements to the underlying libraries with their much larger development communities. Our code is extensively documented to enable other researchers to understand, test, use, and extend it.

It is our hope that the code finds adoption in the mantle convection community. This publication is intended as an overview of the numerical methods considered state-of-the-art today and that are implemented in ASPECT.

In the following sections, we will discuss the various parts of developing a modern implementation of a mantle convection simulator. Specifically, in Section 2 we outline the mathematical formu-

lation of the problem in the form of the Boussinesq approximation. Section 3 discusses the numerical methods used for time discretization, spatial discretization and stabilization of the temperature equation, the linear solvers and preconditioners, and parallelization issues. Section 4 shows numerical results obtained with the code and the results of benchmark problems. Section 5 draws conclusions and gives an outlook to further questions.

2 FORMULATION OF THE PROBLEM

Convection processes in the Earth's mantle are well described by incompressible fluid flow driven by temperature-induced small density differences. Since viscous friction forces in the fluid are large compared to buoyancy forces, the motion is slow and inertial terms can be neglected (Schubert *et al.* 2001). This yields the Boussinesq model, given by the following set of partial differential equations

$$-\nabla \cdot (2\eta\varepsilon(\mathbf{u})) + \nabla p = \rho(T)\mathbf{g}, \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2)$$

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) = \gamma. \quad (3)$$

In this equation, \mathbf{u} denotes the fluid velocity, p the pressure,¹ and T the temperature. η is the viscosity of the material, and $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ is the rate-of-deformation or strain rate tensor. The parameters κ , γ and \mathbf{g} are the thermal conductivity, heat sources and gravity vector, respectively. The simplest approximation for the temperature dependent density $\rho(T)$ is to use the relationship

$$\rho(T) = \rho_{\text{ref}} (1 - \beta(T - T_{\text{ref}})),$$

where ρ_{ref} is the reference density at reference temperature T_{ref} , and β is the thermal expansion coefficient. However, while ASPECT can use this approximation as a particular density model, it internally uses whatever form is given for this dependence. The equations given above need to be augmented by appropriate boundary conditions for the velocity (or pressure) and temperature, as well as initial conditions for T .

This relatively simple system can be non-dimensionalized by introducing the Rayleigh number (see e.g. Zhong *et al.* 2008) to simplify the analysis of these equations. ASPECT does not automatically do this. Instead, it uses the material, geometry and gravity descriptions provided by user-defined modules which may either be given in dimensional physical units (such as the results of Section 4.6) or in non-dimensional units (e.g. the benchmarks given in Sections 4.1–4.4). In any case, ASPECT does make sure that internally the numerical equations are well-balanced (see Section 3.2.4) to avoid the problems with round-off one usually encounters when implementing dimensional equations naively, while at the same time ensuring that all externally visible quantities are in the same units as those given in the model description. We recognize that it is uncommon in geodynamics to work with dimensional quantities; at the

¹ Note that the pressure in this formulation is the *total* pressure, that is, the sum of static and dynamic pressures. Solving for the total pressure is uncommon in most geodynamics codes in an effort to avoid round-off problems. We avoid these problems by using other means as discussed in the Appendix. On the other hand, ASPECT of course allows problem descriptions in which the density on the right hand side is logically only a density *variation*, that is, a deviation from a static value, as we do in the benchmarks shown in Sections 4.1–4.4. In this case, the pressure component of the solution is only the dynamic pressure.

same time, we believe that a well-implemented code can avoid the typical problems while at the same time making sure that all user interaction happens in user-defined units, thus avoiding a common source of errors, for example, in table lookup of experimentally determined material parameters. Furthermore, working in dimensional units avoids the ambiguity of how the non-dimensionalization should happen when coefficients such as gravity, viscosity and density are no longer constant and, consequently, there is no single, easily defined Rayleigh number that holds throughout the entire domain.

We note that for a realistic description of the Earth, several of the coefficients depend on the solution variables. For example, the viscosity η generally decreases with rising temperature T and depends on the strain rate; both viscosity η and density ρ depend on the current location in the p - T phase diagram; and the heating term γ will contain not only radiogenic heating but also adiabatic heating and the viscous dissipation $\eta \varepsilon(\mathbf{u}) : \varepsilon(\mathbf{u})$. Many other factors, for example, inhomogeneous chemical composition or phase changes, also enter any attempt at complete descriptions. We will not take these effects into account here, but we will discuss some of the associated issues at the end of this paper and in a future contribution.

3 NUMERICAL METHODS

Eqs (1)–(3) are not easy to solve numerically. To be efficient, an algorithm has to take into account a number of interconnected issues related to time stepping, spatial discretization and linear solvers, none of which can be considered entirely on their own.

As mentioned in the Introduction, the purpose of this paper is to describe a coherent set of methods for time discretization, adaptive meshing, spatial discretization, parallelization and optimal linear solvers and preconditioners that together yield accurate solutions at optimal cost and that enable the numerical simulation even of processes that were previously considered too difficult, or in a fraction of the time for problems that are typically considered expensive.

In the following subsections, we will present the various building blocks of our approach. An open source implementation of these ideas is available through the extensively documented step-31 and step-32 tutorial programs (Kronbichler & Bangerth 2011; Kronbichler *et al.* 2011) of the widely used finite element library DEAL.II (Bangerth *et al.* 2007, 2012), and the continued development of these programs in the form of ASPECT. The numerical results in Section 4 were obtained with only slightly modified versions of these tutorial programs and are therefore easily replicable. We mention here that the code supports both 2-D and 3-D computations, obviating the need to develop and debug two separate versions of the methods, and enabling the ability to test the 2-D version before switching to production runs in 3-D.

3.1 Time discretization

The primary complications of the Boussinesq system (1)–(3) with regard to the time discretization are (i) the nonlinear coupling of all components; and (ii) the fact that the Stokes equations for \mathbf{u} , p do not contain time derivatives and consequently form the equivalent of an algebraic (instantaneous) constraint to the temperature equation that does have a time derivative. The result of these complications are that simple and cheap time marching schemes are not possible. A large number of schemes have been proposed and used over the past decades to approximate solutions to the Boussinesq equations (for recent discussions of methods see Ismail-Zadeh & Tackley 2010; Gerya 2010). However, while appropriate at the time, most

of them would not be considered highly accurate or highly efficient by today's standards.

The Stokes equation can be considered as a constraint to the temperature equation that has to hold at any given time in general, and at time steps in particular. Time dependent differential equations of this kind are frequently solved using time stepping methods akin to the IMPES (*implicit pressure explicit saturation*) schemes originally developed for porous media flow simulations (Sheldon *et al.* 1959; Stone & Garder 1961; Chen 2006). In these methods, the variables defined by the constraint are computed from the equations without time derivatives. Here, these are velocities and pressure, and since a linear system needs to be inverted, the step is considered implicit. In a second substep of the original IMPES scheme, the other variables are then updated using an explicit time step. The IMPES approach allows to decouple the nonlinear Boussinesq system into two simpler, linear subproblems, and therefore leads to an efficient scheme for the solution of the coupled problem.

Since one alternates between the two substeps, one can consider them in any order. Let us here first discuss the explicit temperature step and then the implicit Stokes solve. In the following, let t^n denote the time of the n th time step and $k_n = t^n - t^{n-1}$ denote the length of the n th time step. We will then write \mathbf{u}^n , p^n , T^n to indicate approximations of the velocity, pressure and temperature at time t^n .

To provide accurate convection dynamics, we approximate the time dependency in the temperature eq. (3) using a second-order accurate implicit/explicit time stepping scheme based on the BDF-2 scheme (Hairer & Wanner 1991). This scheme is a good compromise between high accuracy (which could be increased using higher-order schemes), stability (which typically decreases with the order of the scheme, requiring smaller CFL numbers and consequently higher computational effort) and efficiency of implementation (higher-order schemes often become unwieldy as they require complicated initialization during the first few time steps, and require the storage of many solution vectors from previous time steps). The BDF-2 scheme balances these issues well and leads to reasonable CFL numbers and an accuracy that is balanced with that of the spatial discretization that we will discuss in Section 3.2.

To derive the BDF-2 scheme, we use a quadratic interpolation to find the finite difference approximation of $\frac{\partial T}{\partial t}$ from times t^n , t^{n-1} , t^{n-2} as

$$\frac{\partial T(t^n)}{\partial t} \approx \frac{1}{k_n} \left(\frac{2k_n + k_{n-1}}{k_n + k_{n-1}} T(t^n) - \frac{k_n + k_{n-1}}{k_{n-1}} T(t^{n-1}) + \frac{k_n^2}{k_{n-1}(k_n + k_{n-1})} T(t^{n-2}) \right). \quad (4)$$

Using a Taylor series one can show that this approximation is correct up to second order (Hairer & Wanner 1991). The same formulas also hold for $\mathbf{u}(t^n)$, of course. The usual form in which these equations are stated in the literature is obtained by assuming that $k_n = k_{n-1}$, but we want to keep our formulas more general since we need to choose variable time step sizes to satisfy the CFL condition at each time step.

Taking into account the time step sizes k_n and k_{n-1} , we define the linearly extrapolated temperature $T^{*,n}$ as

$$T^{*,n} = \left(1 + \frac{k_n}{k_{n-1}} \right) T^{n-1} - \frac{k_n}{k_{n-1}} T^{n-2}, \quad (5)$$

and similarly for an extrapolated velocity $\mathbf{u}^{*,n}$.

We then arrive at a semi-implicit BDF-2 version of the temperature eq. (3) by using $T^{*,n}$, $\mathbf{u}^{*,n}$ in the advection term, treating the diffusion term implicitly, and using approximation (4) for the time

derivative

$$\begin{aligned} \frac{2k_n + k_{n-1}}{k_n + k_{n-1}} T^n - k_n \nabla \cdot \kappa \nabla T^n &= \frac{k_n + k_{n-1}}{k_{n-1}} T^{n-1} \\ &- \frac{k_n^2}{k_{n-1}(k_n + k_{n-1})} T^{n-2} - k_n \mathbf{u}^{*,n} \cdot \nabla T^{*,n} + k_n \gamma. \end{aligned} \quad (6)$$

We will discuss solving the discretized version of this equation for T^n in Section 3.3.1. Note that we treat physical heat conduction (diffusion) implicitly while the evaluation of convection and the artificial diffusion terms we will discuss below are made explicit by extrapolation. For a fixed convection this will retain unconditional stability, see (Quarteroni & Valli 1994, p. 411). Since solving the temperature equation does not take more than a few percent of the overall run time of Boussinesq solvers, making diffusion implicit is a useful compromise. Regardless of this detail, the whole scheme is not unconditionally stable, because we extrapolate the convection $\mathbf{u}^{*,n}$ in the temperature eq. (6).

The introduction of this explicit convection limits the time step by a Courant-Friedrichs-Lewy (CFL) condition (Quarteroni & Valli 1994). Specifically, after spatial discretization (see Section 3.2), the time step k_n must satisfy

$$\text{CFL}_K = \frac{k_n \|\mathbf{u}\|_{\infty,K}}{h_K} \leq C$$

on every cell K , for a constant C that depends on the particular time stepping method as well as the method used for spatial discretization and that is experimentally chosen as large as possible while ensuring that the solution remains stable. Here, h_K is the diameter of cell K , and $\|\mathbf{u}\|_{\infty,K}$ the maximal magnitude of the velocity on K . In our implementation, we have experimentally chosen $C = \frac{1}{5.9p}$ in 2-D and $C = \frac{1}{43.6p}$ in 3-D, where p is the polynomial degree with which we discretize the temperature variable, see Section 3.2.5. The difference between 2-D and 3-D results primarily from the different ratio between edge length and cell diameter h_K as well as from the larger distortion of cells in 3-D in the shell geometry we will be using in mantle convection simulations. We choose time steps k_n that satisfy this stability condition, which necessitates choosing them of variable length as in the formulas above.

We end the discussion of the time discretization with three remarks. First, one might believe that a fully implicit time discretization would allow larger time steps. However, since it is difficult to solve the temperature equation fully coupled with the Stokes equation, our use of the extrapolated velocity \mathbf{u}^* already limits the size of time steps. Furthermore, while fully implicit solutions may be stable for advection problems with a CFL number larger than C , they are typically rather inaccurate with large time steps. Secondly, the BDF-2 scheme requires knowledge of the solution at time instances t^{n-1} , t^{n-2} ; consequently, it can not be used for the very first time step and we initialize the scheme by a single, first-order accurate implicit Euler step. Since most geodynamics applications are not interested in the initial transient phase but the long-term behaviour, the reduced accuracy in the initial time step does in general not affect overall results. Finally, after solving for the temperature at time instant t^n using (6), we can compute an updated velocity \mathbf{u}^n using the Stokes system

$$\begin{aligned} -\nabla \cdot (2\eta \varepsilon(\mathbf{u}^n)) + \nabla p^n &= \rho(T^n) \mathbf{g}, \\ \nabla \cdot \mathbf{u}^n &= 0. \end{aligned} \quad (7)$$

Since these equations do not have time derivatives, no special time discretization is necessary here.

3.2 Spatial discretization and stabilization

In each time step, we now first have to solve (6) for the updated temperature T^n , then (7) for the new velocity and pressure. To do so, we need to spatially discretize these equations, for which we use the finite element method.

As with time stepping, spatial discretization raises a number of interconnected issues: (i) What kind of mesh should we choose? (ii) What kind and order of finite elements should we use for the temperature, velocity and pressure variables? (iii) How can we stabilize the solution of the discrete temperature equation to avoid unphysical oscillations in regions where the temperature has strong gradients? We will discuss these issues in turn in the following.

3.2.1 Choice of meshes and local adaptation

With few and mostly recent exceptions (see, e.g. Albers 2000; Davies *et al.* 2007a,b; Burstedde *et al.* 2008, 2009; Stadler *et al.* 2010; Leng & Zhong 2011), mantle convection applications have used meshes that are either obtained by uniform refinement of a coarse mesh, or are obtained from a mesh generator. In either case, the mesh is fixed. In contrast, we will here use a mesh that can be dynamically adapted by local adaptive refinement and coarsening of an initial mesh with a small number of cells. This gives us flexibility to improve mesh resolution close to specific features of the solution, for example, strong temperature gradients, and thereby increase the accuracy of the solution. A different view of this adaptive mesh refinement (AMR) technique is that the mesh is a selectively coarsened version of a uniformly refined one where coarsening happens in parts of the volume where the solution is smooth. This notion supports the view that adaptively refined meshes provide about the same overall accuracy as a uniformly refined mesh with the same minimal mesh size, but at a fraction of the numerical cost. Evidence from the more mathematically oriented literature (see, e.g. Bangerth & Rannacher 2003; Ainsworth & Oden 2000; Babuška & Strouboulis 2001) shows consistently that AMR can achieve levels of accuracy typically required in engineering applications with a factor of around 100 (in 2-D) or 1000 (in 3-D) less computational effort than uniformly refined meshes, and we will confirm this observation in Section 4.2. Convection problems are certainly a prime candidate for savings of this order of magnitude given that the temperature and accompanying flow features frequently vary on length scales of only a few kilometres, much smaller than the size of Earth as a whole. Using adaptively refined meshes is, therefore, a crucially important factor in making highly accurate simulations of complex problems possible at all.

There are a number of practical aspects to using AMR. First, the underlying software is unsurprisingly much more complex than when one wants to use a fixed mesh. Our work is based on a large finite element library, DEAL.II, that already provides this functionality at little additional effort to the implementer of a code, for quadrilaterals in 2-D and hexahedra in 3-D. Secondly, one has to deal with the fact that if neighbouring cells differ in refinement level, some of the nodes of the mesh lie on the midpoints of edges or faces of neighbouring cells. We deal with this through constraints that ensure that the solution remains continuous at these hanging nodes, see (Babuška & Rheinboldt 1978; Carey 1997; Bangerth *et al.* 2007; Bangerth & Kayser-Herold 2009). Finally, we dynamically adapt the mesh every few time steps and we need a criterion to decide which cells to refine or coarsen. Since the variable that is most indicative of abruptly changing features of the solution is the temperature, we apply a criterion to the temperature that is

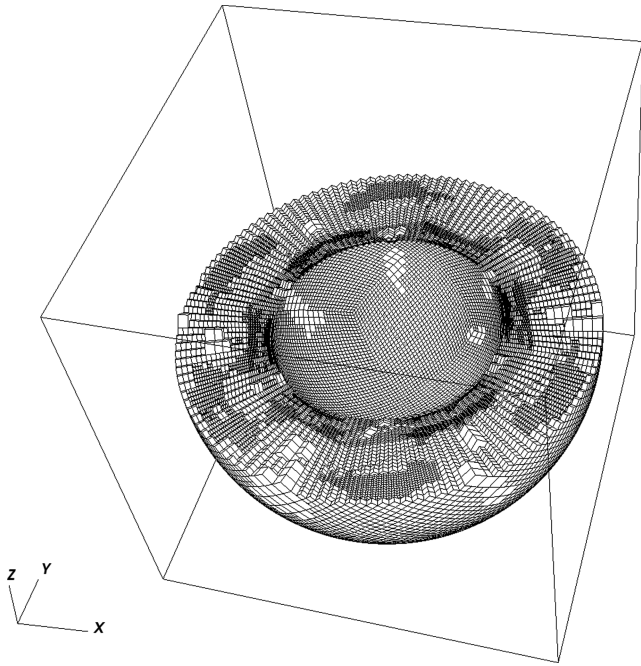


Figure 1. Example of a locally refined mesh. One half of the mesh for the spherical shell geometry in 3-D is shown. The mesh has approximately 890 000 cells; the finest cells are six times refined from the coarse mesh.

commonly referred to as the ‘Kelly error indicator’ (Gago *et al.* 1983) and that computes for each cell an approximation of the size of the second derivatives times the diameter of the cell. This criterion has been found to be a simple, yet universally useful tool in adaptively refining meshes, and is implemented in DEAL.II. While this works well in 2-D, in 3-D it refines primarily into the boundary layers at the inner and outer margins of the mantle; we avoid this by also taking derivatives of the velocity into account when refining. Ultimately, however, finding a good and universal refinement criterion turns out to be non-trivial; we will return to this issue in a future paper.

Even though we adapt the mesh every few time steps, we limit the number of times a single coarse mesh cell can be refined: otherwise, close to steep gradients, the cells would be made smaller and smaller in each refinement step, requiring smaller and smaller global time steps due to the CFL condition.

While implementing the data structures and algorithms outlined above from scratch would require several tens of thousands of lines of codes, they are all readily available in DEAL.II. In fact, using adaptive meshes and related algorithms requires little more than maybe a dozen lines of code in our program. An example of the kind of meshes we use here is shown in Fig. 1 in 3-D; 2-D meshes are shown in Figs 3 and 7 below.

3.2.2 Approximation of geometry

Our program uses Cartesian coordinates. The advantage of this choice is that the shell geometry of the Earth’s mantle is not a hard-coded special case, but no different than any other geometry (it is simply produced by using a mesh consisting of an unstructured collection of coarse cells which are then hierarchically refined) and the code can readily be adapted to use a box geometry (as used in some

of the examples in Section 4), an octant of the shell, or a domain that takes into account the geoid shape or actual topography—none of these is any more difficult than any other, and we need not modify the assembly of matrices or vectors when changing between coordinate systems.

To deal with curved boundaries, one has to map the finite element shape functions discussed below from the reference cell on which they are defined to the location of cells in the unstructured mesh. Traditionally, this is done using polynomial mappings, often chosen to be isoparametric, that is, of the same polynomial degree as the shape functions (Brenner & Scott 2002; Carey 1997), though DEAL.II allows these to be chosen independently. Because the overhead of using a higher-order mapping is negligible compared to the many other operations in a program (the higher-order mapping only leads to higher numerical cost in cells at the surface of the domain), we use a mapping of degree four.

3.2.3 Spatial approximation of the flow variables

On the meshes as described earlier, we discretize all variables using the finite element method, that is, we seek to find approximations for \mathbf{u}^n , p^n , T^n of the form

$$\begin{aligned} \mathbf{u}_h^n(\mathbf{x}) &= \sum_{j=1}^{N_u} U_j^n \varphi_j^n(\mathbf{x}), \\ p_h^n(\mathbf{x}) &= \sum_{j=1}^{N_p} P_j^n \varphi_j^p(\mathbf{x}), \\ T_h^n(\mathbf{x}) &= \sum_{j=1}^{N_T} T_j^n \varphi_j^T(\mathbf{x}). \end{aligned} \quad (8)$$

There are a number of choices for the finite element basis functions φ_j^u , φ_j^p , φ_j^T . Since we will want to match the polynomial degree of the functions for the temperature to those of the velocity, let us here first discuss the choice for the flow variables.

For the Stokes system (7), it is well known that the polynomial degrees of shape functions for the velocity and the pressure can not be chosen arbitrarily (see, e.g. Brenner & Scott 2002; Braess 1997). Rather, one either has to stabilize the Stokes equations, for example, by adding an artificial compressibility term, or by choosing a pair of finite element spaces that satisfy the Babuška-Brezzi (or LBB or inf-sup) condition.

Many homegrown codes use artificial compressibility (or some other) stabilization because it avoids the need to implement shape functions of different polynomial degrees. However, it is known that the resulting solutions are not very accurate; furthermore, these schemes can not easily guarantee mass conservation. Alternatively, codes may choose to use lowest-order elements with piecewise constant pressures; these are locally conservative but have a very low order of accuracy. For these reasons, we choose to use finite element spaces that are known to be LBB-stable (Girault & Raviart 1986) and have at least a piecewise linear pressure space. Specifically, we consider the following two options that are already implemented in DEAL.II:

- (i) $\mathcal{Q}_{q+1}^d \times \mathcal{Q}_q$, ($q \geq 1$): This choice uses continuous shape functions of polynomial degree $q + 1$ for each of the d velocity components, and continuous shape functions of polynomial degree q

for the pressure.² This combination is known as Taylor-Hood elements. The fact that we use a lower polynomial degree for the pressure is not usually a concern since one is not typically interested in highly accurate pressure fields anyway. Furthermore, the pressure is a globally smooth function and almost entirely dominated by the hydrostatic pressure that essentially determines the lookup of pressure-dependent material properties.

(ii) $Q_{q+1}^d \times P_{-q}$, ($q \geq 0$): This choice differs from the one above in that it uses discontinuous elements of polynomial degree q for the pressure and that it omits the tensor product shape functions from the polynomial space. A particular case is the $Q_1^d \times P_0$ element with piecewise constant pressures that is, for example, used in Citcom (Moresi *et al.* 1996).

In either case, one will typically choose $q = 1$, that is, linear elements for the pressure and quadratic ones for the velocity, which results in $27 \times 3 = 81$ velocity nodes per cell in 3-D, and eight or three pressure nodes for the Q_1 and P_{-1} elements, respectively.³ This gives formal third-order accuracy for the velocity variable and second order for the pressure. Choosing larger values for q , as is possible through ASPECT's input file, would yield higher orders of convergence, but non-smooth regions in the solution often limit the global accuracy so that the additional work does not pay off. We will investigate these various choices quantitatively in Section 4.1.

Which of the two options above for the pressure space is preferable is not immediately obvious. It is easy to show that the second implies local mass conservation on each cell, that is, $\int_{\partial K} \mathbf{n} \cdot \mathbf{u}_h^n = 0$ for every cell K of the mesh. On the other hand, this does not necessarily yield a smaller overall error and, in fact, simple experiments show that the pointwise values of the divergence of \mathbf{u}_h^n are in fact larger for the second choice than for the first. A different consideration is that the second choice has approximately d times more pressure variables than the first although as we will see in Section 3.3.2, this does not result in a significantly higher computational effort. To facilitate experiments, our implementation allows to choose either element based on a run-time parameter.

3.2.4 Weak form and fully discrete Stokes system

The coefficients U_j , P_j of the expansion (8) are determined by inserting \mathbf{u}_h^n , p_h^n into the Stokes system (7), multiplying the equations with test functions $\varphi_i^u(\mathbf{x})$, $\varphi_j^p(\mathbf{x})$, respectively, and integrating over the domain. Integrating these terms by parts and using the appropriate boundary conditions on φ_i^u and \mathbf{u}_h^n then yields the weak form of the discrete equations:

$$\begin{aligned} (\varepsilon(\varphi_i^u), 2\eta\varepsilon(\mathbf{u}_h^n))_\Omega - (\nabla \cdot \varphi_i^u, p_h^n)_\Omega &= (\varphi_i^u, \rho(T_h^n)\mathbf{g})_\Omega, \\ (\varphi_j^p, \nabla \cdot \mathbf{u}_h^n)_\Omega &= 0. \end{aligned}$$

² Here and in the following, the finite element space Q_q is generated by mapping complete tensor polynomial spaces from the reference cell to each cell. For example, in 2-D the space Q_1 consists of the bilinear functions 1 , ξ , η , $\xi\eta$, where ξ , η are the coordinates on the reference cell. In contrast, the space P_q consists only of polynomials of maximum degree q . In 2-D, P_1 consists of the functions 1 , ξ , η . Using a negative index, P_{-q} , indicates that functions do not need to be continuous across cell interfaces. See also (Elman *et al.* 2005, section 5.3) for a discussion of element spaces suitable for the Stokes problem.

³ Note, however, that the eight pressure nodes per cell in 3-D for the Q_1 element are shared between all cells adjacent to each vertex, whereas the three pressure nodes per cell for the P_{-1} element are uniquely associated with each cell. Consequently, the P_{-1} pressure element has asymptotically d times as many unknowns.

Our goal is to find functions \mathbf{u}_h^n and p_h^n —that is, to find coefficients U_j^n , P_j^n —such that these equations hold for $i = 1 \dots N_u$, $l = 1 \dots N_p$. As presented here, these equations are unbalanced in their physical units since we have not non-dimensionalized them, and will have vastly different numerical values when using coefficients and length scales as common for Earth. While not a mathematical problem, it leads to severe inaccuracies for both linear and iterative solvers. We avoid these by multiplying the second of the two equations by a factor $s_p = \frac{\eta_0}{L}$ where L is a typical lengthscale of the problem and η_0 is a reference viscosity. We have found that it is best to choose L not as the diameter of the domain but as the size of typical features such as plumes in the Earth to approximate the effect of the missing second derivative in the second equation compared to the first. For example, for global convection problems, we choose $L = 10^4$ m. For problems with constant viscosity, $\eta_0 = \eta$. On the other hand, when solving a problem with variable viscosity, η_0 should be a value that adequately represents the order of magnitude of the viscosities that appear, such as an average value or the value one would use to compute a Rayleigh number.

After multiplying the second equation by s_p , the system is no longer symmetric. We can restore symmetry by replacing the pressure by $p_h^n = s_p \bar{p}_h^n$ and solving for the new variable \bar{p}_h^n with expansion coefficients \bar{P} instead. Note that \bar{p} does not have the physical units of a pressure, but it has the intuitive meaning of a pressure. We obtain the original pressure immediately after solving by multiplying the second component of the solution vector by s_p . With all this, the fully discrete version of the Stokes equations at time step n now reads

$$\begin{aligned} (\varepsilon(\varphi_i^u), 2\eta\varepsilon(\mathbf{u}_h^n))_\Omega - s_p(\nabla \cdot \varphi_i^u, \bar{p}_h^n)_\Omega &= (\varphi_i^u, \rho(T_h^n)\mathbf{g})_\Omega, \\ s_p(\varphi_j^p, \nabla \cdot \mathbf{u}_h^n)_\Omega &= 0, \end{aligned} \quad (9)$$

and we can rewrite these equations in matrix notation as

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} U^n \\ \bar{P}^n \end{pmatrix} = \begin{pmatrix} F^n \\ 0 \end{pmatrix}, \quad (10)$$

where

$$A_{ij} = (\varepsilon(\varphi_i^u), 2\eta\varepsilon(\varphi_j^u))_\Omega, \quad (11)$$

$$B_{ij} = -s_p(\varphi_i^p, \nabla \cdot \varphi_j^u)_\Omega, \quad (12)$$

$$F_i^n = (\varphi_i^u, \rho(T_h^n)\mathbf{g})_\Omega. \quad (13)$$

We will discuss solving this linear system in Section 3.3.2.

3.2.5 Spatial approximation of the temperature variable

Since the temperature structure of the Earth is one of the variables of primary interest in mantle convection computations, we are interested in an accurate representation. Consequently, choosing a higher-order approximation would seem promising. On the other hand, we can not expect the evolution of the temperature field to be more accurate than the velocity field along which it is primarily advected. Thus, we choose to approximate the temperature using the same polynomial degree $q + 1$ as the velocity, that is, $T_h^n \in Q_{q+1}$. Our experience is that Q_q elements yield considerably worse approximations in usual temperature fields, despite the fact that high-order methods are more prone to over- and undershoots in regions of high gradients (LeVeque 2002). We suppress these oscillations through an appropriate stabilization as discussed in the next section.

3.2.6 Stabilization of the temperature equation

Eq. (6) for the temperature at time step n is of advection-diffusion type. In mantle convection simulations, the diffusivity κ is very small compared to the velocity. Even for very fine meshes, the local Péclet number on cell K , $Pe_K = \frac{h_K |\mathbf{u}|_K}{\kappa}$ is usually in the range of 10^2 to 10^4 . For such high Péclet number problems, standard finite element discretizations introduce spurious oscillations around steep gradients (Donea & Huerta 2003). Therefore, stabilization must be added to the discrete formulation to obtain correct solutions.

One commonly used stabilization is to add artificial diffusion, either uniformly or, in the SUPG method, only along streamlines (Brooks & Hughes 1982). Such methods are used, for example, in the widely used Conman (King *et al.* 1990) and Citcom (Moresi *et al.* 1996) codes. While popular, these methods have the disadvantage that they add diffusion to any cell on which the local Péclet number is large, even in regions where the temperature is smooth and there is no danger of oscillations. We therefore adopt a more recent method, the so-called entropy viscosity method (Guermond *et al.* 2011), that only adds artificial diffusion where the local Péclet number is large and the solution is non-smooth. This method solves the modified temperature equation

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla \cdot (\kappa + \nu_h(T)) \nabla T = \gamma, \quad (14)$$

subject to the time discretization discussed in Section 3.1, with an artificial dissipation $\nu_h(T)$ added to the equation. Conceptually, in regions where the temperature field T is smooth ν_h should be small, and in regions with significant variability ν_h should be of similar size as the diffusive flux in a first-order upwind scheme (LeVeque 2002). This nonlinear definition of the artificial viscosity makes sure that the dissipation is as small as possible, while still large enough to prevent oscillations in the temperature field. In particular, the global approximation property of the scheme will not be affected, as would be the case with a simple linear dissipation with a constant ν_h .

Following (Guermond *et al.* 2011), on cell K we choose $\nu_h|_K$ as

$$\nu_h|_K = \min(\nu_h^{\max}|_K, \nu_h^E|_K). \quad (15)$$

The maximum dissipation ν_h^{\max} is defined as

$$\nu_h^{\max}|_K = \alpha_{\max} h_K \|\mathbf{u}\|_{\infty, K},$$

where the constant $\alpha_{\max} = 0.026d$ depends only on the spatial dimension d , and where h_K denotes the characteristic size of cell K . On the other hand, the entropy viscosity is defined as

$$\nu_h^E|_K = \alpha_E \frac{h_K^2 \|r_E(T)\|_{\infty, K}}{\|E(T) - E_{\text{avg}}\|_{\infty, \Omega}},$$

where we choose the constant $\alpha_E = 1$, see also the discussion in Guermond *et al.* (2011). The entropy viscosity is scaled globally by $\|E(T) - E_{\text{avg}}\|_{\infty, \Omega}$, based on the maximum deviation of the temperature entropy $E(T) = \frac{1}{2}(T - T_m)^2$, $T_m = \frac{1}{2}(T_{\min} + T_{\max})$ from the space-average $E_{\text{avg}} = \frac{1}{|\Omega|} \int_{\Omega} E(T)$. The residual $r_E(T)$ is associated with the entropy of the temperature equation,

$$r_E(T) = \frac{\partial E(T)}{\partial t} + (T - T_m)(\mathbf{u} \cdot \nabla T - \kappa \nabla^2 T - \gamma).$$

This residual is zero if applied to the exact solution of the temperature equation, leading to no artificial diffusion, but it is non-zero when applied to the numerical approximation we compute and will be large in areas where the numerical approximation is poor, such as close to strong gradients. We note that this definition of an artificial dissipation is similar to the $YZ\beta$ discontinuity capturing proposed

in Bazilevs *et al.* (2007), where the residual is based on the equation itself instead of the entropy, though.

In practice, we need to evaluate the formula above for the discrete solution. Since we do not want the artificial viscosity to introduce a non-linear dependence on the current temperature T^n , we make it explicit by approximating the time derivative from the two previous time levels in the BDF-2 time stepping, $\partial E(T^h)/\partial t \approx (E(T^{n-1}) - E(T^{n-2}))/k_{n-1}$, and evaluating all other occurrences of the temperature at the midpoint as $(T^{n-1} + T^{n-2})/2$, including the average temperature. We will treat the artificial viscosity term $-\nabla \cdot \nu_h(T^{n-1}, T^{n-2}) \nabla T^{*,n}$ as a whole explicitly, based on the extrapolated temperature defined in (5). Since the maximum artificial viscosity is proportional to the mesh size and the velocity, the CFL number is not changed substantially, which we have also verified numerically.

3.2.7 Fully discrete temperature system

In the same way as for the Stokes equations, we obtain the fully discrete linear system corresponding to the time-discretized temperature eq. (6)

$$\left(\frac{2k_n + k_{n-1}}{k_n + k_{n-1}} M + k_n K \right) T^n = G^n, \quad (16)$$

where $M_{ij} = (\varphi_i^T, \varphi_j^T)_{\Omega}$ is the mass matrix, and $K_{ij} = (\kappa \nabla \varphi_i^T, \nabla \varphi_j^T)_{\Omega}$ the stiffness matrix. The right hand side term G^n contains all terms from previous time levels,

$$G_i^n = \left(\varphi_i^T, \frac{k_n + k_{n-1}}{k_{n-1}} T^{n-1} - \frac{k_n^2}{k_{n-1}(k_n + k_{n-1})} T^{n-2} \right)_{\Omega} + (\varphi_i^T, -k_n \mathbf{u}^{*,n} \cdot \nabla T^{*,n} + k_n \gamma)_{\Omega} - (\nabla \varphi_i^T, k_n \nu_h \nabla T^{*,n})_{\Omega},$$

where the artificial viscosity ν_h is defined by (15) and constant within a cell K , and the extrapolated values for temperature and velocity are according to (5).

3.3 Linear solvers

Applying temporal and spatial discretization to the Boussinesq system leads to the two linear equation systems (16) and (10) that need to be solved in each time step. To accurately represent problems in geodynamics, this leads to large systems with up to hundreds of millions or billions of unknowns for which the only realistic choice are iterative solvers (Saad 2003). We discuss our choices in the following sections.

3.3.1 Temperature system

Solving the temperature system (16) is relatively straightforward. The system is symmetric, positive definite and dominated by the mass matrix part since the remainder is proportional to $k_n \kappa = \frac{h^2}{Pe}$ with typical local Péclet numbers on the order of 100 or more. Consequently, the eigenvalues of the matrix are well clustered, independent of the mesh size and the CG method converges in a number of steps independent of the mesh size (Saad 2003). We use an incomplete LU decomposition as a preconditioner. Typical iteration counts are between 10 and 30.

3.3.2 Stokes system

The Stokes system (10) is more challenging because of the saddle point structure with zero diagonal block. Solving linear equations

with such a structure is discussed in great detail in Elman *et al.* (2005). An extensive overview of methods for the Stokes system in the context of mantle convection is given in Geenen *et al.* (2009), and we basically follow their approach.

Of the available iterative solvers for indefinite symmetric problems such as GMRES, SymmLQ or MinRes (Saad 2003), only GMRES can deal with the non-symmetric preconditioners we will discuss below and that have been shown to be the most efficient for this problem. Standard GMRES determines whether to stop the iteration by estimating the norm of the residual based on preconditioned iterates. Since the preconditioner we consider below uses inexact solves, it is not a linear operation, and consequently the residual estimate is inaccurate and not a reliable stopping criterion. Therefore, we use the flexible GMRES (FGMRES) variant of GMRES that uses one explicit residual computation per iteration to determine whether the stopping criterion has been met.

Any iterative solver for large problems requires preconditioners to lower the condition number of linear systems, preferably to a value that is independent of the mesh size. This could in principle be done by looking at the matrix as a whole [see, e.g. Saad (2003), or the multigrid approach in Kameyama (2005)], but is uncommon for block systems such as the one considered here. Rather, most efficient preconditioners found so far for the Stokes system are based on variants of the ones described in Silvester & Wathen (1994) and are defined by the non-symmetric block triangular matrix

$$Y = \begin{pmatrix} A & B^T \\ 0 & -S \end{pmatrix}, \quad \text{with} \quad Y^{-1} = \begin{pmatrix} A^{-1} & A^{-1}B^TS^{-1} \\ 0 & -S^{-1} \end{pmatrix},$$

where $S = BA^{-1}B^T$ is the Schur complement of the Stokes operator. Applying Y^{-1} as a right preconditioner yields

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} Y^{-1} = \begin{pmatrix} I & 0 \\ BA^{-1} & I \end{pmatrix},$$

for which it can be shown that GMRES converges in at most two iterations (Silvester & Wathen 1994).

This preconditioner is not practically useful because it involves exact inverses A^{-1} and S^{-1} . In our computations, we therefore use the following preconditioner instead

$$Y^{-1} = \begin{pmatrix} \widetilde{A}^{-1} & -\widetilde{A}^{-1}B^T\widetilde{S}^{-1} \\ 0 & \widetilde{S}^{-1} \end{pmatrix}, \quad (17)$$

where \widetilde{A}^{-1} , \widetilde{S}^{-1} approximate the exact inverses. We will discuss our choices for these matrices next. In their construction, it is important to remember that iterative solvers do not need element-by-element representations of matrices like \widetilde{A}^{-1} but only the results of products like $\widetilde{A}^{-1}r$ for a given vector r , using whatever convenient way this can be computed.

Choice of \widetilde{A}^{-1} . Since A is symmetric, we compute the product $x = \widetilde{A}^{-1}r$ by solving the linear system $Ax = r$ for x using a CG solver with a loose tolerance of 10^{-2} (relative residual). This is a sufficient approximation, and the outer GMRES solver will take care that the solution to the whole system will converge to the desired tolerance.

This approximate CG solve needs to be preconditioned. Candidate preconditioners that yield mesh-independent convergence are multigrid methods, see e.g. (Hackbusch 1985; Trottenberg *et al.* 2001). We use the algebraic multigrid (AMG) implementation provided as part of the ML package (Tuminaro & Tong 2000; Gee *et al.* 2006) of the TRILINOS library (Heroux *et al.* 2005, 2012) for this purpose due to its robustness with respect to variable viscosities and scalability even on very large parallel machines.

The performance of the ML-AMG preconditioner depends on the sparsity structure of the matrix. High-order methods and systems where the different vector components of shape functions couple like in the A matrix in (11) tend to deteriorate the quality of the preconditioner, see also (Geenen *et al.* 2009). Therefore, when preconditioning the inexact solution of A , we do not apply the AMG to the A matrix but instead to a matrix \hat{A} with

$$\begin{aligned} \hat{A}_{ij} &= \sum_{d=1}^{\dim} (\varepsilon([\varphi_i]_d \mathbf{e}_d), 2\eta\varepsilon([\varphi_j]_d \mathbf{e}_d))_{\Omega} \\ &= (\nabla\varphi_i^u, 2\eta\nabla\varphi_j^u)_{\Omega}, \end{aligned}$$

where \mathbf{e}_d is the unit vector in coordinate direction d . In other words, the bilinear form that defines \hat{A} does not couple shape functions that correspond to different velocity components, and \hat{A} consequently has only one-third of the number of entries as A in three space dimensions. On the other hand, we note that when building this preconditioner matrix, we have to ensure that it respects the correct set of boundary conditions on the velocity which may introduce coupling between vector components after all if the boundary conditions require tangential flow; forgetting this coupling turns out to have a devastating effect on the quality of the preconditioner.

Choice of \widetilde{S}^{-1} . The inverse of the Schur complement matrix $S = BA^{-1}B^T$ can be accurately approximated by the inverse of a (weighted) mass matrix in pressure space with entries $M_{ij}^p = (\eta^{-1}\varphi_i, \varphi_j)$. This can be explained by the fact that B approximates a gradient operator, B^T a divergence operator, whereas A^{-1} is the inverse of a matrix that is spectrally close to a Laplace matrix, see also (Silvester & Wathen 1994).

Consequently, we choose $\widetilde{S}^{-1} = (M^p)^{-1}$ in the preconditioner, and computing the action $\widetilde{S}^{-1}r$ on a vector r only requires an approximate CG solve with M^p , which we precondition using an ILU of M^p . This solver converges in 1–5 iterations. This is again independent of the mesh size since the condition number of the mass matrix is independent of the refinement level.

Compared to the application of \widetilde{A}^{-1} , this step is cheap. Consequently, choosing the larger but locally conservative pressure space P_{-q} over the smaller space Q_q (see Section 3.2.3) has only a minor effect on overall run times; furthermore, the matrix M^p is block diagonal when using the discontinuous space P_{-q} , making the inversion of this matrix particularly cheap.

Summary of preconditioner. In summary, applying the preconditioner (17) to a vector, that is, computing

$$\begin{pmatrix} x_U \\ x_P \end{pmatrix} = Y^{-1} \begin{pmatrix} r_U \\ r_P \end{pmatrix},$$

requires the following steps

- (i) form $x_P = -\widetilde{S}^{-1}r_P$ by performing an inexact CG solve with ILU preconditioner of the system $M^p x_P = -r_P$;
- (ii) compute $y = r_U - B^T x_P$
- (iii) form $x_U = \widetilde{A}^{-1}y$ by performing an inexact CG solve of the system $Ax_U = y$ with ML-AMG preconditioner based on the matrix \hat{A} .

Overall performance of solver. The components of the linear solvers outlined above are chosen in such a way that they provide a performance that is mostly mesh size independent and can therefore scale from small to very large problems. In particular, in accordance with theoretical considerations, we observe that the number of outer FGMRES iterations is independent of the mesh size, whereas the number of iterations in the inner solves with the velocity block

using the AMG preconditioner increases only weakly (e.g. from 10 to 15 iterations when increasing the number of unknowns from 10^6 to 2.4×10^8). Inversion of the pressure mass matrix also requires a number of iterations that is independent of the mesh size. The total number of operations for solving the linear Stokes system is therefore almost linear in the number of unknowns, and thus of almost optimal complexity. We verify this through weak scaling experiments in Section 4.5.

It is possible that the solver performance can be further improved by noting that there is a trade-off between the accuracy in inverting A and the number of outer FGMRES iterations. For example, one could choose \tilde{A}^{-1} to be only a single, cheap V-cycle with \hat{A} at the expense of more outer iterations (see also Geenen *et al.* 2009). On the other hand, while overall faster for isoviscous problems, we observe that this occasionally leads to a breakdown of the iteration and is therefore not robust. Consequently, we are experimenting with first trying a preconditioner that only employs a single V-cycle and, if FGMRES has not converged in a certain number iterations with this preconditioner, switching to the more accurate preconditioner that actually uses the approximate inverse of A . We will report on results for this scheme elsewhere.

We end this section by noting that the number of FGMRES iterations can be reduced by more than a factor of 5 (from an average of around 40 to an average around 7) by not starting the iteration with a zero vector, but rather with the extrapolation of the solution vector from the previous time steps using a formula like (5), providing a very significant speedup of the overall runtime. This choice of starting guess is also important to avoid round-off problems, see the Appendix.

3.4 Parallelization

The simulation of 3-D mantle convection requires highly resolved computations, with sometimes hundreds of millions or billions of unknowns, to yield reliable results. With today's computer hardware, these requirements cannot be met on single machines, but instead need parallelization, see also (Burstedde *et al.* 2008). Our implementation of the algorithms outlined above provides for parallelization both via MPI between a possibly large number of distributed memory machines as well as via threads on shared memory machines or within individual nodes of a cluster of computers. Both kinds of parallelization are mostly transparent to the application code and are primarily handled in library code in DEAL.II (for the mesh and finite element specific parts) or TRILINOS (for the linear algebra).

To be efficient, parallelization requires that all parts of a program be parallelized to the same degree. In adaptive finite element codes, this implies that the mesh creation, assembly of linear systems, linear solvers and preconditioners, postprocessing steps such as the evaluation of the solution, generation of output files for visualization, or the evaluation of error indicators and the adaptation of the mesh are all parallelized. Our code provides for all of these components. In particular, all mesh operations in DEAL.II build on the P4EST library for parallel mesh management (Burstedde *et al.* 2011) that has been shown to scale to more than 200 000 processors, and the linear algebra components in TRILINOS's Epetra and ML packages have also been demonstrated to scale to machines of this size. We have previously verified the scalability of a large number of DEAL.II components up to at least 16 384 processor cores in Bangerth *et al.* (2011) where we also report on scalability of a 2-D version of the code discussed here. We show additional data below in Section 4.5.

4 RESULTS

To verify the correctness, accuracy and efficiency of our code, we have run a number of benchmarks. We report results on five of these below, namely the SolCx and SolKz benchmarks for the solution of the Stokes equation from Duretz *et al.* (2011) in Section 4.1, the pure shear inclusion benchmark from the same paper in Section 4.2, one of the 2-D convection benchmarks from Blankenbach *et al.* (1989) in Section 4.3 and one of the 3-D convection benchmarks from Busse *et al.* (1993) in Section 4.4, both of which are widely used in other papers as well. We show parallel scalability in Section 4.5 and some results of global mantle convection simulations in Section 4.6.

Additional parallel scalability analyses are provided in Bangerth *et al.* (2011) and we will report on results for the semi-analytic benchmark of Tan & Gurnis (2007) in Geenen *et al.* (2012, in preparation).

4.1 2-D variable viscosity Stokes benchmarks

To verify the correctness of our Stokes solver, we use the SolCx benchmark that was previously used in Duretz *et al.* (2011, section 4.1.1) and whose analytic solution is given in Zhong (1996). We use the implementation of this analytic solution that is available as part of the Underworld package (see Moresi *et al.* 2007, and <http://www.underworldproject.org/>), correcting for the mismatch in sign between the implementation and the description in Duretz *et al.* (2011).

The SolCx benchmark computes the Stokes flow field of a fluid driven by spatial density variations, subject to a spatially variable viscosity. Specifically, the domain is $\Omega = [0, 1]^2$, gravity is $\mathbf{g} = (0, -1)^T$ and the density is given by $\rho(\mathbf{x}) = \sin(\pi x_1) \cos(\pi x_2)$; this can be considered a density perturbation to a constant background density. The viscosity is

$$\eta(\mathbf{x}) = \begin{cases} 1 & \text{for } x_1 \leq 0.5, \\ 10^6 & \text{for } x_1 > 0.5. \end{cases}$$

This strongly discontinuous viscosity field yields an almost stagnant flow in the right half of the domain and consequently a singularity in the pressure along the interface. Boundary conditions are free slip on all of $\partial\Omega$. The temperature plays no role in this benchmark. The prescribed density field and the resulting velocity field are shown in Fig. 2.

To assess the accuracy of the solution, we compute the error between the computed velocity field and the analytic one. If the viscosity is chosen constant throughout the domain, we have confirmed that the velocity solution is accurate on uniform meshes to order $\mathcal{O}(h^3)$, and the pressure solution accurate to order $\mathcal{O}(h^2)$ when measuring the error in the L_2 norm, where h is the cell size of the mesh. This is the expected rate when using $Q_2^d \times Q_1$ elements as discussed in Section 3.2.3; we also obtain the correct higher convergence orders when increasing the polynomial degree of finite element shape functions.

The more interesting case is of course if $\eta(\mathbf{x})$ is chosen discontinuous as discussed earlier. For that case, we show the errors in velocity and pressure measured in the L_2 norm in Table 1. Even though this is not a common error measure in finite element computations, the figure also shows the errors measured in the L_1 norm to allow comparison with the results in Duretz *et al.* (2011). The velocity here converges as $\mathcal{O}(h^3)$ in both the L_1 and L_2 norms, again as expected. For the error in the pressure, we observe $\mathcal{O}(h)$ in the L_1 norm and $\mathcal{O}(h^{1/2})$ in the L_2 norm. This behaviour is consistent with an internal layer of width $\delta \ll h$ along the discontinuity in the

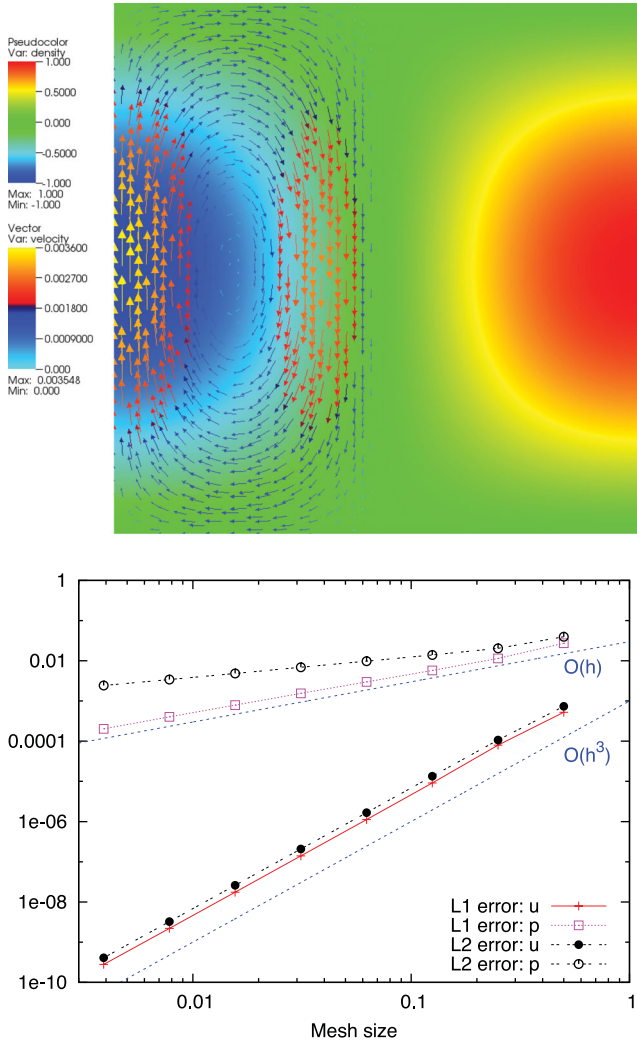


Figure 2. SolCx Stokes benchmark. Top panel: Given (relative) density and velocity field. Bottom panel: Error in the velocity and pressure fields with respect to mesh size.

viscosity on which the pressure error does not decay. We expect to recover the maximal theoretical order once the mesh is fine enough to resolve the internal layer in the exact solution, that is, when $h \approx \delta$. However, this may not be within the computationally feasible range for a jump in η as large as that used here.

The errors shown here are far below the range where we should care about their size, even for modestly fine meshes. However, to verify the efficiency of our implementation, we have also recorded

the errors when using the $Q_3^d \times Q_2$ element in Table 1. The results confirm the higher-order accuracy of the velocity field, and the limiting effect of the discontinuous viscosity on the pressure error convergence order, albeit at a smaller magnitude.

A more interesting comparison, namely with the $Q_2^d \times P_{-1}$ element (see Section 3.2.3) is also shown in Table 1. This element is locally conservative because it has a discontinuous pressure element. In all our experiments we have found that whether we use Q_q or P_{-q} elements has virtually no influence on the error in the velocity. Furthermore, for problems with smooth solutions the price to pay for the local conservation property is that the error in the pressure is about twice as large. However, for the current case with an almost singular pressure along the viscosity jump, the discontinuous shape functions allow to resolve the pressure so well that we in fact recover the full convergence order in the pressure.

While this might suggest that the $Q_{q+1}^d \times P_{-q}$ element is a good choice for problems with discontinuous or strongly varying viscosity fields, the last two columns of the table show that the improved accuracy can only be obtained if the discontinuity in the viscosity is aligned with cell boundaries. If the discontinuity cuts through cells, convergence orders revert to $\mathcal{O}(h)$ for the velocity and $\mathcal{O}(h^{1/2})$ for the pressure, in accordance with observations elsewhere in the literature (see, e.g. Deubelbeiss & Kaus 2008; Durtz *et al.* 2011). For such non-aligned meshes, the errors obtained with the $Q_2^d \times Q_1$ are of same convergence order and about the same magnitude as those shown in the table for the $Q_2^d \times P_{-1}$ element.

In addition to the SolCx benchmark, we have also run the SolKz benchmark that has a similar setup but a smoothly varying viscosity field (see Durtz *et al.* 2011, Section 4.1.2, and references therein). Given the lack of an internal layer, one expects full convergence orders $\mathcal{O}(h^3)$ and $\mathcal{O}(h^2)$ for $\|e_u\|_{L_2}$, $\|e_p\|_{L_2}$ when using the $Q_2^d \times Q_1$ element. These are indeed observed.

4.2 2-D circular inclusion benchmark

To further explore the question of what happens when a jump in the viscosity is not aligned with cell boundaries, we have also implemented the ‘pure shear/inclusion’ benchmark [see Durtz *et al.* (2011), Section 4.1.3 and for the analytic solution see Schmid & Podladchikov (2003)]. In this benchmark, a rather rigid disk with viscosity of 10^3 is embedded in material with viscosity 1, and the medium is compressed in vertical and stretched in horizontal direction by appropriate boundary conditions. This setup ensures that cell boundaries can never be aligned with the circular interface between the regions with different viscosities, and the only hope of improving on the reduced convergence orders is to resolve the interface better using adaptive meshes. As this benchmark again

Table 1. SolCx Stokes benchmark. Velocity and pressure errors e_u , e_p and convergence rates for different choices of the Stokes finite element spaces as discussed in Section 3.2.3, using globally refined meshes. For ‘odd’ meshes, the numbers shown are the average errors from nearby meshes (e.g. for $h = 1/64$, the average of the errors on 63×63 and 65×65 meshes).

h	$Q_2^d \times Q_1$		$Q_3^d \times Q_2$		$Q_2^d \times P_{-1}$, even mesh		$Q_2^d \times P_{-1}$, odd mesh	
	$\ e_u\ _{L_2}$	$\ e_p\ _{L_2}$	$\ e_u\ _{L_2}$	$\ e_p\ _{L_2}$	$\ e_u\ _{L_2}$	$\ e_p\ _{L_2}$	$\ e_u\ _{L_2}$	$\ e_p\ _{L_2}$
1/8	1.3×10^{-5}	1.4×10^{-2}	6.3×10^{-7}	8.8×10^{-3}	1.3×10^{-5}	1.5×10^{-3}	6.5×10^{-4}	1.1×10^{-2}
1/16	1.7×10^{-6}	9.8×10^{-3}	4.0×10^{-8}	6.2×10^{-3}	1.7×10^{-6}	3.7×10^{-4}	3.6×10^{-4}	6.8×10^{-3}
1/32	2.1×10^{-7}	6.9×10^{-3}	2.6×10^{-9}	4.4×10^{-3}	2.2×10^{-7}	9.2×10^{-5}	1.9×10^{-4}	4.5×10^{-3}
1/64	2.6×10^{-8}	4.9×10^{-3}	1.7×10^{-10}	3.1×10^{-3}	2.6×10^{-8}	2.3×10^{-5}	9.8×10^{-5}	3.3×10^{-3}
1/128	3.3×10^{-9}	3.4×10^{-3}	2.0×10^{-11}	2.2×10^{-3}	3.2×10^{-9}	5.7×10^{-6}	5.0×10^{-5}	2.1×10^{-3}
1/256	4.1×10^{-10}	2.4×10^{-3}	1.7×10^{-11}	1.5×10^{-3}	4.1×10^{-10}	1.4×10^{-6}	2.5×10^{-5}	1.5×10^{-3}
	$\mathcal{O}(h^3)$	$\mathcal{O}(h^{1/2})$	$\mathcal{O}(h^4)$	$\mathcal{O}(h^{1/2})$	$\mathcal{O}(h^3)$	$\mathcal{O}(h^2)$	$\mathcal{O}(h)$	$\mathcal{O}(h^{1/2})$

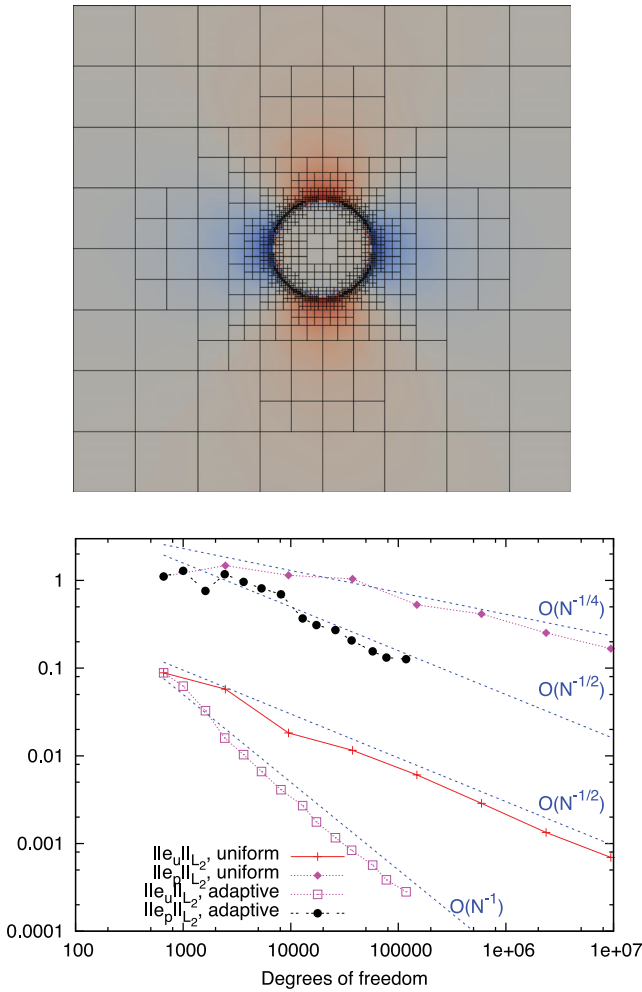


Figure 3. Inclusion benchmark. Top panel: Pressure and adaptive mesh. Bottom panel: Errors in velocity and pressure fields for uniformly and adaptively refined meshes as a function of the number of degrees of freedom.

only requires the solution of the Stokes system, we apply the error estimator to the velocity instead of the temperature (as it is described in Section 3.2.1).

As in the SolCx benchmark, the significant jump in viscosity and consequent jump in pressure leads to pressure oscillations along the interface. This can be seen inside the disk in Fig. 3 along with an adaptive mesh. Convergence results for uniformly and adaptively refined meshes are shown in the bottom panel of this figure and show that for uniformly refined meshes, we obtain convergence orders $\mathcal{O}(h)$ for the velocity and $\mathcal{O}(h^{1/2})$ for the pressure (also seen in Section 4.1 when the discontinuities are not aligned with the mesh). For these meshes, this then translates to orders $\mathcal{O}(N^{-1/2})$ and $\mathcal{O}(N^{-1/4})$ where N is the total number of degrees of freedom. On the other hand, by using adaptive meshes as those shown in the top panel of the figure, we can reach the same error level with a much smaller number of unknowns: for the last computed data points using adaptive meshes, the same error level was obtained with approximately 100 times fewer cells, and corresponding savings in compute time. Furthermore, as indicated in the figure, we recover convergence orders $\mathcal{O}(N^{-1})$ and $\mathcal{O}(N^{-1/2})$ for velocity and pressure errors.

We note that the results shown in Fig. 3 were obtained using the $Q_2^d \times Q_1$ element. As before, given the non-aligned interfaces, the $Q_2^d \times P_{-1}$ element achieves the same convergence order but with

a constant that makes the errors in velocity and pressure about 2–3 times larger compared to the continuous pressure element.

4.3 2-D convection benchmark

While the previous benchmarks only tested the accuracy of the Stokes solver, in this and the following section, we will also include the effects of the advected temperature. Specifically, we compare our implementation to the well-known 2-D dynamic benchmark problem described in Blankenbach *et al.* (1989). The benchmark is solved in non-dimensional units in the form of eqs (1)–(3), using the parameters given in Table 2. The computational domain is the rectangle $[0, l] \times [0, h]$. The strength of buoyancy is described by the Rayleigh number $Ra = \beta g \gamma h^5 / \kappa^2 \rho c_p \eta = 216\,000$. The body is heated homogeneously from within with a non-dimensional heat rate $\gamma = 1$.

On the side boundaries, reflective symmetry conditions are assumed, that is, no-normal-flux for velocity, $\mathbf{u} \cdot \mathbf{n} = 0$ (free slip) and $\mathbf{n} \cdot \nabla T = 0$. On the top and bottom, no-slip conditions $\mathbf{u} = 0$ are applied. On the bottom face, the heat flux is zero, $\mathbf{n} \cdot \nabla T = 0$, and we set $T = 0$ on the top face. The simulation is started with a perturbation from the purely conductive state and is run until we reach the periodic cycle after around non-dimensional time $t = 2$. A snapshot of the solution is shown in Fig. 4 (top panel).

We compare results using two measures: (i) the Nusselt number, defined as the ratio between the mean surface temperature gradient and the mean bottom temperature,

$$Nu = \frac{\int_{\partial\Omega_t} \nabla T \cdot \mathbf{n} \, ds}{\int_{\partial\Omega_b} T \, ds}, \quad (18)$$

where $\partial\Omega_t$ is the top face at $z = h$ and $\partial\Omega_b$ the bottom face at $z = 0$. And (ii) the (non-dimensional) root mean square velocity

$$v_{\text{rms}} = \sqrt{\frac{1}{hl} \int_{\Omega} |\mathbf{u}|^2 \, dx}. \quad (19)$$

The bottom panel of Fig. 4 shows a phase diagram with the Nusselt number over the rms velocity, illustrating the periodic nature of the flow after the initial transient has decayed.

We compare the values we obtain for the two measures above to the benchmark data in Blankenbach *et al.* (1989). The results for different mesh sizes with global (non-adaptive) mesh refinement are given in Table 3, and results with AMR are given in Table 4. These

Table 2. Parameters for the benchmark discussed in Section 4.3 based on Blankenbach *et al.* (1989).

Explanation	Nondimensional value	Dimensional value (SI-units)
h Cell height	1	10^6
l Cell length	1.5	1.5×10^6
ρ Fluid density	1	4×10^3
η Kinematic viscosity	1	1.157×10^{18}
κ Thermal diffusion	1	10^{-6}
g Gravity acceleration	1	10
β Thermal expansion coefficient	1	2.5×10^{-5}
γ Rate of internal heating	1	5×10^{-9}
c_p Heat capacity	1	1.25×10^3
Ra Rayleigh number	2.16×10^5	—

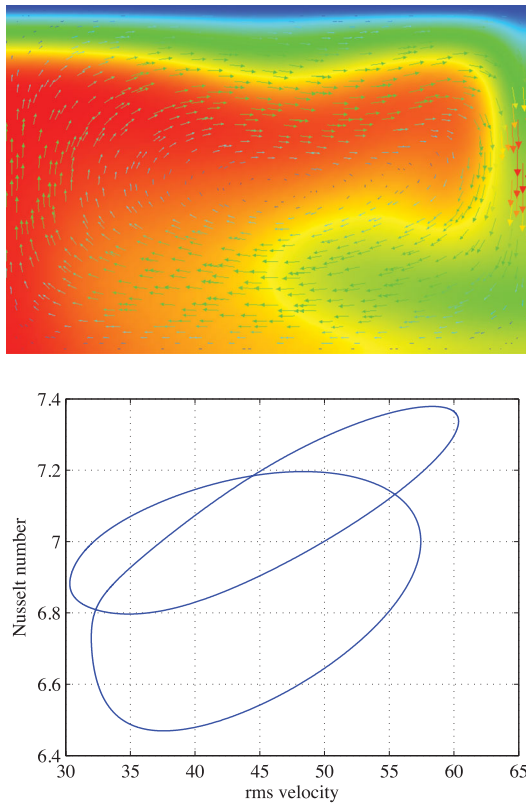


Figure 4. 2-D benchmark. Top panel: Temperature field (with values between 0 and 0.1793) and velocity field (with velocities up to 89.4) for one time step. Bottom panel: Nusselt number over rms velocity. The curve shows the time evolution after the initial transient has decayed and illustrates the periodic nature of the flow.

results show that we correctly reproduce the benchmark results and the substantial savings that can be obtained through adaptive meshes. Not surprisingly, given the advances in numerical methods and computer hardware since (Blankenbach *et al.* 1989), we believe that our results are substantially more accurate than the ones given in the original reference.

4.4 3-D convection benchmark

As a 3-D benchmark, we choose benchmark problem 1a from Busse *et al.* (1993). The problem is posed in a box of dimensions $a \times b \times 1$ with $a = 1.0079$ and $b = 0.6283$, and for Rayleigh number $Ra = 30\,000$. The flow develops to a stationary bimodal flow. A snapshot is shown in Fig. 5.

To find the steady state, we simulate the problem up to non-dimensional time $t = 5$ and record values for the Nusselt number (18) and root mean square velocity (19). Moreover, we also compare the average temperature T_m over the plane $z = 0.75$, point values for the vertical velocity u_3 and temperature at $(0, 0, 0.5)$, and the heat flux $Q(x_1, x_2) = \frac{\partial T}{\partial x_3}|_{x_3=1}$ at the top surface. Results for mesh sizes $24 \times 14 \times 24$, $32 \times 20 \times 32$ and $48 \times 30 \times 48$ (with approximately 220 k, 540 k and 1.8 M unknowns for the velocity/pressure system and 70 k, 170 k and 570 k temperature unknowns) are recorded in Table 5. The results are in good accordance with the reference values, which shows correctness of our implementation also in three spatial dimensions. Note that quantities derived from the FE function values (v_{rms} , T , u_3) are considerably more accurate for coarser meshes than those derived from gradients (Nu, Q). On current hardware and running without parallelization, each time step takes on average approximately 2.5 s, 6 s, 15 s for the three different meshes, respectively.

Table 3. Results for the 2-D benchmark problem with uniform mesh refinement. # DoFs indicates the number of degrees of freedom. Reference results from Blankenbach *et al.* (1989).

Mesh size	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$	Reference
# DoFs	5 276	20 532	80 996	3.2×10^5	1.3×10^6	—
Period	0.048231	0.048051	0.048031	0.048030	0.048029	0.04803 ± 0.00003
Nu^{max}	7.4065	7.3822	7.3789	7.3788	7.3788	7.379 ± 0.005
Nu^{min}	6.5062	6.4717	6.4691	6.4691	6.4692	6.468 ± 0.005
Nu^{max}	7.2637	7.2047	7.1969	7.1960	7.1960	7.196 ± 0.005
Nu^{min}	6.7878	6.7949	6.7961	6.7965	6.7966	6.796 ± 0.005
$v_{\text{rms}}^{\text{max}}$	60.726	60.398	60.361	60.359	60.360	60.367 ± 0.015
$v_{\text{rms}}^{\text{min}}$	31.829	31.965	31.981	31.981	31.982	31.981 ± 0.02
$v_{\text{rms}}^{\text{max}}$	58.225	57.517	57.442	57.437	57.436	57.43 ± 0.05
$v_{\text{rms}}^{\text{min}}$	30.392	30.330	30.324	30.323	30.322	30.32 ± 0.03

Table 4. Results for the 2-D benchmark problem with adaptive mesh refinement. The number of degrees of freedom (# DoFs) for each finest mesh size h varies between time steps; the indicated numbers provide a typical range. Reference results from Blankenbach *et al.* (1989).

Finest mesh size	$\frac{1}{64}$	$\frac{1}{128}$	$\frac{1}{256}$	Reference
# DoFs	$4.5 \dots 6.0 \times 10^4$	$1.6 \dots 2.2 \times 10^5$	$5.6 \dots 8.0 \times 10^5$	—
Period	0.048029	0.048030	0.048030	0.04803 ± 0.00003
Nu^{max}	7.3809	7.3792	7.3788	7.379 ± 0.005
Nu^{min}	6.4718	6.4695	6.4691	6.468 ± 0.005
Nu^{max}	7.1996	7.1967	7.1960	7.196 ± 0.005
Nu^{min}	6.7986	6.7969	6.7965	6.796 ± 0.005
$v_{\text{rms}}^{\text{max}}$	60.366	60.361	60.360	60.367 ± 0.015
$v_{\text{rms}}^{\text{min}}$	31.980	31.981	31.981	31.981 ± 0.02
$v_{\text{rms}}^{\text{max}}$	57.449	57.434	57.435	57.43 ± 0.05
$v_{\text{rms}}^{\text{min}}$	30.322	30.322	30.322	30.32 ± 0.03

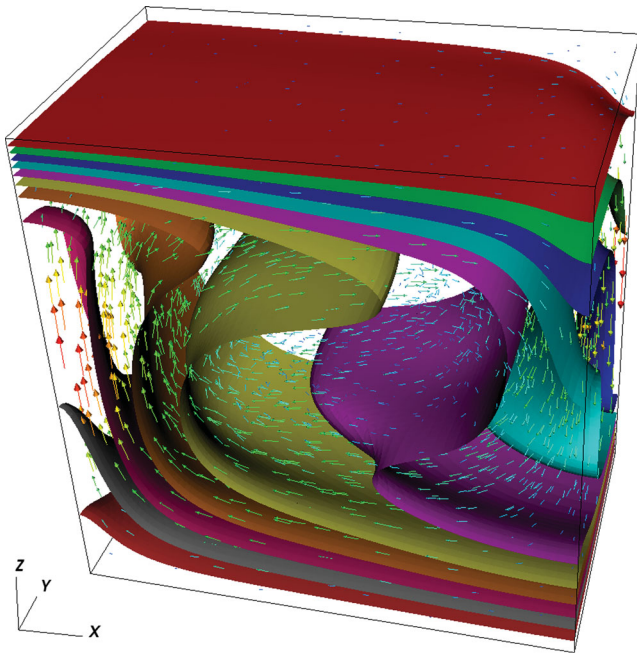


Figure 5. 3-D benchmark. Velocity field and isosurfaces of the temperature.

Table 5. Selected results for the 3-D benchmark problem with uniform mesh refinement. Reference results from Busse *et al.* (1993).

Mesh size	$\frac{1}{24}$	$\frac{1}{32}$	$\frac{1}{48}$	Reference
# DoFs	2.9×10^5	7.2×10^5	2.4×10^6	—
Nu	3.5539	3.5447	3.5397	3.5374 ± 0.0005
v_{rms}	40.997	40.999	40.999	40.999 ± 0.004
$T_m(0.75)$	0.52148	0.52148	0.52148	0.52148 ± 0.00003
$u_3(0, 0, 0.5)$	116.605	116.618	116.623	116.625 ± 0.03
$T(0, 0, 0.5)$	0.80126	0.80128	0.80129	0.80130 ± 0.00005
$Q(0, 0)$	6.7679	6.7357	6.7189	6.7127 ± 0.05
$Q(a, b)$	0.7237	0.7205	0.7174	0.7140 ± 0.05

4.5 Scalability of the solution scheme

Having verified the correctness of the solver, let us now consider its scalability and efficiency. To this end, we start with a spherical shell consisting of 96 coarse mesh cells which we refine either adaptively or globally a number of times until we reach a desired number of unknowns. On this mesh, we then perform one complete time step of our scheme and measure the wall time for the major building blocks of our code for a fixed number of MPI processes each tied to one CPU core (weak scaling). Alternatively, we select a fixed mesh size and measure times for a variable number of MPI processes (strong scaling). Specifically, we measure the run time of the following components

- (i) *Setup DoFs*: This includes giving all degrees of freedom globally unique numbers, computing constraints for hanging nodes, evaluating boundary values, and setting up matrices and vectors.
- (ii) *Assemble Stokes*: Computing and assembling the entries of the Stokes matrix and right-hand side.
- (iii) *Build preconditioner*: Computing and assembling the entries for the Stokes preconditioner matrices as well as initializing the AMG preconditioner for \hat{A} .
- (iv) *Solve Stokes*: Solving the Stokes system.
- (v) *Assemble T RHS*: Computing and assembling the entries of the right-hand side vector for the temperature system.
- (vi) *Solve T*: Solving the temperature equation.

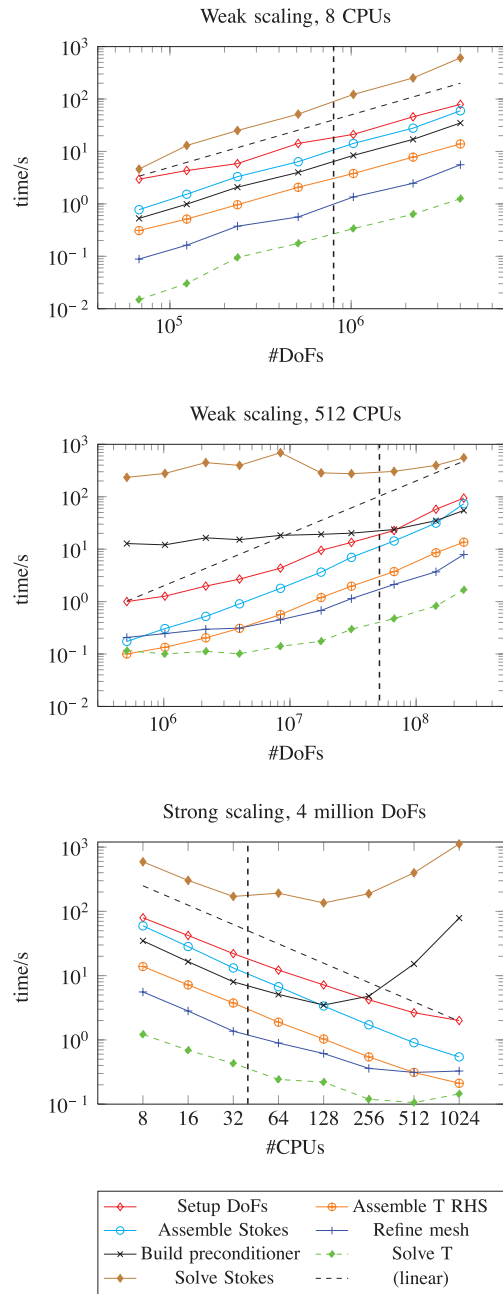


Figure 6. Weak and strong scaling experiments for one time step of a 3-D mantle convection simulation. In each of the graphs, the vertical line indicates 10^5 degrees of freedom per processor core; cores have more than than this threshold to the right of the line in the top two panels, and to the left of the line in the strong scaling results.

- (vii) *Refine mesh*: Computing error indicators for the solution, refining and coarsening the mesh, re-partitioning it between processors, and transferring the solution vectors from the previous to the new mesh.

Fig. 6 shows results for these operations, for both weak and strong scaling experiments. From the figures it is apparent that all operations in our program scale well with increasing problem size (weak scaling) once the problem size per MPI process becomes large enough. Similarly, run times can be reduced inversely proportional to the number of processors (strong scaling) as long as the local size or the problem is sufficiently large. The threshold for this scalability

is approximately a minimal local problem size of 100 000 degrees of freedom per MPI process, indicated by the vertical lines in Fig. 6. This is also consistent with our observations in Bangerth *et al.* (2011). Note that this threshold depends on the computer hardware in use, in particular on the latency of the network which is critical for fast global synchronization in the multigrid preconditioner and in inner products.

We note that in all cases, the time to build the preconditioner and solve the Stokes system dominates all other operations by about an order of magnitude. This is partly due to the fact that, for lack of an alternative, we here start the solver with a zero vector. In contrast, when doing time dependent simulations, we start with the previous solution vector (see the discussion at the end of Section 3.3.2), thereby reducing the fraction of wall time devoted to the Stokes solution from more than 90 per cent to around 70 per cent of the overall run time. With this reduction, we can solve problems at a rate of approximately one time step per minute for large 3-D simulations on current cluster hardware when using 100 000 DoFs per processor core, more or less independently of the overall problem size.

4.6 Modelling the Earth's mantle

To illustrate the ability of ASPECT to solve problems that are relevant to modelling the Earth's mantle, Figs 7 and 8 show snapshots in time of 2-D and 3-D simulations. These simulations use a no-slip velocity boundary condition at the inner rim, a slip boundary condition at the outer rim and keep the temperature constant at either boundary. Neither computation includes adiabatic heating, but compared to the simple model (1)–(3), the 2-D case does include a temperature and pressure (but not strain-rate) dependent viscosity and includes compressibility in the Stokes equation. In both computations, mesh refinement was driven by the second derivative of the temperature which in 3-D primarily resolves the inner boundary layer rather than the plumes (however, see also the solution in Section 3.2.1).

These simulations show the excellent spatial resolution adaptive meshes can provide. We will provide results for computations of more direct geodynamic interest in Geenen *et al.* (2012, in preparation) and elsewhere.

5 CONCLUSIONS AND OUTLOOK

The simulation of convection in the Earth's mantle is complicated by a host of problems related to the mathematical structure of the equations as well as of the disparity of the lengthscales implied by the sizes of physical coefficients in the Earth. Consequently, geodynamics has a long history of the development of methods that can make at least some problems tractable. Nevertheless, fully resolved, 3-D simulations have largely remained beyond the ability of current codes and computers.

On the other hand, modern numerical methods can close a significant part of this gap and make many previously intractable problems possible. In this paper, we have presented a collection of state-of-the-art algorithms for mantle convection and their implementation in the ASPECT code. Specifically, we have shown how the interconnected choice of adaptive meshes, discretization, stabilization, solvers and preconditioners leads to a method that not only provides excellent accuracy at very modest numerical cost, but also allows scaling to very large problems with hundreds of millions of unknowns on hundreds or thousands of processor cores with almost perfect complexity. The implementation of these methods is

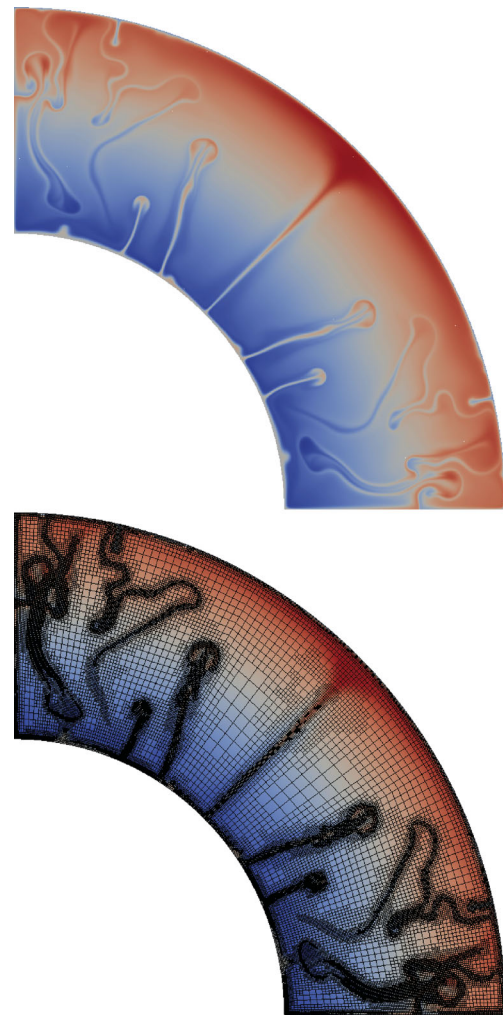


Figure 7. Solution of a 2-D convection problem on a quarter shell domain. The solution has around 700 000 degrees of freedom and was obtained on 16 processors; each time step took less than one second on average. Top panel: Temperature field. Bottom panel: Adaptive mesh of the same solution.

available under an Open Source license in the form of the ASPECT code.

Despite all this, the methods described here are not sufficient to solve entirely realistic models. Specifically, there are at least three obvious places where the simple Boussinesq model described in eqs (1)–(3) is not an adequate description of the real processes that act in the Earth's interior. First, the various coefficients, such as η , ρ , κ or γ , are in reality all non-linear functions of the solution variables \mathbf{u} , p , T . This dependence can either be direct, such as the dependence of the viscosity on the strain rate, or more indirect by considering which rock phases are thermodynamically stable for the current pressure and temperature value, and then using coefficient values appropriate for this phase. A simple approach to deal with this nonlinearity is to evaluate coefficients at the solution values of the previous time step (or at a value extrapolated from the previous time steps), rendering the system linear again. However, this may lead to an inaccurate account of the transition zones that provide the most direct signal that can be compared with data from seismic inversion. Consequently, an iteration is necessary that resolves the nonlinearity. A common solution is to use a Picard-type iteration (see, e.g. Burstedde *et al.* 2008). A more efficient algorithm may be

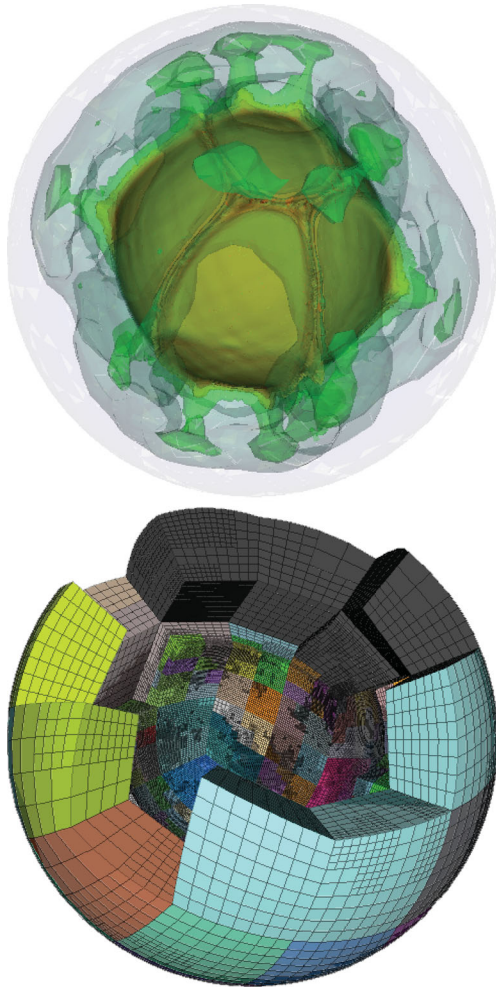


Figure 8. Solution of a 3-D convection problem in a spherical shell. Top panel: Isocontours of the temperature field. Bottom panel: Partitioning of the domain onto 512 processors. The subdomains and corresponding meshes of the first half of all processors are shown. The mesh has 1 424 176 cells, and the solution has approximately 54 million unknowns (39 million velocities, 1.7 million pressures, and 13 million temperatures).

Newton’s method, but it has to be integrated with the linear solvers and preconditioners to be efficient, and it has to be globalized to guarantee convergence even from poor starting guesses (Nocedal & Wright 1999; Worthen 2012). Furthermore, a realistic description of the coefficients often leads to highly heterogeneous coefficients that make the construction of efficient solvers and preconditioners a challenge (Gerya 2010; Ismail-Zadeh & Tackley 2010).

A second challenge is to deal with compressibility effects. While velocities in the Earth’s mantle are orders of magnitude too slow to compress material based on inertial effects, the large hydrostatic pressure significantly increases the density with depth; temperature and the thermodynamically stable rock phase also affect the density. Consequently, a realistic description needs to modify the continuity equation (2) to read $\nabla \cdot (\rho \mathbf{u}) = 0$ instead, where $\rho = \rho(p, T)$. A simple linearization of this equation in the original set of variables \mathbf{u}, p unfortunately leads to a non-symmetric variant of the Stokes system for which the choice of preconditioner is entirely unclear; furthermore, it leads to difficult to solve problems with the compatibility condition this equation implies for the right hand side of the divergence equation. We note that some of the associated questions

have already been addressed in Leng & Zhong (2008), King *et al.* (2010).

A final topic is that Earth’s mantle is not a homogeneous mixture of materials. Rather, material entrained from plates or the core–mantle boundary may have a significantly different chemical composition. It has also been suggested that different layers have different composition (Schubert *et al.* 2001). Simulating heterogeneity entails additional advected fields that describe mass fractions of materials. They can be treated in the same way as the temperature field, with a non-linear viscosity stabilization of sharp interfaces. Alternatively, a number of approaches such as the particle in cell (PIC) method, marker chains or phase fields have been proposed to avoid smearing of interfaces (for a small sample of methods, see (van Keken & Zhong 1999; Tackley & King 2003; Lin & van Keken 2006; Leng & Zhong 2011); see also (Gerya 2010; Ismail-Zadeh & Tackley 2010) for general overviews).

We are working on extending ASPECT in each of the directions outlined above for future releases and will report on our algorithms in a future paper.

ACKNOWLEDGMENTS

WB would like to thank Scott King for insightful discussions on mantle convection and its simulation. The authors are also grateful for discussions with Magali Billen, Thomas Geenen, Eric Heien, Louise Kellogg, Shijie Zhong and others in the community.

The first author was supported by the Graduate School in Mathematics and Computing (FMB) at Uppsala University, Sweden. The second and third authors are supported in part through the Computational Infrastructure in Geodynamics initiative (CIG), through the National Science Foundation under Award No. EAR-0949446 and The University of California—Davis. This publication is based in part on work supported by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST). The third author is also supported in part by an Alfred P. Sloan Research Fellowship.

Some computations for this paper were performed on the ‘Ranger’ and ‘Lonestar’ clusters at the Texas Advanced Computing Center (TACC), and the ‘Brazos’ and ‘Hurr’ clusters at the Institute for Applied Mathematics and Computational Science (IAMCS) at Texas A&M University. Ranger was funded by NSF award OCI-0622780, and we used an allocation obtained under NSF award TG-MCA04N026. The authors acknowledge the Texas A&M Supercomputing Facility for providing computing resources on ‘Lonestar’ useful in conducting the research reported in this paper. Part of Brazos was supported by NSF award DMS-0922866. Hurr is supported by Award No. KUS-C1-016-04 made by King Abdullah University of Science and Technology (KAUST). Some computations were performed on resources provided by SNIC through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under project p2010002.

REFERENCES

- Ainsworth, M. & Oden, J.T., 2000. *A Posteriori Error Estimation in Finite Element Analysis*, John Wiley and Sons, New York, NY.
- Albers, M., 2000. A local mesh refinement multigrid method for 3-D convection problems with strongly variable viscosity, *J. Comp. Phys.*, **160**, 126–150.
- Babuška, I. & Rheinboldt, W.C., 1978. Error estimates for adaptive finite element computations, *SIAM J. Numer. Anal.*, **15**, 736–754.
- Babuška, I. & Strouboulis, T., 2001. *The Finite Element Method and its Reliability*, Clarendon Press, New York, NY.

- Bangerth, W. & Rannacher, R., 2003. *Adaptive Finite Element Methods for Differential Equations*, Birkhäuser Verlag, Basel.
- Bangerth, W. & Kayser-Herold, O., 2009. Data structures and requirements for *hp* finite element software, *ACM Trans. Math. Softw.*, **36**(1), 4/1–4/31.
- Bangerth, W., Hartmann, R. & Kanschat, G., 2007. deal.II—a general purpose object oriented finite element library, *ACM Trans. Math. Softw.*, **33**(4), 24:1–24:27, doi:10.1145/1268776.1268779.
- Bangerth, W., Burstedde, C., Heister, T. & Kronbichler, M., 2011. Algorithms and data structures for massively parallel generic adaptive finite element codes, *ACM Trans. Math. Softw.*, **38**(2), 14:1–14:28, doi:10.1145/2049673.2049678.
- Bangerth, W., Heister, T. & Kanschat, G., 2012. *deal.II Differential Equations Analysis Library, Technical Reference*, available at: <http://www.dealii.org/> (last accessed 2012 August 2).
- Bazilevs, Y., Calo, V.M., Tezduyar, T.E. & Hughes, T.J.R., 2007. YZ β discontinuity capturing for advection-dominated processes with application to arterial drug delivery, *Int. J. Numer. Meth. Fluids*, **54**, 593–608.
- Blankenbach, B. *et al.*, 1989. A benchmark comparison for mantle convection codes, *Geophys. J. Int.*, **98**, 23–38.
- Braess, D., 1997. *Finite Elements*, Cambridge University Press, Cambridge.
- Brenner, S.C. & Scott, R.L., 2002. *The Mathematical Theory of Finite Elements*, 2nd edn, Springer, Berlin.
- Brooks, A. & Hughes, T., 1982. Streamline upwind/Petrov-Galerkin formulation for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, *Comp. Meth. Appl. Mech. Engrg.*, **32**, 199–259.
- Burstedde, C., Ghattas, O., Gurnis, M., Tan, E., Tu, T., Stadler, G., Wilcox, L.C. & Zhong, S., 2008. Scalable adaptive mantle convection simulation on petascale supercomputers, in *SC '08: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, Austin, TX, ACM/IEEE.
- Burstedde, C., Burtcher, M., Ghattas, O., Stadler, G., Tu, T. & Wilcox, L.C., 2009. ALPS: A framework for parallel adaptive PDE solution, *J. Physics: Conf. Series*, **180**, 012009, doi:10.1088/1742-6596/180/1/012009.
- Burstedde, C., Wilcox, L.C. & Ghattas, O., 2011. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, *SIAM J. Sci. Comput.*, **33**(3), 1103–1133.
- Busse, F. *et al.*, 1993. 3D convection at infinite Prandtl numbers in cartesian geometry—a benchmark comparison, *Geophys. astr. Fluid Dyn.*, **75**, 39–59.
- Carey, G.F., 1997. *Computational Grids: Generation, Adaptation and Solution Strategies*, Taylor & Francis, Washington, D.C.
- Chen, Z., 2006. *Computational Methods for Multi-phase Flows in Porous Media*, SIAM, Philadelphia, PA.
- Davies, D.R., Davies, J.H., Hassan, O., Morgan, K. & Nithiarasu, P., 2007a. Investigations into the applicability of adaptive finite element methods to two-dimensional infinite Prandtl number thermal and thermochemical convection, *Geochem. Geophys. Geosyst.*, **8**, Q05010, doi:10.1029/2006GC001470.
- Davies, D.R., Davies, J.H., Hassan, O., Morgan, K. & Nithiarasu, P., 2007b. Adaptive finite element methods in geodynamics: convection dominated mid-ocean ridge and subduction zone simulations, *Int. J. Numer. Meth. Heat Fluid Flow*, **18**, 1015–1035.
- Deubelbeiss, Y. & Kaus, B.J., 2008. Comparison of Eulerian and Lagrangian numerical techniques for the Stokes equations in the presence of strongly varying viscosity, *Phys. Earth planet. Inter.*, **171**, 92–111.
- Donea, J. & Huerta, A., 2003. *Finite Element Methods for Flow Problems*, J. Wiley & Sons, Chichester.
- Duretz, T., May, D.A., Garya, T.V. & Tackley, P.J., 2011. Discretization errors and free surface stabilization in the finite difference and marker-in-cell method for applied geodynamics: A numerical study, *Geochem. Geophys. Geosyst.*, **12**, Q07004, doi:10.1029/2011GC003567.
- Elman, H., Silvester, D. & Wathen, A., 2005. *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics*, Oxford Science Publications, Oxford.
- Gago, J. P. d. S.R., Kelly, D.W., Zienkiewicz, O.C. & Babuška, I., 1983. A posteriori error analysis and adaptive processes in the finite element method: Part II—Adaptive mesh refinement, *Int. J. Num. Meth. Eng.*, **19**, 1621–1656.
- Gee, M.W., Siefert, C.M., Hu, J.J., Tuminaro, R.S. & Sala, M.G., 2006. *ML 5.0 Smoothed Aggregation User's Guide*, Tech. Rep. 2006-2649, Sandia National Laboratories.
- Geenen, T., ur Rehman, M., MacLachlan, S.P., Segal, G., Vuik, C., van den Berg, A.P. & Spakman, W., 2009. Scalable robust solvers for unstructured FE geodynamic modeling applications: Solving the Stokes equation for models with large localized viscosity contrasts, *Geochem. Geophys. Geosyst.*, **10**(9), Q09002, doi:10.1029/2009GC002526.
- Gerya, T., 2010. *Introduction to Numerical Geodynamic Modelling*, Cambridge University Press, Cambridge.
- Girault, V. & Raviart, P.-A., 1986. *Finite Element Methods for the Navier–Stokes Equations*, Springer-Verlag, New York, NY.
- Guermond, J.-L., Pasquetti, R. & Popov, B., 2011. Entropy viscosity method for nonlinear conservation laws, *J. Comput. Phys.*, **230**, 4248–4267.
- Hackbusch, W., 1985. *Multi-Grid Methods and Applications*, Springer, Berlin.
- Hairer, E. & Wanner, G., 1991. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin.
- Heroux, M.A. *et al.*, 2005. An overview of the Trilinos project, *ACM Trans. Math. Softw.*, **31**, 397–423.
- Heroux, M.A. *et al.*, 2012. Trilinos web page, <http://trilinos.sandia.gov>.
- Ismail-Zadeh, A. & Tackley, P., 2010. *Computational Methods for Geodynamics*, Cambridge University Press, Cambridge.
- Kameyama, M., 2005. ACuTEM: a multigrid-based mantle convection simulation code and its optimization to the Earth simulator, *J. Earth Simulator*, **4**, 2–10.
- van Keken, P. & Zhong, S., 1999. Mixing in a 3D spherical model of present-day mantle convection, *Earth planet. Sci. Lett.*, **171**, 533–547.
- King, S.D., Raefsky, A. & Hager, B.H., 1990. Conman: vectorizing a finite element code for incompressible two-dimensional convection in the earth's mantle, *Phys. Earth planet. Inter.*, **59**, 195–207.
- King, S.D., Lee, C., van Keken, P.E., Leng, W., Zhong, S., Tan, E., Tosi, N. & Kameyama, M.C., 2010. A community benchmark for 2-D Cartesian compressible convection in the Earth's mantle, *Geophys. J. Int.*, **180**, 73–87.
- Kronbichler, M. & Bangerth, W., 2011. deal.II Tutorial Program Step-31, http://www.dealii.org/developer/doxygen/deal.II/step_31.html (last accessed 2012 August 2).
- Kronbichler, M., Heister, T. & Bangerth, W., 2011. deal.II Tutorial Program Step-32, http://www.dealii.org/developer/doxygen/deal.II/step_32.html (last accessed 2012 August 2).
- Leng, W. & Zhong, S., 2008. Viscous heating, adiabatic heating and energetic consistency in compressible mantle convection, *Geophys. J. Int.*, **173**, 693–702.
- Leng, W. & Zhong, S., 2011. Implementation and application of adaptive mesh refinement for thermochemical mantle convection studies, *Geochem. Geophys. Geosyst.*, **12**, Q04006, doi:10.1029/2010GC003425.
- LeVeque, R.J., 2002. *Finite Volume Methods for Hyperbolic Problems*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge.
- Lin, S.-C. & van Keken, P.E., 2006. Deformation, stirring and material transport in thermochemical plumes, *Geophys. Res. Lett.*, **33**, L20306, doi:10.1029/2006GL027037.
- Moresi, L., Zhong, S.J. & Gurnis, M., 1996. The accuracy of finite element solutions of Stokes' flow with strongly varying viscosity, *Phys. Earth planet. Inter.*, **97**, 83–94.
- Moresi, L., Quenette, S., Lemiale, V., Meriaux, C., Appelbe, B. & Mühlhaus, H.B., 2007. Computational approaches to studying non-linear dynamics of the crust and mantle, *Phys. Earth planet. Inter.*, **163**, 69–82.
- Nocedal, J. & Wright, S.J., 1999. *Numerical Optimization*, Springer Series in Operations Research, Springer, New York, NY.
- Quarteroni, A. & Valli, A., 1994. *Numerical Approximation of Partial Differential Equations*, Springer, Heidelberg.
- Saad, Y., 2003. *Iterative Methods for Sparse Linear Systems*, 2nd edn, SIAM, Philadelphia, PA.

- Schmid, D. & Podladchikov, Y., 2003. Analytical solutions for deformable elliptical inclusions in general shear, *Geophys. J. Int.*, **155**(1), 269–288.
- Schubert, G., Turcotte, D.L. & Olson, P., 2001. *Mantle Convection in the Earth and Planets, Part 1*, Cambridge University Press, Cambridge.
- Sheldon, J.W., Zondek, B. & Cardwell, W.T., 1959. One-dimensional, incompressible, non-capillary, two-phase fluid flow in a porous medium, *Trans. SPE AIME*, **216**, 290–296.
- Silvester, D. & Wathen, A., 1994. Fast iterative solution of stabilised Stokes systems. Part II: using general block preconditioners, *SIAM J. Numer. Anal.*, **31**, 1352–1367.
- Stadler, G., Gurnis, M., Burstedde, C., Wilcox, L.C., Alisic, L. & Ghattas, O., 2010. The dynamics of plate tectonics and mantle flow: From local to global scales, *Science*, **329**, 1033–1038.
- Stone, H.L. & Garder, A.O., 1961. Analysis of gas-cap or dissolved-gas reservoirs, *Trans. SPE AIME*, **222**, 92–104.
- Tackley, P.J. & King, S.D., 2003. Testing the tracer ratio method for modeling active compositional fields in mantle convection simulations, *Geochem. Geophys. Geosyst.*, **4**, 8302, doi:10.1029/2001GC000214.
- Tan, E. & Gurnis, M., 2007. Compressible thermochemical convection and application to lower mantle structures, *J. geophys. Res.*, **112**, doi:10.1029/2006JB004505.
- Trottenberg, U., Oosterlee, C. & Schüller, A., 2001. *Multigrid*, Elsevier Academic Press, London.
- Tuminaro, R. & Tong, C., 2000. Parallel smoothed aggregation multigrid: aggregation strategies on massively parallel machines, in *Super Computing 2000 Proceedings*, Dallas, TX.
- Worthen, J., 2012. Inverse problems in mantle convection: models, algorithms, and applications (in preparation), *PhD thesis*, University of Texas at Austin, TX.
- Zhong, S., 1996. Analytic solution for Stokes' flow with lateral variations in viscosity, *Geophys. J. Int.*, **124**, 18–28.
- Zhong, S., McNamara, A., Tan, E., Moresi, L. & Gurnis, M., 2008. A benchmark study on mantle convection in a 3-D spherical shell using CitcomS, *Geochem. Geophys. Geosyst.*, **9**, Q10017, doi:10.1029/2008GC002048.

APPENDIX: ACCURATE PRESSURE SOLVES AND LARGE-SMALL SPLITS

ASPECT solves the Stokes system in the Boussinesq equations in the form (1)–(2). In this formulation, the pressure one computes is the total pressure of which the overwhelming component (by several orders of magnitude at the bottom of the Earth's mantle) is the static pressure from the overlying rock while the dynamic pressure is much smaller. Thus, if one were to naively solve the discrete system (10) to, say, two or three digits of accuracy, one only gets this accuracy in the static part of the pressure. Subtracting the static from the total pressure would then leave us with no accuracy whatsoever in the dynamic part. This is an important consideration since for many applications only the dynamic part is important; furthermore, it is only the dynamic part that drives the flow. Consequently, geodynamics codes have traditionally not solved this form of the equations.

In the following section, we will explain that the solution method used in ASPECT is equivalent to a scheme where one solves for the change in the dynamic pressure instead of the full pressure. This is very similar to a traditional splitting of the pressure into a static and dynamic component and allows us to compute the dynamic pressure with high accuracy (as shown numerically at the end of this section).

A1 Large-small splits to avoid round-off problems

One can circumvent the problem by splitting the pressure $P = \hat{P} + P'$ into a large and known component \hat{P} and a small but

unknown component P' and then solving

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} U \\ P' \end{pmatrix} = \begin{pmatrix} F - B^T \hat{P} \\ 0 \end{pmatrix} \quad (\text{A1})$$

instead of (10). Here, if \hat{P} is known to high accuracy and if we solve the linear system for P' with two or three digits of accuracy, then $P = \hat{P} + P'$ will be known to high accuracy as well and we can obtain the dynamic pressure from it with high accuracy.

Traditionally, this split $P = \hat{P} + P'$ has been done by setting $\hat{P} = P_s$, that is, the static pressure, where P_s satisfies the equation $\nabla P_s = \rho(P_s, T_s)\mathbf{g}$ in a radially symmetric coordinate system and with an adiabatic temperature T_s matching P_s (or simply a prescribed temperature profile). Using this choice for \hat{P} , we then have that $P' = P_d$, that is, the dynamic pressure, and the problem is equivalent to solving the modified Stokes equations

$$-\nabla \cdot (2\eta\epsilon(\mathbf{u})) + \nabla p_d = (\rho(p, T) - \rho(p_s, T_s))\mathbf{g}, \quad (\text{A2})$$

$$\nabla \cdot \mathbf{u} = 0 \quad (\text{A3})$$

instead of (1)–(2). This form makes clear that the flow is in fact driven by density variations (buoyancy) and not the density itself.

That said, it is not necessary to choose the particular split of variables $P = P_s + P_d$ —any other split into a large and known component \hat{P} and a small but unknown component P' would work as well. In particular, we could choose $\hat{P} = P^{n-1}$, that is, the (discrete) pressure from the previous time step and $P' = \Delta P^n = P^n - P^{n-1}$, the difference between the pressure of the current and previous time steps. We will see below, that this is equivalent to the method used in ASPECT. Similar to the split into static and dynamic pressure, P' is typically many orders of magnitude smaller than P : first, because it is a quantity that must go to zero as the time step size goes to zero; second, because $P' = P^n - P^{n-1} = (P^n - P_s) - (P^{n-1} - P_s) = P_d^n - P_d^{n-1}$, that is, it is in reality only the change in dynamic pressures between time steps, both of which are already small. In other words, using this splitting, and solving

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} U \\ \Delta P^n \end{pmatrix} = \begin{pmatrix} F - B^T P^{n-1} \\ 0 \end{pmatrix} \quad (\text{A4})$$

we can again get a high accuracy solution $P = P^{n-1} + \Delta P^n$ if we assume that we have computed the previous time step's solution with high accuracy and solve for ΔP with moderate accuracy. Doing this recursively means that we have to solve for the first time step's pressure accurately—but there, ASPECT uses the static pressure which is known to high accuracy. In other words, the approach taken in ASPECT in the first time step boils down to the conventional technique of solving for the dynamic pressure in the iterative solver, whereas it uses a different scheme at later time steps.

We prefer using P^{n-1} for the splitting instead of the static pressure P_s for several reasons. First, P^{n-1} is readily available, whereas computing P_s is an additional computational effort—and a non-trivial effort in particular if one were to use a horizontally averaged temperature rather than a prescribed temperature profile. Secondly, P^{n-1} is more accurate because it is defined on the correct interpolation points, so no averaging or interpolation needs to be done. Thirdly, it is not obvious how to generalize the static pressure when surface geometry is included. Note that we do compute a horizontally averaged pressure when needed in ASPECT, but it is currently only used for graphical output.

The conclusion so far is that to compute the total pressure to high accuracy (yielding an accurate dynamic component of the total

pressure) only requires a large-small split and that one can do that using the previous time step's pressure, among other choices.

A2 Equivalence of the split and unsplit schemes

Let us assume that we indeed use the scheme where we split the pressure as discussed above into $P = P^{n-1} + \Delta P^n$ and then proceed to solve (A4) using the GMRES Krylov space solver discussed in the Section 3.3.2.⁴ For simplicity, let us abbreviate the linear system as

$$HX_{\text{split}} = Z_{\text{split}}, \quad (\text{A5})$$

where

$$H = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}, X_{\text{split}} = \begin{pmatrix} U \\ \Delta P^n \end{pmatrix}, Z_{\text{split}} = \begin{pmatrix} F - B^T P^{n-1} \\ 0 \end{pmatrix}.$$

Since there is no good starting value for the pressure update ΔP^n in the split scheme, we start the GMRES iteration with the vector $X_{\text{split}}^{(0)} = [U^{n-1}, 0]^T$. GMRES then proceeds to find its solution in the k th iteration by constructing the minimal residual vector from the Krylov subspace

$$V_{\text{split}}^{(k)} = \{R_{\text{split}}, HR_{\text{split}}, H^2 R_{\text{split}}, \dots, H^{k-1} R_{\text{split}}\},$$

see, for example, (Saad 2003). Here, $R_{\text{split}} = Z_{\text{split}} - HX_{\text{split}}^{(0)} = [F - B^T P^{n-1} - AU^{n-1}, -B^T U^{n-1}]^T$ is the initial residual of the iteration.

On the other hand, consider what would happen if we were to solve the problem in the original, unsplit formulation (10), which we can write as

$$HX_{\text{unsplit}} = Z_{\text{unsplit}}, \quad (\text{A6})$$

where

$$H = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}, X_{\text{unsplit}} = \begin{pmatrix} U \\ P \end{pmatrix}, Z_{\text{unsplit}} = \begin{pmatrix} F \\ 0 \end{pmatrix}.$$

In contrast to above, we have a good initial guess for P , namely P^{n-1} for $n \geq 1$ and the static pressure for the initial time step. Thus, we start the GMRES iteration using the starting vector $X_{\text{unsplit}}^{(0)} = [U^{n-1}, P^{n-1}]^T$. Consequently, in its k th iteration, GMRES computes the current iterate from the space

$$V_{\text{unsplit}}^{(k)} = \{R_{\text{unsplit}}, HR_{\text{unsplit}}, H^2 R_{\text{unsplit}}, \dots, H^{k-1} R_{\text{unsplit}}\}.$$

Again, $R_{\text{unsplit}} = Z_{\text{unsplit}} - HX_{\text{unsplit}}^{(0)} = [F - AU^{n-1} - B^T P^{n-1}, -B^T U^{n-1}]^T$ is the initial residual of the iteration.

Now, notice that in fact $R_{\text{unsplit}} = R_{\text{split}}$. Consequently, we have that $V_{\text{unsplit}}^{(k)} = V_{\text{split}}^{(k)}$. What this means is that GMRES computes, in every iteration, the exact same approximation $X^{(k)}$ whether we

⁴In the following, we will ignore the fact that we are using a preconditioner in our code—all results below are equally valid with or without a preconditioner.

start the iteration with a zero pressure update and bring the previous time step's pressure to the right hand side, or whether we solve for the total pressure and simply start with the previous pressure! This realization validates the approach we take in ASPECT, namely to solve the unsplit scheme (10) without bringing the large part of the pressure from the previous time step to the right hand side as in (A4), but with a good starting guess for GMRES. Hence, ASPECT should produce an accurate total pressure from which the dynamic pressure can be obtained even if it is orders of magnitude smaller.

A3 Interpretation and numerical validation

The realization above that one can compute the dynamic component of the pressure accurately while only solving for the total pressure is so surprising—and so much against common experience in geodynamics—that it is worth to reflect on it. Indeed, it is true that one can only show the equivalence of the two schemes because the method we use—GMRES—is a Krylov subspace method. It is quite clear that one would not be able to obtain an accurate dynamic pressure component from the unsplit scheme using other kinds of linear solvers for the Stokes system, such as preconditioned fixed point iterations or the Uzawa method.

Secondly, while we have shown that the computations GMRES makes are exactly the same in both cases, one may ask whether there are other potential sources of round-off errors that could lead to problems. To this end, we have considered a variation of the SolCx benchmark discussed in Section 4.1. There, we use a density $\rho(\mathbf{x}) = \sin(\pi x_1) \cos(\pi x_2)$ that has mean value zero and, consequently, yields a pressure that varies between -0.255 and $+0.255$ and has no static component. For this case, on a mesh with cell size $h = 1/16$, we then obtain an error between the computed solution u_h, p_h and the exact one u, p of $\|u - u_h\|_{L_2} = 1.67 \times 10^{-6}$, $\|p - p_h\|_{L_2} = 9.78 \times 10^{-3}$, see also Fig. 2. These values were computed using the minimal number of GMRES iterations so that the iteration error does not dominate the discretization error; in other words, we do not use an excessive number of linear iterations.

On the other hand, let us consider a case where the density is given by $\rho(\mathbf{x}) = 10^6 + \sin(\pi x_1) \cos(\pi x_2)$. Here, the total pressure is the sum of a static pressure $p_s = 10^6(\frac{1}{2} - z)$ that we have normalized in such a way that it has mean zero, and the same dynamic pressure as above. Indeed, given the large mean density, the static pressure is about 10^6 times larger than the dynamic one, and conventionally one would expect that the error in the numerical solution may be small relative to the total pressure, but not small relative to the dynamic component. Yet, computing again the difference between the solution ASPECT has found when using the same number of GMRES iterations as in the case above, and the exact solution, yields errors $\|u - u_h\|_{L_2} = 1.67 \times 10^{-6}$, $\|p - p_h\|_{L_2} = 9.78 \times 10^{-3}$. These are, indeed, exactly the same as in the previous case, thus validating numerically that the approach taken in ASPECT is sound. Note in particular that since the error was small relative to the dynamic pressure in the first case, and since it is exactly the same here, that the error in the current case is also small relative to the dynamic pressure, and not only relative to the total pressure.