Large-scale adaptive mantle convection simulation

Carsten Burstedde,1,2 Georg Stadler,1 Laura Alisic,3,4 Lucas C. Wilcox,1,5 Eh Tan,6,7 Michael Gurnis3 and Omar Ghattas1,8,9

1Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, Austin, TX, USA. E-mail: burstedde@ices.utexas.edu
2Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms-Universität Bonn, Germany. E-mail: burstedde@ins.uni-bonn.de
3Seismological Laboratory, California Institute of Technology, Pasadena, CA, USA
4Bullard Laboratories, University of Cambridge, Cambridge, UK
5Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA, USA
6Computational Infrastructure for Geodynamics (CIG), Pasadena, CA, USA
7Institute of Earth Sciences, Academia Sinica, Taiwan
8Jackson School of Geosciences, The University of Texas at Austin, Austin, TX, USA
9Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX, USA

Accepted 2012 November 13. Received 2012 October 31; in original form 2012 March 20

SUMMARY
A new generation, parallel adaptive-mesh mantle convection code, Rhea, is described and benchmarked. Rhea targets large-scale mantle convection simulations on parallel computers, and thus has been developed with a strong focus on computational efficiency and parallel scalability of both mesh handling and numerical solvers. Rhea builds mantle convection solvers on a collection of parallel octree-based adaptive finite element libraries that support new distributed data structures and parallel algorithms for dynamic coarsening, refinement, rebalancing and repartitioning of the mesh. In this study we demonstrate scalability to 122 880 compute cores and verify correctness of the implementation. We present the numerical approximation and convergence properties using 3-D benchmark problems and other tests for variable-viscosity Stokes flow and thermal convection.

Key words: Numerical solutions; Mantle processes; Dynamics: convection currents, and mantle plumes.

1 INTRODUCTION
Solid earth dynamics are governed by processes that occur over a wide range of time and length scales. A classic example is plate tectonics, where the large-scale motion of plates over timescales of millions of years and length scales of thousands of kilometres intimately couples to seismic processes that occur at timescales of minutes and less over lengths scales generally under 100 km. The upwellings associated with mantle convection are also typified by a wide range of length scales with large super plumes 1000's of kilometres across with small plumes detaching from their periphery that have thermal and mechanical boundary layers 100's of meters in thickness (Tan et al. 2011). Many of the transport processes that occur in mantle convection are thermochemical where chemical boundaries (e.g. next to subducted oceanic crust) can be sharp over submeter length scales.

The advent of petascale computing promises to make multiscale simulations of mantle convection and plate tectonics possible. Still, capturing global convection processes at realistic Rayleigh numbers requires resolution down to faulted plate boundaries. A uniform discretization of the mantle at for instance 1 km resolution would result in meshes with nearly a trillion elements, which is far beyond the capacity of the largest available supercomputers. An alternative is to employ adaptive mesh refinement (AMR) and coarsening methods that can reduce the number of unknowns drastically by placing resolution only where needed. Thus, AMR has the potential to enable high-resolution global mantle convection simulations, and to reduce the simulation wallclock time for many mantle convection problems significantly. Unfortunately, the added complexity of AMR methods can also impose significant overhead, in particular on highly parallel computing systems, because of the need for frequent readaptation and repartitioning of the mesh over the course of the simulation. Several recent studies have applied AMR methods to mantle convection, including (Davies et al. 2007; Stadler et al. 2010; Davies et al. 2011; Leng & Zhong 2011).

Here, we present the numerical strategies behind and verification of Rhea, a new generation adaptive mantle convection code that scales to hundreds of thousands of processors and has negligible overhead for all operations related to mesh adaptation. Rhea builds solvers for mantle convection problems on a collection of new libraries for parallel dynamic AMR (Burstedde et al. 2008b). It integrates parallel finite elements with forest-of-octrees based mesh adaptivity algorithms and includes support for dynamic coarsening, refinement, rebalancing and parallel repartitioning of the mesh. Rhea implements a parallel variable-viscosity non-linear Stokes solver, based on Krylov solution of the (stabilized) Stokes...
system (Burstedde et al. 2009), with pre-conditioning carried out by approximate block factorization and algebraic multigrid (AMG) V-cycle approximation of the inverse of the viscous and pressure Schur complement operators.

Rhea has been used previously to compute lithospheric and mantle flow models with resolutions below 1 km near-fault and subduction zones, and generally in areas where strain-weakening is observed; see Stadler et al. (2010) and Alisic et al. (2010). Here the parallel capabilities of Rhea have been essential to routinely perform simulations using O(10^6) compute cores. In this paper, we discuss the parallel adaptive mesh capabilities as well as the solvers used in Rhea in more detail. To verify the correctness of the implementation and to study convergence of the solution we use problems for which the exact solution is known, as well as benchmark problems previously used in the literature. Furthermore, we demonstrate that for problems of high Rayleigh number, adaptive meshes yield smaller errors compared to uniform meshes of the same element count, and report significant savings in the number of degrees of freedom and the overall run-time compared to highly resolved uniform meshes.

2 MANTLE CONVECTION EQUATIONS

The dynamics of mantle convection are governed by the equations of balance of mass, linear momentum and energy. Under the Boussinesq approximation for a mantle with uniform composition and the assumption that the mantle deforms as a viscous medium, the non-dimensionalized version of these equations reads (e.g. McKenzie et al. 1974; Zhong et al. 2000):

\[ \nabla \cdot \mathbf{u} = 0, \tag{1} \]

\[ \nabla p - \nabla \cdot [\mu(T, \mathbf{u}) (\nabla \mathbf{u} + \nabla \mathbf{u}^T)] = \text{Ra} T \mathbf{e}_z, \tag{2} \]

\[ \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla T \cdot \nabla T = \gamma, \tag{3} \]

where \( \mathbf{u} \), \( p \), \( \mu \), and \( T \) are the velocity, pressure, temperature and strain-rate-dependent viscosity and density, respectively; \( \gamma \) is the rate of internal heat generation; \( \mathbf{e}_z \) is the unit vector in the radial direction and \( \text{Ra} \) is the Rayleigh number that controls the vigour of convection and is defined as \( \text{Ra} = \alpha \rho_o \kappa \Delta T (D a R_o)^3 / (\kappa \mu_o) \). Here \( \alpha \), \( \rho_o \), \( \mu_o \) and \( \kappa \) are the reference coefficients of thermal expansion, density, viscosity and thermal diffusivity, respectively; \( \Delta T \) is the temperature difference across a mantle with relative thickness \( D \), and \( g \) is the gravitational acceleration. We use relative top and bottom radii \( R_1 = 1 \), \( R_2 = 0.55 \) throughout, which determines \( D = 0.45 \). Our length unit is thus the earth radius \( R_0 = 6371 \) km and not the mantle thickness \( D a R_0 \) which has been used elsewhere. In fact, removing \( D \) from the definition of \( \text{Ra} \) is equivalent to a scaling of units. According to eq. (10) in Zhong et al. (2008), the units of the temperature equation differ by \( D^2 \), and the velocity by \( 1/D \). These factors are taken into account when comparing numerical results. The boundary conditions (not shown) specify zero normal velocities and zero tangential traction at both the free surface and the core–mantle boundary, and impose fixed boundary temperature values.

In the stated form we do not account for variations in chemical composition, which are transported by the velocity field in analogy to the temperature (3). The diffusivity for the composition variable is negligible which would require a numerical method well suited to pure advection equations. One approach used elsewhere is to distribute tracer particles throughout the domain and advect them along streamlines (McNamara & Zhong 2004).

Eqs (1) and (2) are instantaneous and need to be satisfied at all times. Together they describe a non-linear Stokes system of partial differential equations that needs to be solved for velocity and pressure. The energy eq. (3) captures the evolution of the mantle and needs to be integrated forward in time, which is done after space discretization transforms it into a system of ordinary differential equations. Consequently, the numerical solution methods for these two systems as discussed in the next section are substantially different.

3 DISCRETIZATION AND SOLVERS

The Rhea code is custom written in C. It uses the Message Passing Interface to implement distributed parallelism. For the discretization of the temperature, velocity and the pressure in (1)-(3), we use (tri)linear finite elements on locally refined hexahedral meshes. These meshes are adapted to resolve features of the velocity, pressure or viscosity fields. Practical challenges, as well as the technical details required for parallel adaptive simulations, are discussed in Section 4. In this section, we focus on the discretization and on the solvers used in Rhea. Because of the large size of the matrices that result from the discretization, linear system cannot be solved using direct factorization-based solvers but have to be solved using iterative solution algorithms.

3.1 Variational formulation of Stokes equations

The finite element discretization is based on the weak form of the system of partial differential equations derived from (1) and (2) by multiplication with admissible test functions \( v \) and \( q \) (omitting the differentials dx, etc. for brevity),

\[ \int_\Omega [\nabla \cdot (p \mathbf{I} - \mu (\nabla \mathbf{u} + \nabla \mathbf{u}^T)) - \mathbf{f}] \cdot \mathbf{v} = 0 \quad \text{for all } \mathbf{v}, \tag{4a} \]

\[ \int_\Omega (\nabla \cdot \mathbf{u}) q = 0 \quad \text{for all } q, \tag{4b} \]

and integration by parts which yields

\[ A(\mathbf{u}, v) + B(v, p) + E(p, u, v) = F(v) \quad \text{for all } v, \tag{5a} \]

\[ B(\mathbf{u}, q) = 0 \quad \text{for all } q, \tag{5b} \]

where we use the definitions

\[ A(\mathbf{u}, v) = \int_\Omega \frac{\mu}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T) : (\nabla v + \nabla v^T), \tag{6a} \]

\[ B(\mathbf{u}, q) = -\int_\Omega (\nabla \cdot \mathbf{u}) q, \quad F(v) = \int_\Omega \mathbf{f} \cdot \mathbf{v}, \tag{6b} \]

\[ E(p, u, v) = \int_{\partial\Omega} [(p \mathbf{I} - \mu (\nabla \mathbf{u} + \nabla \mathbf{u}^T)) \mathbf{n}] \cdot \mathbf{v}, \tag{6c} \]

and \( \mathbf{f} = \text{Ra} T \mathbf{e}_z \) denotes the volume force. When we impose free-slip boundary conditions on \( \partial \Omega \), namely

\[ \mathbf{u} \cdot \mathbf{n} = 0, \quad \mathbf{v} \cdot \mathbf{n} = 0, \tag{7a} \]

\[ \mathbf{n} \cdot [(p \mathbf{I} - \mu (\nabla \mathbf{u} + \nabla \mathbf{u}^T)) \mathbf{n}] = 0, \tag{7b} \]
for an outside normal vector \( \mathbf{n} \) and any tangential vector \( \mathbf{t} \), we see that the term in (6c) vanishes. The discrete Stokes problem can then be written as the following saddle point system of equations:

\[
Q \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \quad \text{with} \quad Q = \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix},
\]

(8)

where \( \mathbf{u}, \mathbf{p}, \mathbf{f} \) denote the nodal values of the finite element approximations of \( \mathbf{u}, \mathbf{p}, \mathbf{f} \), respectively, and the matrices \( A, B, C \) are defined by inserting the subset of finite element shape functions \( \{ \phi_s \} \), that satisfy the boundary conditions (7a) into the corresponding bilinear forms \( A(\cdot, \cdot), B(\cdot, \cdot) \) and \( C(\cdot, \cdot) \). The purpose of the contribution

\[
C(p, q) = \sum_{n, x_n \in \Omega} \int_{\Omega} \frac{1}{\mu} (p - \Pi p)(q - \Pi q)
\]

is to stabilize the linear system (8). Here, \( \Omega_e \) for \( e = 1, 2, \ldots \) denote the finite elements and \( \Pi \) the \( L^2 \)-projection onto the space of element-wise constant functions. This is an \( L^2 \)-projection for the mapped (and thus possibly deformed) elements in physical space given by

\[
\Pi p = \Omega_e^{-1} \int_{\Omega_e} p \, dx,
\]

where the integral is approximated using numerical quadrature and \( \Omega_e \) is the volume of \( \Omega_e \). This stabilization is necessary because linear elements for velocity and pressure do not satisfy the inf-sup (or LBB, named after Ladyzenskaja, Babuška and Brezzi) condition for stability of numerical methods for saddle point problems; we refer to Elman et al. (2005), Bochev et al. (2006), Dohrmann & Bochev (2004) for details. Stabilized equal-order elements for velocity and pressure are convenient to implement and can be shown to converge at optimal order as the mesh is refined. Because of the choice of continuous elements for the pressure, element-wise mass conservation is not guaranteed for discontinuous pressure elements (Pelletier et al. 1989). Because of the stabilization matrix \( C \), the numerical solution satisfies the incompressibility condition only approximately. Local mesh refinement as discussed in Section 4 helps to control these unwanted effects. Note that the blocks \( A \) and \( C \) are symmetric and positive and, thus, (8) is an indefinite symmetric system.

The solution for the pressure is unique only up to a constant, which we address by penalizing the integral of the pressure over the domain. Concerning the velocity, all rigid-body rotations are non-trivial solutions to the homogeneous Stokes equations in a spherical geometry with free-slip boundary conditions. We remove this ambiguity by transforming the velocity field after each solve to a zero angular momentum state, as is done in Zhong et al. (2008).

### 3.2 Boundary terms and topography

The above derivation of the discrete Stokes system incorporates the free-slip boundary conditions, but at the same time removes information on the boundary traction from the formulation. Because the normal component of the traction vector,

\[
s = \mathbf{n} \cdot \left[ (\rho \mathbf{I} - \mu (\nabla \mathbf{u} + \nabla \mathbf{u}^T)) \mathbf{n} \right],
\]

(10)

is an important ingredient in determining the topography, we include a brief description of how it can be recovered in a post-processing step.

Assuming a Stokes solution \( (\mathbf{u}, p) \) that satisfies the boundary condition (7b), we can simplify the boundary term

\[
E(p, \mathbf{u}, \mathbf{v}) = \int_{\partial \Omega} (\mathbf{v} \cdot \mathbf{n}) \mathbf{p}.
\]

(11)

Note that this term can also be introduced as part of a Lagrangian functional to enforce (7a) in a variational form; in this case the normal traction \( s \) is identified with the Lagrange multiplier for the normal velocity component. Eqs (5a) and (11) hold for arbitrary velocity fields \( \mathbf{v} \), in particular those not satisfying \( \mathbf{v} \cdot \mathbf{n} = 0 \). We can exploit this fact by constructing a discretization of the normal field on the boundary,

\[
\mathbf{v}(x) = \sum_{n, x_n \in \partial \Omega} v_n \mathbf{n}_n \phi_n(x),
\]

(12)

defined by a coefficient vector \( \mathbf{v} = [v_n] \) whose index \( n \) loops over the subset of finite element shape functions \( \phi_n \) on the boundary, and \( \mathbf{n}_n \) denotes the vector that contains the normals of all boundary nodes \( x_n \). Inserting this function \( \mathbf{v} \) into (5a) and rearranging in terms of the coefficient vector \( \mathbf{v} \), we obtain a system of equations for the discretized normal traction \( s = \sum \mathbf{n}_n \phi_n \) with nodal values \( s = [s_n] \).

\[
\hat{M} s = (\mathbf{f} - A \mathbf{u} - B^T \mathbf{p}) \cdot \mathbf{n}.
\]

(13)

Here, the bar notation denotes matrices and vectors whose leading dimension corresponds to the boundary degrees of freedom, and the dot product is understood to collapse three coefficients into one independently at each node. The surface mass matrix \( \hat{M} \) with entries

\[
\hat{M}_{mn} = \int_{\partial \Omega} \phi_m(x) \phi_n(x) \, dx
\]

(14)

derives from the boundary integral in (11), with indices \( m, n \) restricted to the boundary nodes. In our numerical experiments we use a lumped version, that is a diagonal approximation, of \( \hat{M} \) that is easily invertible.

This procedure to obtain the normal traction is equivalent to the consistent boundary flux described in Zhong et al. (1993). Note that the method can be modified to compute tangential tractions for problems with prescribed flow at the boundaries, as is the case when plate motions are imposed.

An alternative approach to compute the tomography would be to allow the surface geometry of the domain to vary, and to compute the equilibrium between normal traction and gravity for every point at the surface. To ensure well-shaped elements, the surface deformation field would need to be extruded downward into the spherical shell which would couple the flow and deformation variables. We did not pursue this variant because of the expected increase in mathematical and numerical complexity.

### 3.3 Stokes solver

Because the coefficient matrix \( Q \) is symmetric and indefinite, we employ the pre-conditioned minimum residual iterative method (MINRES) for its numerical solution. MINRES (Paige & Saunders 1975) is a generalization of the conjugate gradient method to indefinite systems. Each MINRES iteration requires one application of the matrix \( Q \) to a vector and two inner products. The overall number of vectors stored does not increase with the number of
MINRES iterations, thus the memory footprint is small. Applications of finite element matrices are performed without assembling them in memory using loops over all finite elements. For a comprehensive discussion of the approach used in Rheo see Burstedde et al. (2009); for alternative approaches see Elman et al. (2005), May & Moresi (2008), Geenes et al. (2009).

To obtain a mesh-independent (or almost mesh-independent) number of iterations, that is a constant number of iterations as the problem size increases, one needs to employ a suitable pre-conditioner for (8). MINRES requires a symmetric and positive definite pre-conditioner. The block factorization

\[
\begin{pmatrix}
  A & B^T \\ 
  B & C
\end{pmatrix} =
\begin{pmatrix}
  I & 0 \\ 
  B A^{-1} & I
\end{pmatrix}
\begin{pmatrix}
  A & 0 \\ 
  0 & A^{-1} B^T + C
\end{pmatrix}
\begin{pmatrix}
  I & A^{-1} B^T \\ 
  0 & I
\end{pmatrix}
\]

(15)

does that \( Q \) is congruent to a block diagonal matrix. Neglecting the off-diagonal terms \( B A^{-1} \) and \( A^{-1} B^T \), motivates the use of the symmetric and positive definite matrix

\[
P = \begin{pmatrix}
  A & 0 \\ 
  0 & S
\end{pmatrix}, \quad \text{with} \quad S = B A^{-1} B^T + C
\]

(16)
as pre-conditioner. However, because the Schur complement \( S \) involves \( A^{-1} \), systems of the form \( P \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \) cannot be solved easily which makes \( P \) unsuitable as a pre-conditioner. Thus, we replace the Schur complement \( S \) by a lumped mass matrix weighted with the inverse viscosity \( \mu^{-1} \). For instance in Elman et al. (2005), it is shown that in the case of constant viscosity the resulting diagonal matrix is spectrally equivalent to \( S \). For varying viscosity and interface Stokes problems, similar results are obtained in Ginevich & Olshanskii (2009). Note that, when lumped, the pressure stabilization matrix \( C \) drops out. This is because of the fact that at the element level, constants are in the null space of \( C \). The resulting diagonal matrix reflects the local element size as well as the local viscosity. This is essential for favourable scalability of the MINRES iterations as the problem grows, and is particularly important for adaptively refined meshes.

Although a solve with the lumped mass matrix is trivial, the viscous block \( A \) is obtained from a discretization on highly heterogeneous meshes with large variations in the viscosity \( \mu \) (up to six orders of magnitude). To approximately calculate \( A^{-1} \) for a given residual \( \tilde{\mathbf{r}} \), we use one V-cycle of an AMG method (see, e.g., Briggs et al. 2000). Compared to geometric multigrid, AMG can have advantages because of its ability to account for variations in viscosity and adaptively refined meshes in the grid hierarchy. AMG requires a setup phase, in which a coarse grid hierarchy and corresponding restriction and interpolation operators are constructed. Parallel implementations of AMG require significant communication for this setup step (Chow et al. 2006; Falgout 2006). Generally, there is a trade-off between increased time/memory and the effectiveness of the coarse grid hierarchy. Rheo interfaces to two different parallel implementations of AMG, either to BoomerAMG

from the hypre\(^{2}\) package (De Sterck et al. 2006; The HYPRE Team 2007), or to the smoothed aggregation implementation ML\(^{3}\) from the Trilinos project (Gee et al. 2006). Both packages are available under free software licenses and allow the user to choose among various coarse-grid strategies, and to set parameters that influence the complexity of the coarse grid hierarchy and the interpolation and restriction operators. The pre-conditioner must be passed to the AMG packages in assembled form; the code to compute the matrix entries for \( A \) in a compressed sparse format is closely related to the code that performs the matrix-free application of \( A \) for the MINRES iterations.

3.4 Advection-diffusion solver

When the advection-diffusion eq. (3) is discretized with Galerkin finite elements, the transport term can give rise to spurious oscillations of the numerical solution. Among various stabilization methods, the streamline upwind Petrov–Galerkin (SUPG) approach can be formulated by multiplying the residual of (3),

\[
R(T) = \frac{\partial T}{\partial t} - \mathbf{u} \cdot \nabla T + \nabla^2 T,
\]

(17)

with the modified test function \( W + \mathbf{r} \cdot \nabla W \), where \( \mathbf{r} \) is a stabilization parameter:

\[
\int_{\Omega} R(T)(W + \mathbf{r} \cdot \nabla W) = 0.
\]

(18)

The value of \( \mathbf{r} \) is derived from the element Peclet number, that is the relation between advection, diffusion and element size (Brooks & Hughes 1982). Integration by parts and invoking Dirichlet boundary conditions for the test space, \( W_{|\partial\Omega} = 0 \), yields bilinear forms

\[
M(T, W) = \int_{\Omega} (W + \mathbf{r} \cdot \nabla W),
\]

(19a)

\[
G(T, W) = \int_{\Omega} (\mathbf{u} \cdot \nabla W),
\]

(19b)

\[
K(T, W) = \int_{\Omega} \nabla \cdot (\mathbf{I} + \mathbf{r} \mathbf{u} \otimes \mathbf{u}) \cdot \nabla W,
\]

(19c)

which give rise to the non-symmetric extended mass matrix \( \tilde{\mathbf{M}} \) and advection matrix \( \mathbf{G} \) and the extended stiffness matrix \( \tilde{\mathbf{K}} \), respectively. Thus, the SUPG stabilization can be interpreted as the introduction of artificial diffusion along the streamlines of the velocity field, and the semi-discrete energy equation becomes

\[
\mathbf{R}(T) = \mathbf{g} - \tilde{\mathbf{M}} \frac{\partial T}{\partial t} - (\mathbf{G} + \tilde{\mathbf{K}}) T = 0,
\]

(20)

\(^{2}\)For the parameters chosen in hypre, that is the coarsening strategy and the choice of smoothers we refer to Burstedde et al. (2009). We do not use hypre for the results described in this paper because the spherical boundary conditions and the vector-valued problem appeared to pose a difficulty for the version that we tested.

\(^{3}\)In ML, we use a processor-local (uncoupled) coarse grid aggregation scheme. When the number of unknowns per processor becomes small in the aggregation process, we repartition to a smaller number of processors. The new parallel partitioning often allows aggregation of unknowns that used to be on different processors. We use an aggregation threshold of 0.01, and 3 sweeps of a Chebyshev smoother for both the pre- and post-smoothing. For this choice of ML parameters, the small coarse grid problem is set up on a single processor and solved by a direct method.
where $g$ is the discretization of the heat generation rate $\gamma$ in (3). This system of ordinary differential equations is integrated in time by an iterative $\alpha$-predictor-corrector method that operates on pairs of vectors $(\vec{T}, \hat{T})$. For each time step $k$, the first iteration $i = 0$ is initialized by

$$T^0_i = 0, \quad T^i_k = T^i_k + \Delta t (1 - \alpha) \hat{T}_i_k. \quad (21)$$

The iterations proceed from $i$ to $i + 1$,

$$M^* \Delta T = R(T^i_i). \quad (22a)$$

$$T^{i+1}_i = T^i_i + \Delta T. \quad (22b)$$

$$T^{i+1}_i = T^i_i + \alpha \Delta t \Delta T. \quad (22c)$$

We use three iterations per time step and $\alpha = 1/2$, which provides second-order accuracy in the (implicit) limit $i \to \infty$. The matrix $M^*$ in (22a) can be understood as a pre-conditioner that may be approximate; we choose the diagonally lumped standard mass matrix which avoids an implicit solve. At the beginning of the simulation we obtain the time derivative $\hat{T}$ by executing one zero-length time step with the initial value of $T$. The spherical mantle convection code CitcomS (Zhong et al. 2008) uses a similar time integration scheme. The method is described in detail in Hughes (2000); see also Cottrell et al. (2009).

The velocity field $u$ enters the energy equation, and we update $u$ by a Stokes solve between each two time steps, thus decoupling it from the time integration. This amounts to an explicit, first-order splitting with respect to the velocity. This also means that the size of the time step is bounded by a Courant–Friedrichs–Lewy (CFL) condition that is dominated by the advection limit in the problems considered here.

4 ADAPTIVITY

Our goal is to simulate global mantle convection although taking into account the effects of faulted plate boundaries, trenches and other tectonic features. These features require a spatial resolution of approximately 1 km (Stadler et al. 2010). However, covering the volume of the mantle (which is of the order $10^{12}$ km$^3$) with an appropriately spaced grid would require roughly a trillion mesh elements, which is still beyond the storage capacity of even large supercomputers. Furthermore, significant overresolution would be created in areas such as the lower mantle, and any computation on this many elements would take an unacceptably long time.

We address this problem by AMR, that is we cover the mantle with elements of different sizes depending on the local resolution requirements. Because the number of elements per volume scales with the third power of the resolution, large savings in element number are possible. In our computations, we are able to achieve sub-km resolution of lithospheric features with less than $10^8$ elements globally; this amounts to savings of three orders of magnitude.

Various approaches to AMR exist, differing in the type of the elements (tetrahedra, hexahedra, prisms), their organization in space (unstructured or hierarchical) and the refinement pattern (conforming or non-conforming); see for example Flaherty et al. (1997), Berger & LeVeque (1998). Compared to a uniform mesh approach, AMR adds significant topological and mathematical complexity. Implementing AMR efficiently on large parallel computers is challenging, because of the irregularity of element-ordering schemes and communication patterns, and the requirement to distribute the computational work equally between all processors (parallel partitioning). Solving a stationary equation with a coarse-to-fine sweep of subsequently refined meshes, or evolving a dynamic problem with moving features in time, both call for frequent re-adaptation and repartitioning of the mesh over the course of the simulation. Ideally, the time needed for all AMR components should remain small compared to solver time, so that the gains accrued for having fewer degrees of freedom are not offset by inefficiencies of the algorithms for adaptivity (Luijten et al. 2007; Burstedde et al. 2010). For Rhea, we have chosen a hierarchical non-conforming approach based on a forest of octrees that satisfies all of the above requirements, described below.

4.1 Parallel adaptive meshes based on a forest of octrees

The term octree refers to a logical tree structure where each node is either a leaf or has eight child nodes. The recursive tree structure can be identified with a subdivision of a cubic volume, obtained by splitting the volume into eight similar child elements and applying these splits recursively where higher resolution is desired. The leaves of the octree, also called octants, then correspond bijectively to the mesh elements; see Fig. 1.

After defining a fixed ordering sequence for any eight elements created in a split, traversing the hierarchical tree structure left to right establishes a total ordering of all elements. This so-called space-filling curve is depicted in Fig. 1. Because of the shape of the curve, this particular child sequence is also called z-order. We use the total ordering not only to establish the storage sequence of elements and associated degrees of freedom, but also to determine the partition of the mesh into processor domains that have equal numbers of elements, which is essential for parallel load balancing. In addition, the locality properties of the space-filling curve allow near-optimal cache efficiency when looping over the elements in this order.

Efficient implementations of parallel adaptive octrees have been developed recently (Tu et al. 2005; Sundar et al. 2008). However, a single cube allows only a very restrictive set of computational domains. To lift this restriction, we decompose the domain into multiple octrees, conveniently called a forest of octrees, that are topologically equivalent to a hollow sphere. As an extension of the so-called cubed sphere approach we use 24 octrees, grouped into 6 caps of $2 \times 2$ octrees each. Because of the specific geometry of earth’s mantle, this subdivision provides nearly uniform aspect ratio of the octrees, which is inherited by the elements (see Fig. 2).

![Figure 1](https://example.com/fig1.png)

Figure 1. A 2-D cartoon of an octree on the left and the corresponding mesh on the right. The leaves of the octree, also called octants, correspond one-to-one to the elements of the mesh. A traversal of the leaves as indicated by the red curve (left) establishes a total ordering of the mesh elements in space (right), also called z-order because of its shape. Cutting this space-filling curve into equal-sized pieces creates a parallel partition of the elements, in this example between the three processors 0, 1 and 2.)
4.2 Handling of non-conforming meshes

Rhea uses a continuous trilinear finite element discretization to represent all variables. A field such as the temperature is approximated by a linear combination of basis functions that in our case are trilinear, that is defined as tensor products of linear functions in the three space dimensions. Our meshes are non-conforming, which means that adjacent elements can have different sizes and the endpoints of neighbouring faces or edges need not coincide; see Fig. 3. This results in nodes that are ‘hanging’, that is that do not correspond to element basis functions on all adjacent elements. To enforce global continuity of finite element functions, the space of element-local basis functions must be restricted to a continuous subset. This can be done through algebraic constraints as outlined next.

Let us introduce local basis functions on each element e, denoted by $\psi_i^e(x)$, $i \in \{1, \ldots, 8\}$. We choose nodal basis functions that assume the value 1 at exactly one of the eight nodes $x_j^e$ of the element, $\psi_i^e(x_j^e) = \delta_{ij}$. These element-local basis functions are zero outside of the element. A function that is trilinear on each element but possibly discontinuous between elements can be represented by element-local coefficients $c_i^e$ as $f(x) = \sum_{i=1}^{8} c_i^e \psi_i^e(x)$. Vector-valued functions such as the velocity field and force term are represented by coefficients $c_i^e \in \mathbb{R}^3$; we do not use spherical coordinates or other non-Cartesian coordinate systems.

To fully specify the element-local basis functions $\psi_i^e(x)$, we take into account both the transformation from the octree coordinates $\vec{x} = (\xi, \eta, \zeta)$ into physical coordinates $x(\vec{x})$ and the scaled shift $\vec{r} = (r, s, t)$ from the reference element $r \in [0, 1]^3$ into the octant that corresponds to element $e$, covering a cubic subvolume $V_e$ of octree coordinate space. Combined with a tensor-product ansatz for three space dimensions, this yields

$$\psi_i^e(x) = \psi_i^e(x(\vec{x}(\vec{r}))) = \epsilon_{ij}(\vec{r}) \psi_{ij}(s) \epsilon_{ij}(t) = (\prod_{i} \epsilon_{ji})^{\vec{r}}(\vec{r}).$$

The linear basis functions $\epsilon_{ji}$ are 1 at one end of the reference interval and zero on the other, based on the coordinate direction $d$ and the corner number $i$.

Adaptive refinement and coarsening produces non-conforming meshes where nodes of one element are not necessarily nodes of a neighbouring element but may instead be hanging (see Fig. 3). Continuity of the trilinear representation can be enforced by identifying only the non-hanging nodes with global independent degrees of freedom $g_n$, where $n \in \{1, \ldots, N\}$ and $N$ is the number of

![Figure 3. Illustration of a hanging face in a non-conforming adaptive discretization. The values of a variable field at the hanging nodes $h_1, h_2, h_3$ are computed from the values at $a_1, a_2, a_3$ through interpolation. For instance, for the edge-hanging node $h_1$ the value is given by the mean of the values at $a_2$ and $a_3$, similarly, the value at the face-hanging node $h_3$ is given by the mean of $a_1, a_2, a_3$.](image-url)
independent nodes, and generating the element-local coefficients through an interpolation matrix $S$,

$$c_f = \sum_k S_{k}^e \phi_k.$$

The matrix $S$ is sparse and thus never assembled or stored; instead, the submatrix $S_e = (S_{k}^e) \in \mathbb{R}^{8 \times N}$ is applied for an element $e$. For vector-valued variables $c_f, g_e \in \mathbb{R}^3$, $S_e$ is applied to each component in turn. Through the identification

$$f(x) = \sum_{c, f} c_f \psi_f(x) = \sum_{c, f} S_{k}^e g_e \psi_f(x) = \sum_g g_e \phi_e(x),$$

we define global basis functions $\phi_e = \sum_{c, f} S_{k}^e \psi_f$ that are locally supported and continuous by construction.

For parallel computation we distribute the global degrees of freedom among the processors. Hanging nodes are always understood as processor-local and their values are interpolated when needed from associated independent nodes (Fig. 3). We assign ownership of an independent node to the lowest numbered processor whose elements touch it. Given local copies of one layer of off-processor elements (so-called ghost elements), each processor can determine the hanging status and processor ownership of all nodes touching any of its elements without further communication. To determine a globally unique numbering of all degrees of freedom, each processor counts its owned independent nodes and shares this number with all other processors. Every processor then offsets its owned node indices by the number of independent nodes owned by all lower numbered processors.

The values of an independent node may be needed on other processors than its owner, either through an independent node on the processor boundary or through referral by an off-processor hanging node that depends on its value for interpolation. Thus, for each independent node we maintain a list of sharing processors. Most independent nodes are away from interprocessor boundaries because of the surface-to-volume ratio of the parallel partition; these have no sharers. Those on a processor boundary usually have a small and bounded number of sharers because of the locality properties of the space-filling curve. In fact, for typical examples covered here the maximum number of sharers is less or equal seven (which is expected for hexahedral meshes), with an overall average number of sharers per node between 0.5 and 0.1.

The authoritative value for a degree of freedom is stored on its owner processor; we use the sharer lists to send its value to other processors, and to receive updates when necessary. The algorithms for creation of the ghost layer and the trilinear node numbering for a forest-of-octree mesh are detailed in Burstedde et al. (2011). Fig. 4 illustrates the global node numbering and sharer lists.

Although all finite element variables are stored as global degrees of freedom it is more convenient to apply discretized operators, such as mass or stiffness matrices, using the element-local formulation. With the definitions introduced above we decompose for example the mass matrix $M_e = (M_{ef}) \in \mathbb{R}^{8 \times N}$ as follows,

$$M_{ef} = \int_{\Omega_e} \phi_e(x) \phi_f(x) dx = \sum_{c, i, j} \int_{A_e} S_{k}^e \psi_f(x) S_{l}^e \psi_j(x) dx,$$

or, equivalently in matrix notation,

$$M = \sum_e S_e^T M_e S_e \quad \text{with} \quad M_{ef} = \int_{A_e} \psi_f(x) \psi_j(x) dx \quad \text{(23)}.$$
yielding under high strain rates. To keep the number of elements small in dynamically refined AMR problems, meshes also have to be coarsened wherever high resolution is no longer necessary. The algorithmic framework for dynamic adaptivity used in Rhea is described in Burstedde et al. (2008b).

Accurate element-based error indicators are essential for effective solution adaptivity. Various choices for such error indicators are summarized next.

**Physics-based error indicators.** Often, physical intuition can be used to devise an indicator for adapting the mesh to the problem: simple examples are element-wise temperature gradients for the energy equation and viscosity gradients or numerical velocity divergence for the Stokes equation. In this case, the error indicator can be given by a weighted sum of the element integrals of the local temperature gradient \( \nabla T_e \), the viscosity gradient \( \nabla \mu_e \), and the second invariant of the strain rate \( \varepsilon_{ij} \varepsilon_{ij} \):

\[
x_e = w_1 \| \nabla \mu_e \| + w_2 \| \nabla T_e \| + w_3 \nabla T_e \cdot \varepsilon_e + w_4 \varepsilon_{ij} \varepsilon_{ij},
\]

where \( \varepsilon \) denotes element-based quantities.

**Residual-based error indicators.** For some problems, error indicators are available that can be proven to provide bounds for the actual error. These indicators (also called error estimators) involve element equation residuals and jumps of the solution derivatives across element faces, or they require a reconstruction of the solution over a patch consisting of several elements (Ainsworth & Oden 2000).

**Goal-oriented error indicators.** Often, one is not interested in minimizing the global discretization error, but in obtaining high accuracy in a certain quantity of interest, for instance, the solution in a part of the domain or its mean. Goal-oriented error indicators (Ainsworth & Oden 2000; Becker & Rannacher 2001; Oden & Prudhomme 2001) lead to meshes that target maximal accuracy in the quantity of interest. However, they require the solution of an adjoint problem, which makes them comparatively costly (Burstedde et al. 2009).

Having an error indicator at hand, it remains to decide which elements to refine and coarsen. Several strategies can be used; for instance, to coarsen/refine elements with an error indicator under/above a certain threshold. Alternatively, one can coarsen and refine a certain percentage of elements, because it is often desirable to control the size of the simulation. This approach relies on choosing appropriate refinement/coarsening thresholds to obtain a target number of elements. In a parallel simulation environment, these thresholds can be determined by the iterative bisection algorithm MarkElements described in Burstedde et al. (2008b). For most time-dependent simulations we use the latter strategy to keep the number of elements constant throughout the simulation. When starting with an element number that differs from the target, it is reached automatically within a tolerance of 3 percent after the first few adaptation intervals; see also the next section.

4.4 Mesh adaptation for time-dependent problems

Simulation of the energy transport of mantle convection (3) often reveals the creation and disappearance of localized features, and a motion of plumes and other structures through space. Thus, the mesh needs to be adapted dynamically to resolve physics that evolve with time. Although adaptation after each time step is technically possible, in practice it is sufficient to only adapt the mesh after a time interval corresponding to a fixed number of time steps (e.g., 10-50 steps). To obtain a properly adapted mesh for such a time interval, we determine the maximum in time of an error indicator separately for each element. Implementing this strategy naively however would allow features that move from a finely resolved into a coarsely resolved area during the same interval, resulting in a loss of information. This risk can be eliminated by a multipass algorithm as described in Sun & Wheeler (2004). Here, one or more passes for estimation can be executed to gather the error information, which is then used to create a new mesh and run the simulation pass starting from a checkpoint that was saved previously. Our adaptation of this process is illustrated in Fig. 5 and has been described in Burstedde et al. (2008b) in more detail.

The numerical result of the estimation pass is discarded after mesh adaptation and can thus be approximate. For the simulation of mantle convection, where solving the Stokes system consumes the majority of computation time, we hold the flow solution constant for the error estimation pass to avoid solving the Stokes equation at each estimation time step.

5 TESTS AND BENCHMARKS

The purpose of this section is twofold: First, we provide evidence for the correctness of the Rhea code by comparing numerical against analytical solutions and studying convergence rates. Secondly, we analyse the potential of adaptively refined meshes for typical mantle convection benchmarks and discuss for which scenarios adaptive mesh capabilities are most beneficial.

5.1 Analytical solutions for the Stokes equations

Analytical solutions can be employed effectively to demonstrate the correctness of the implementation and to verify convergence rates for finite element discretizations of partial differential equations. However, it is generally not possible to construct an analytical solution for a given right-hand side \( f \). What is possible, in contrast, is to postulate velocity and pressure fields and to insert them into the system of equations to derive an appropriate \( f \) that is used as forcing for the simulation. The postulated and computed velocity and pressure can then be compared. This approach is often called the method of manufactured solutions. It can reveal errors in the implementation and deliver precise convergence rates of numerical approximations. In this section we present two manufactured solutions, namely a polynomial and a trigonometric formulation.
5.1.1 Polynomial solution benchmark

We begin by postulating a simple polynomial solution for the Stokes equations Dohrmann & Bochev (2004),

\[
\mathbf{u} = \begin{pmatrix}
    x + x^2 + xy + x^3 y \\
    y + xy + y^2 + x^2 y^2 \\
    -2x - 3xz - 3yz - 5xz yz
\end{pmatrix}, \quad (25a)
\]

\[p = xyz + x^3 y z - 5/32, \quad (25b)\]

which is divergence-free. Inserting this solution into the momentum equation with a given viscosity \(\mu\), we obtain the right-hand-side forcing

\[
f = -\nabla p + \mu \begin{pmatrix}
    -2 - 6xy \\
    -2 - 2x^2 - 2y^2 \\
    10yz
\end{pmatrix}
\]

\[-\mu_x \begin{pmatrix}
    2 + 4x + 2y + 6xy \\
    x + 3x + y + 2xy^2 \\
    -3x - 10x y z
\end{pmatrix} - \mu_y \begin{pmatrix}
    x + 3x + y + 2xy^2 \\
    2 + 2x + 4y + 4x^2 y \\
    -3y - 10y z x
\end{pmatrix} - \mu_z \begin{pmatrix}
    -3x - 10x y z \\
    -3y - 5x z \\
    -4 - 6x - 6y - 10x^2 y
\end{pmatrix}, \quad (26)\]

We also impose exact velocity boundary conditions derived from (25). Then we solve the Stokes equations with Rheo and compute the \(H^1\)-norm of the difference between numerical and exact solutions \((\mathbf{u}_h, p_h)\) and \((\mathbf{u}, p)\), respectively,

\[
|\mathbf{u} - \mathbf{u}_h|_{L^2} := \left( \int_{\Omega} (\mathbf{u}_h - \mathbf{u})^2 \, dx \right)^{1/2}, \quad (27a)
\]

\[
|p - p_h|_{L^2} := \left( \int_{\Omega} (p_h - p)^2 \, dx \right)^{1/2}, \quad (27b)
\]

In Table 1 we summarize the convergence results for constant viscosity \(\eta = 1\) on a \(45^\circ \times 45^\circ\) portion of the spherical shell as well as the global mantle geometry, for which the radius has been scaled to 1. In addition, we show the number of MINRES iterations to achieve a drop in the residual by a factor of \(10^6\).

| Mesh | \(|\mathbf{u}^r - \mathbf{u}_b|_{L^2}\) | \(|p^r - p_b|_{L^2}\) | #Iter | #Elem |
|------|-----------------|-----------------|--------|--------|
| 8^3  | 8.08e-4         | 3.85e-2         | 47     |        |
| 16^3 | 2.25e-4         | 1.15e-2         | 47     |        |
| 32^3 | 5.84e-5         | 3.43e-3         | 54     |        |
| 64^3 | 1.46e-5         | 1.03e-3         | 54     |        |
| 24 \times 4^3 | 1.53e-2 | 2.66e-1 | 75     |        |
| 24 \times 8^3 | 4.40e-3 | 8.95e-2 | 50     |        |
| 24 \times 16^3 | 1.16e-3 | 2.98e-2 | 57     |        |
| 24 \times 32^3 | 2.94e-4 | 1.01e-2 | 67     |        |

Table 1. Polynomial solution example: Error between exact and numerical solution for constant viscosity \(\mu = 1\) for a \(45^\circ \times 45^\circ\) portion of the spherical shell (upper part) and the full mantle geometry (lower part). The number of MINRES iterations is reported in the rightmost column.

We include results for a spatially smoothly varying viscosity

\[
\mu = \exp(1 - 4(x(1-x) + y(1-y) + z(1-z))) \quad (28)
\]

in Table 2. With the above formula, the viscosity varies over six orders of magnitude, which requires a finer mesh to resolve its gradients. We also include two adaptive solves with different error indicators (the norm of the viscosity gradient, \(w_1 = 1\) in (24), and the divergence residual \(\nabla \cdot \mathbf{u}_h\), respectively) and see that these further decrease the velocity and pressure residuals.

As expected from the theory Dohrmann & Bochev (2004), with each uniform mesh refinement (that halves the mesh size) the velocity error decreases by a factor of 4, and thus the convergence rate is of order 2. For the pressure error, finite element theory only predicts a decrease of linear order for a uniform refinement. However, our numerical tests yield a better value of approximately 1.6, which is also observed in Dohrmann & Bochev (2004). Note that the number of iterations required to solve the problems is almost constant across different refinement levels. Such a mesh-independent convergence rate of solvers is necessary to obtain optimal scalability when problems become very large, and constitutes the main motivation to employ multigrid-type pre-conditioners.

5.1.2 Diverging flow Stokes example

We now use an example that models diverging flow that has similarities to the mantle flow found at a midocean ridge. The viscosity \(\mu\) and the forcing \(f(r, \theta, \psi) = (f_r, f_\theta, f_\psi)\) are, in spherical coordinates, given as follows:

\[
\mu = r^2, \quad f_r = \frac{8(1 + r^5) \left(1 - \tanh \left( \frac{\psi}{w_\mu} \right)^2 \right)}{5w_\mu r^3},
\]

\[
f_\theta = \frac{2(1 - r^3) \left(1 - \tanh \left( \frac{\psi}{w_\mu} \right)^2 \right) \left(3 \tanh \left( \frac{\psi}{w_\mu} \right)^2 - 1 \right)}{5w_\mu r^3 \sin(\theta)^2}.
\]
\[ f_0 = \frac{\sin^3(\theta) \sin^2(\theta) - \frac{\theta^3}{r^2}}{\sin(\theta) \sin^3(\theta)} + \frac{r \cos(\theta) \sin(3\theta)}{r^2} + \frac{3 \theta^2}{r^2} \]
\[ f_r = \frac{\sin(\theta) \sin^2(\theta) \left( 1 - \tanh\left( \frac{\varphi}{\psi_0} \right) \right)^2 + \frac{2(1 + r^2) \tanh\left( \frac{\varphi}{\psi_0} \right)}{\sin^2(\theta) \sin^3(\theta)} - \frac{14r^3 \sin(\theta) \sin(3\theta)}{\sin^2(\theta) \sin(3\theta)}. \]

Above, the parameter \( \psi_0 > 0 \) controls the smoothness of the ridge. The smaller \( \psi_0 \), the faster and more localized is the radial flow under the ridge; see Fig. 6. It can be verified that the solution \((u_r, u_\theta, u_\varphi, p)\) to the ridge Stokes flow problem is given by

\[ u_r = \frac{(1 - r^2) \left( 1 - \tanh\left( \frac{\varphi}{\psi_0} \right) \right)^2}{\sin^3(\theta) \sin^2(\theta)} \]
\[ u_\theta = 0 \]
\[ u_\varphi = r^3 \sin(\theta) \tanh\left( \frac{\varphi}{\psi_0} \right) \]
\[ p = r^2 \sin(\theta) \sin(3\theta) + \frac{\theta^3}{r^2} \]

To study the accuracy of our numerical method, we set the velocity on the boundary to the exact solution and then solve the Stokes problem for the forcing given above. Again, we report the \( L^2 \)-norm of the difference between numerical and exact solution \((u_r, p, u_\theta, u_\varphi)\) and \((u_r, p, u_\theta, u_\varphi)\) respectively; see Table 3. Note that for the same mesh, for large \( \psi_0 \), the numerical solution is a better approximation of the exact solution. This can be explained by the fact that for small \( \psi_0 \), the solution becomes less smooth, which makes the numerical solution of the problem harder.

Furthermore, note that the number of iterations remains stable as the mesh is refined, enabling the efficient solution of large-scale problems.

### 5.2 Benchmarks for Stokes solver

We now use a common Stokes benchmark problem (see e.g. Choblet et al. 2007; Zhong et al. 2008) to verify the flow solution of the Stokes solver, as well as the computation of surface and core-mantle boundary (CMB) topography. The problem uses constant viscosity, the Rayleigh number is unity and the temperature is specified as a delta function at a radius \( r_0 \) in the radial direction and a spherical harmonic function \( Y^m_l \) of degree \( l \) and order \( m \) in the tangential directions, that is

\[ T(r, \varphi, \theta) = \delta(r - r_0) Y^m_l(\varphi, \theta). \]

The \( \delta \)-function in the radial direction is approximated by a triangle with unit area:

\[ \delta(r - r_0) = \begin{cases} \frac{n_{r_0}}{r - r_0} & \text{if } r = r_0, \\ 0 & \text{otherwise}, \end{cases} \]

where \( n_{r_0} \) is the number of elements in the radial direction in a uniform mesh. The spherical harmonic function is described by

\[ Y^m_l(\varphi, \theta) = \cos(m\varphi) \sin^{l-m}(\theta) \]

The normalized associated Legendre polynomial \( p_m \) is related to the associated Legendre polynomial \( P_m \) by:

\[ p_m(\theta) = \frac{(2l + 1)(l - m)!}{2\pi(1 + \delta_m)(l + m)!} P_m(\theta). \]

The usual free-slip boundary conditions are used. Because of properties of the spherical harmonics functions, this setting allows the computation of the Stokes flow by solving numerically an ordinary
differential equation for the coefficient of the spherical harmonic; see Hager & Richards (1989). This semi-analytical solution is used to compare with the finite element-based solution obtained in Rhea.

As in Hager & Richards (1989), Zhong et al. (2008) and Choblet et al. (2007), we report the responses of flow and topography at the top surface and the CMB when changing the radius \( r_h \), at which the force is imposed; see Fig. 7. The mesh size is varied from \( 2^n \) to \( 2^n \) elements. We perform a detailed error analysis for the various resolutions (Fig. 8). The errors in response functions with respect to the semi-analytical solution decrease quadratically with increasing resolution, as expected. The error increases with increasing spherical harmonic degree as the complexity of the forcing is made larger. In addition, the error decreases with increasing forcing depth. Because of the spherical geometry of the domain, elements have smaller dimensions at larger depth and therefore errors with respect to the semi-analytical solution are smaller. These results are in agreement with those of Zhong et al. (2008).

We use this benchmark problem to assess parallel scalability as we simultaneously increase the problem size and the number of processing cores. A breakdown of different components of Rhea by run-time is presented in Table 4. We observe that the number of iterations remains essentially constant over a three-orders-of-magnitude increase in problem size and number of processor cores. Thus, we observe algorithmic scalability out to 123 000 cores and 631 M elements (which corresponds to roughly 2.5B degrees of freedom). Parallel scalability can be assessed by observing the growth in CPU time of the dominant components of the Stokes solver: AMG setup at the beginning of each Stokes solve, the matrix-vector product time for each Krylov iteration and the V-cycle time associated with the application of the AMG pre-conditioner at each Krylov iteration. As can be seen, the latter two times remain relatively stable over the thousand-fold increase in problem size and number of cores (for perfect weak scaling, they would not grow at all). However, the AMG setup time experiences large growth above \( 10^5 \) processor cores. This is understandable, given the large communication induced in the AMG setup, and is merely a problem in practice, because even at 123 000 cores, the AMG setup time is still dominated by the total time taken (across Krylov iterations) in matrix-vector products and V-cycle applications; moreover, the AMG setup can often be reused for several Stokes solves.

5.3 Time-dependent benchmark

The time-dependent solver in Rhea is benchmarked using a spherical harmonic temperature perturbation, superimposed onto a conductive profile in a shell. The temperature field is defined as follows:

\[
T(r, \theta, \phi) = \frac{r_0 (r - r_s)}{r (r_h - r_s)} + \varepsilon_s \cos(m \phi) + \varepsilon_t \sin(m \theta) p_{m, \alpha}(\theta) \sin \frac{\pi (r - r_h)}{(r_h - r_s)}, \tag{35}
\]
Figure 8. Errors in response functions with respect to the semi-analytical solution for surface topography (s), CMB topography (b), velocity at the surface ($V_s$) and velocity at the CMB ($V_b$), for spherical harmonic degrees 2, 5, 8 and 15 in a sphere with uniform viscosity. Three forcing depths are shown left to right, namely $0.25d$, $0.5d$ and $0.75d$. 
Table 4. Weak scaling with approximately 5000 elements per core for the midocean ridge Stokes example, obtained on the Jaguar supercomputer. The mesh contains elements of three different sizes determined by a strain rate error indicator and the viscosity varies over one order of magnitude. Reported are the number of MINRES iterations to decrease the residual by a factor of $10^5$, the time for the AMG setup (using ML from Trilinos), the overall time for matrix-vector and inner products and for the V-cycles in MINRES. ML employs the recursive coordinate bisection repartitioning algorithm from ZOLTAN to improve the parallel efficiency of the multigrid hierarchy.

<table>
<thead>
<tr>
<th>#Cores</th>
<th>#Elem/core</th>
<th>#Elem</th>
<th>#Iter</th>
<th>Setup time [s]</th>
<th>Matvecs time [s]</th>
<th>V-cycle time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>5,800</td>
<td>700K</td>
<td>24</td>
<td>1.39</td>
<td>2.75</td>
<td>2.88</td>
</tr>
<tr>
<td>960</td>
<td>4,920</td>
<td>4,720M</td>
<td>22</td>
<td>2.30</td>
<td>3.94</td>
<td>2.89</td>
</tr>
<tr>
<td>7680</td>
<td>4,805</td>
<td>369M</td>
<td>33</td>
<td>4.07</td>
<td>3.99</td>
<td>5.72</td>
</tr>
<tr>
<td>61440</td>
<td>5,145</td>
<td>316M</td>
<td>21</td>
<td>3.60</td>
<td>4.60</td>
<td>9.03</td>
</tr>
<tr>
<td>122880</td>
<td>5,135</td>
<td>631M</td>
<td>26</td>
<td>11.48</td>
<td>6.29</td>
<td>8.39</td>
</tr>
</tbody>
</table>

Figure 9. Temperature field at steady state for the time-dependent benchmark. Shown are contours at temperatures 0.05 (blue) and 0.5 (yellow). Left: isoviscous model. Right: model with viscosity variation of factor 20.

where $\rho_{\text{min}}$ is given by (34). The parameters $\epsilon_1$ and $\epsilon_2$ are set to 0.01, and the degree $l$ and order $m$ are 3 and 2, respectively. The viscosity is given by:

$$\mu = \exp \left( E (0.5 - T) \right),$$

where the viscosity variation within the model is determined by the activation energy $E$. Cases with $\Delta \mu = 1$ (isoviscous) and $\Delta \mu = 20$ are run. These cases have also been reported by Bercovici et al. (1989) and Zhong et al. (2000) for $\Delta \mu = 1$, and by Ratcliff et al. (1996), Yoshida & Kageyama (2004) and Stemmer et al. (1996) for $\Delta \mu = 1, 20$. Zhong et al. (2008) showed results for a wide range of viscosities from $\Delta \mu = 1$ to $10^5$. We use a Rayleigh number of $7.6818 \times 10^4$. The mesh is uniform at level 5, corresponding to 32 elements in the radial direction, which is comparable to that of Zhong et al. (2008).

The resulting temperature field in steady state has tetrahedral symmetry for the viscosity ranges tested here. The steady-state temperature field shows four well defined plume-like upwellings, and a set of interconnected downwelling sheets(Fig. 9). The time-series of average temperature, average root mean square velocity and Nusselt numbers at the top and bottom of the mantle reproduce results described by, for example Zhong et al. (2008) (Fig. 10).

5.4 Adaptive resolution of rising plume

In the final benchmark presented here, we illustrate the effectiveness of mesh adaptation. We compute plume models in a $45^\circ \times 45^\circ$

section of a spherical shell, with an initial temperature field given by

$$T(x) = T_0 + \exp \left( -\frac{1}{2\sigma^2} |x - x_0|^2 \right),$$

where $\sigma = 1/20$ determines the extent of the anomaly and $x_0$ denotes its centre, situated $D/10$ below the core-mantle boundary (which is outside of the domain, but still has an effect in the lower mantle). A thermal boundary layer is used at the bottom of the domain for $r < r_b + w_{\text{BL}}$, with $w_{\text{BL}} = 0.0785$ chosen to cover the
bottom-most \( w_{\text{TL}}, R_h = 500 \) km. This temperature profile is described using an error function:

\[
T_\theta = 1.0 - 0.5 \frac{c_{\text{ef}}(r - r_s)}{w_{\text{TL}}/2}.
\]  

(38)

Elsewhere, the background temperature \( T_\theta \) is 0.5. The Rayleigh number is set to \( 10^4 \) and \( 10^5 \), respectively (Fig. 11). The viscosity is given by (36), with \( E = 7.0 \). The solutions for meshes with various amounts of coarsening are compared to the solution obtained on a uniform mesh. We start with a static uniform mesh in both cases, using mesh level 7 (\( 2^{21} \) elements) for the model with \( Ra = 10^4 \), and mesh level 8 (\( 2^{24} \) elements) for the model with \( Ra = 10^5 \). Time-series of the average temperature, average root mean square velocity and Nusselt numbers at the top and bottom of the mantle are computed. We then decrease the target number of elements using dynamic adaptive coarsening in consecutive model runs (Table 5), but only allow a maximum decrease in resolution of two mesh levels. Starting with the previous uniform mesh, the target number is reached in the first few adaptation cycles and kept constant within a 3 per cent range afterwards. For this adaptive coarsening, an error indicator is used with weights \( w_2 = w_3 = 1 \) in (24) for the \( \nabla \cdot T \) and \( \nabla \cdot \mathbf{e} \) terms, respectively; the other terms are not activated. The coarsened meshes are then compared to models with uniform meshes with the same total number of elements.

The time-series show that in the case with Rayleigh number \( 10^5 \), a steady configuration develops (Fig. 12). Quantitative comparisons are provided in Table 5. The models with \( 2^{18} \) elements reproduce the results of the uniform high-resolution mesh (\( 2^{21} \) elements) well, and the adaptive better than the uniform. The model with \( 2^{18} \) elements coarsened from level 7 has a smaller \( V_{\text{rms}} \) error than the model with uniform mesh at level 6, as does the coarsened model with \( 2^{15} \) elements compared to the model with uniform mesh at level 5. Comparing the uniform high-resolution model with the adapted one at the same number of elements, it can be seen that adaptivity allows an overall 8\( \times \) reduction in both elements and run-time, only with a minor loss in accuracy. Choosing increasingly coarser models, the errors increase gradually, which is expected at this Rayleigh number. The temperature field is smooth and does not show sharp features.

The models with a Rayleigh number of \( 10^5 \) show a much increased sensitivity to mesh resolution. The plume is narrower, temperature gradients are sharper and flow velocities are larger with increased Rayleigh number. In this model, no steady-state solution is achieved. The original plume is only stable up to \( t_{\text{model}} \approx 3 \times 10^{-4} \), and is then replaced with smaller, more ephemeral features for the duration of the model run. These features are harder to resolve than the original plume, and therefore a uniform reduction in the number of elements under-resolves the solution and eventually fails (see Fig. 12). In contrast, an adaptive coarsening from level 8 to \( 2^{21} \) elements reproduces the results from the uniform level 8 mesh (\( 2^{24} \) elements) well in 8\( \times \) less run-time, and provides a 12\( \times \) smaller error than the model with a uniform level 7 mesh at the same number of elements (see again Table 5). This adaptive model is the only lower cost variant that yields an acceptable error. Considering a further reduction of the problem size, the model with \( 2^{18} \) elements...
Table 5. Comparison of the time evolution of a rising plume on static uniform and dynamically adapted meshes, with activation energy $E = 7$. The first column lists the Rayleigh number and the non-dimensional model time at which errors are assessed. The second column indicates the mesh level at the start of the simulation, whereas the third column contains the number of elements after adaptive meshing. The fourth column shows the number of cores used for the computation. The fifth column shows the total compute time $t_{\text{comp}}$, computed as the overall run time in seconds times the number of cores used for the computation. The last column shows the relative error in $T_{\text{rms}}$ compared with the highest resolution uniform mesh case.

<table>
<thead>
<tr>
<th>Rayleigh</th>
<th>Level</th>
<th># Elements</th>
<th># Cores</th>
<th>$t_{\text{comp}}$ (s)</th>
<th>$T_{\text{rms}}$ error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{4}, 8.0 \times 10^{-2}$</td>
<td>Level 7 uniform</td>
<td>$2^{16}$ (128)</td>
<td>192</td>
<td>1.6268 x $10^7$</td>
<td>21181 x $10^6$</td>
</tr>
<tr>
<td></td>
<td>Level 7 coarsened</td>
<td>$2^{18}$</td>
<td>96</td>
<td>$9.1380 \times 10^8$</td>
<td>0.044</td>
</tr>
<tr>
<td></td>
<td>Level 6 uniform</td>
<td>$2^{18}$ (64)</td>
<td>24</td>
<td>3.2419 $\times 10^9$</td>
<td>0.083</td>
</tr>
<tr>
<td></td>
<td>Level 7 coarsened</td>
<td>$2^{16}$</td>
<td>24</td>
<td>2.0125 $\times 10^9$</td>
<td>0.159</td>
</tr>
<tr>
<td></td>
<td>Level 7 coarsened</td>
<td>$2^{15}$</td>
<td>24</td>
<td>6.0709 $\times 10^9$</td>
<td>0.226</td>
</tr>
<tr>
<td></td>
<td>Level 5 uniform</td>
<td>$2^{15}$ (32)</td>
<td>8</td>
<td>1.0199 $\times 10^9$</td>
<td>0.800</td>
</tr>
</tbody>
</table>

Adaptively coarsened from a level 6 mesh has a much reduced error compared with a uniform level 6 mesh (also 2$^{18}$ elements). These results indicate that adaptive coarsening can preserve high accuracy although providing a much faster time to solution. When Rayleigh numbers become large, the adaptive simulation becomes increasingly favourable compared to a uniform mesh simulation of the same element count.

6 DISCUSSION AND CONCLUSIONS

In this paper we have presented the design and functionality of the Rhea code for instantaneous and time-dependent simulation of mantle convection. The uniqueness of Rhea lies in the combination of dynamic AMR capabilities that enable the resolution of multiple scales, and large-scale parallel scalability that enables efficient use of petaflop-class supercomputers. Rhea has been used previously to simulate global mantle convection to 1 km resolution, satisfactorily recovering the motion of plates and microplates. In this document we detail the choices made for the computational algorithms and numerical solvers, and the technical background for their implementation, and we discuss their performance and accuracy using problems with exact solutions, as well as community benchmarks.

In all cases, our focus was on maximal algorithmic efficiency, which is reflected in the following considerations.

We cover the computational domain by what we call a forest of octrees—a collection of conforming mapped hexahedra, each of which is the root of an adaptive octree. This leads to logically cubic elements that feature hanging faces and edges when elements of different sizes meet. The main benefit of this approach is that it allows us to define a space-filling curve that we exploit for fast mesh partitioning and search of element neighbours. In particular, we do not depend on external graph-partitioning software that would introduce additional overhead and complexity.

We choose continuous trilinear finite elements for both the velocity and the pressure. The introduction of an element-wise projection term in the pressure block stabilizes the Stokes system and allows us to handle all variables within the same fast finite element framework. Because this term can introduce artificial compressibility, we are considering different-order velocity-pressure pairings, as well as discontinuous elements for the pressure. However, higher order finite elements for the velocity and discontinuous elements for the pressure require adapted data structures, and complicate the preconditioning of the Stokes operator.

To apply the inverse of the block-diagonal pre-conditioner, we use an AMG solver for the viscous operator and approximate the inverse of the pressure Schur complement with an inverse-viscosity pressure mass matrix. This pre-conditioner is symmetric as is the original Stokes system, and thus allows us to use the MINRES iterative solver that does not need to store a history of previous iterates as opposed to GMRES variants. Block-triangular pre-conditioners are interesting alternatives promising faster convergence at the cost of destroying the symmetry of the system. The viscosity-scaled mass matrix is a reasonable approximation of the Schur complement for smoothly varying viscosity. However, for extreme viscosity gradients, the approximation degrades, and convergence of the iterative solver can become slower.

The $\alpha$-predictor-corrector iteration that we use for time integration is well established in elastodynamics and other finite element applications. Although the early truncation of the iteration yields a rather small residual, it still implies that the method is not implicit and thus limits the time step by a CFL condition. Because we operate in the advection-dominated regime, the quadratic dependence of the diffusion time step on the mesh size does not take effect and the linear dependence because of the advection component prevails. This situation may change at resolutions of roughly 10 m for a global run, which seems far beyond the accuracy of current tectonic models. Still, we may consider treating at least the diffusion term implicitly, or to switch to fully implicit time integrators. Another alternative is to consider an altogether different approach to solving the energy equation, for example the discontinuous Galerkin method. This method would be ideally suited to simulate advected quantities, such as chemical concentrations. Finally, the time step size limit may be considered separately for each element to avoid overresolution in time for large elements. These are common challenges that will generally need to be addressed in future AMR simulations.

Having outlined the design principles of Rhea, we demonstrate its correctness by the method of manufactured solutions, and by solving a series of community benchmark problems both instantaneous
and time-dependent. We argue that adaptivity has the potential to increase accuracy and reduce the computation time for high-Rayleigh number simulations such as presented by earth’s geodynamics. We demonstrate the parallel efficiency of Rhea by scaling a variable-viscosity Stokes solve to 122 880 cores of the Jaguar supercomputer. Our results indicate that Rhea is indeed an accurate and scalable code for simulating global mantle convection and possibly other thermal convection scenarios.

ACKNOWLEDGMENTS

The authors would like to thank Shijie Zhong for discussion and feedback. The NSF PetraApps program (OCI-0749334, OCI-0748898), the NSF CDI program (CMMI-1028889, CMMI-1028978), TerraGrid allocation (TG-MCA04N026) and further grants (EAR-0426271, EAR-0810303, DMS-072474) are gratefully acknowledged, as well as funding by the DOE Office of Science (DE-FC02-06ER25782, DE-SC0002710) and support by the Caltech Tectonics Observatory (by the Gordon and Betty Moore Foundation). The Texas Advanced Computing Center (TACC) and Oak Ridge National Laboratories provided outstanding help and support for our use of the Ranger and Jaguar supercomputers, respectively.

REFERENCES


