TerraFERMA: the Transparent Finite Element Rapid Model Assembler for multi-physics problems in Earth sciences

Cian R. Wilson,[1] Marc Spiegelman[1] and Peter E. van Keken[2]

[1]Lamont-Doherty Earth Observatory,

Columbia University, Palisades, NY 10964,

USA.

[2]Department of Earth and Environmental

Sciences, 1100 North University Avenue,

2534 CC Little Building, University of

Michigan, Ann Arbor, Michigan 48109-1005,

USA.

Abstract.

We introduce and describe a new software infrastructure TerraFERMA, the *Transparent Finite Element Rapid Model Assembler* for the rapid and reproducible description and solution of coupled multi-physics problems. The design of TerraFERMA is driven by two overarching computational needs in Earth sciences. The first is the need for increased flexibility in both problem description and solution strategies for coupled problems where small changes in model assumptions can often lead to dramatic changes in physical behavior. The second is the need for software and models that are more transparent so that results can be verified, reproduced and modified in a manner such that the best ideas in computation and Earth science can be more easily shared and reused.

TerraFERMA leverages three advanced open-source libraries for scientific computation that provide high level problem description (FEniCS), composable solvers for coupled multi-physics problems (PETSc) and a science neutral options handling system (SPuD) that allows the hierarchical management of all model options. TerraFERMA integrates these libraries into an easier to use interface that organizes the scientific and computational choices required in a model into a single options file, from which a custom compiled application is generated and run. Because all models share the same infrastructure, models become more reusable and reproducible. TerraFERMA inherits much of its functionality from the underlying libraries. It currently solves partial differential equations (PDE) using finite element methods on meshes

of intervals (1D), triangles (2D) and tetrahedra (3D). The software is particularly well suited for non-linear problems with complex coupling between components. TerraFERMA is open-source and available as a git repository at bitbucket.org/tferma/tferma.

1. Introduction

Computational models have been an essential component of solid Earth geodynamics since the advent of plate tectonics [e.g. *McKenzie*, 1969; *Minear and Toksöz*, 1970; *Hasebe et al.*, 1970; *Torrance and Turcotte*, 1971; *McKenzie et al.*, 1973]. While restricted to limited geometries and parameter spaces, these simplified model problems gave useful insights into thermal convection, and had the benefit of being reasonably understandable and reproducible. With exponential increases in computational hardware and algorithmic advances, significantly more complicated (and potentially more "realistic") models have become more accessible. For example, many recent geodynamic models include complex rheologies, geometries, and the incorporation of additional physics such as compressibility, multi-phase flow, and magma transport [e.g. *Tackley*, 2002; *Katz et al.*, 2007a; *May and Moresi*, 2008; *Stadler et al.*, 2010; *Faccenda et al.*, 2012; *Gerya et al.*, 2013].

With increased complexity comes the potential for reduced transparency and reproducibility of computational models. While community benchmarking exercises [e.g. *Blankenbach et al.*, 1989; *Busse et al.*, 1994; *van Keken et al.*, 1997; *King et al.*, 2010; *Koglin et al.*, 2005; *van Keken et al.*, 2008; *Schmeling et al.*, 2008; *Zhong et al.*, 2008; *King et al.*, 2010], have been useful and remain essential for code comparison and validation, they can be time consuming and relatively unglamorous. And while they often identify the strengths and weaknesses of various computational and algorithmic choices, they do not necessarily facilitate code improvements through sharing of ideas or methods between codes.

We believe that much of this rigidity stems from the software design of most current computational models used in geodynamics. At their most abstract, models are simply a very large set of choices including governing equations, constitutive relations, geometry, tessellation, discretization, solvers, diagnostic output, etc. Traditionally, most of these choices are made early in the development process and hard-wired into the resulting application code. This code design can make it extremely challenging to make modifications to either the solution algorithm or the model physics because changing one aspect of the model often requires considerable recoding of others. Thus the ability to combine the best features of different codes becomes nearly impossible.

This lack of flexibility is particularly problematic for "multi-physics" problems such as thermo-chemical convection, coupled fluid-solid dynamics, reactive flows, and magmadynamics. These problems often demonstrate a sensitivity to assumptions where small changes in the coupling between variables or constitutive relations can lead to dramatic changes in physical behavior. For example, introduction of rheological weakening due to melt or thermal dissipation can potentially lead to strong localization into shear bands, that would not develop for uncoupled rheologies [e.g. *Katz et al.*, 2006; *Braeck and Podladchikov*, 2007; *Braeck et al.*, 2009]. In turn, these changes in physics often affect critical computational choices such as discretizations, solvers and preconditioners, which can make hard-wired solver choices fragile. Because many of these changes due to coupling are unanticipated, and therefore interesting, efficient exploration of such coupled systems requires a computational framework that provides flexibility for both describing multiphysics problems and modifying the solution algorithms.

While an increase in flexibility gives a user more options for designing and exploring complex models, it becomes increasingly imperative that these models track and archive scientific and computational choices in a hierarchical and logical fashion. Published models should not only be reproducible but also allow other users to modify an existing model to explore their own interests while retaining transparency and reproducibility of all models. These features are particularly critical when designing software for use by a broader community (e.g. by the Computational Infrastructure for Geodynamics [CIG, geodynamics.org] and other organizations).

We argue that the adoption and exploration of complex multi-physics models requires two fundamental design changes for computation in the solid Earth sciences. First is an increased flexibility for users in choosing the problem description and type of solvers. Second is the availability of improved interfaces for building, running, archiving and sharing model results. These features are available from existing open-source software libraries that provide common interfaces and a wide range of choices to key aspects of core functionality. For example, PETSc [*Balay et al.*, 1997] or Trilinos [*Heroux et al.*, 2003] provide access to a large number of linear and non-linear algebraic solvers. Software such as deal.ii [*Bangerth et al.*, 2007], FEniCS [*Logg et al.*, 2012] and OpenFOAM (www.openfoam.com) provide assembly routines over discrete domains. Separately, SPuD [*Ham et al.*, 2009] provides an application-neutral hierarchical options system. These libraries are open-source and interoperable by design, allowing a user to leverage the best features for a given application. However, while they provide the key building blocks for any multi-physics code, they do not provide guidance on how to build the application itself or how to manage the large suite of underlying options.

To provide this functionality we have developed the Transparent Finite Element Rapid Model Assembler (TerraFERMA), which is built on the FEniCS, PETSc and SPuD libraries. The main advantage over traditional finite element or finite volume models used in geodynamics is that a single options file fully describes the problem, including all equations, boundary and initial conditions, coefficients, and solver options. Modifying equations, adding physics or changing solver strategies is straightforward in TerraFERMA. Custom-compiled applications are generated from the options file that share an infrastructure for services common to all models, including diagnostics, checkpointing, and global non-linear convergence monitoring. TerraFERMA is open-source (GNU Lesser General Public License, version 3) and developed using best practice software engineering including version control, issue tracking, and buildbots. The latter perform automatic builds and tests on a suite of standard geodynamic benchmarks.

The following section describes the design principles behind TerraFERMA and reviews the key computational libraries that provide most of the underlying functionality. We then illustrate the utility of TerraFERMA with a specific example of thermal convection. This is followed by a discussion of a larger suite of benchmark results and a demonstration of its ability to solve coupled fluid-solid flow in subduction zones. Finally we discuss current limitations and future directions.

2. Design Overview

While specific applications or physics may vary from code to code, the overall structure of most models used in geodynamics is the same. TerraFERMA exploits these similarities to provide a common, reusable infrastructure for continuum models. The end-user is still required to input the science-specific components of the model, but TerraFERMA

organizes and manages these components to build the final application with much of the low-level functionality provided by existing advanced, open-source, computational libraries (specifically FEniCS, PETSc and SPuD).

2.1. Code Structure

The basic structure of most model codes used in geodynamics can be represented by a series of nested loops as illustrated in the flow diagram in Figure 1. Iterations used to converge linear solvers are nested within iterations to converge non-linearities between or within equations, which in turn are nested within a time-step loop. Each loop acts on a different level of the model - a hierarchy we reproduce within TerraFERMA.

At the highest level, following initialization, the time loop (in time-dependent simulations) acts on the entire problem (yellow, Figure 1) and generally includes standard model services such as diagnostic output and checkpointing. Below this, iterations attempt to converge non-linearities that exist within or between the model equations. This happens on two levels. The highest (blue, Figure 1) deals with unknowns, or fields, that are only loosely coupled together. Non-linearities here are dealt with in a 'Picard' style loop, simply updating the non-linear dependencies at every iteration. More tightly coupled fields are grouped together into systems, where the equations can be fully coupled together, for example using one or more non-linear solvers (red and green, Figure 1), e.g. a Newton solver.

The hierarchy of loops may be used to describe a set of nested model levels, as illustrated in Figure 2. At the highest level, the problem contains the model geometry, time-stepping options and diagnostic output as well as information about one or more systems making up the problem. In turn, a system contains both the fields grouped together in it and

information about one or more non-linear solvers, which act on the fields to update their values according to the model equations.

The structure of TerraFERMA mirrors the hierarchy described above. The *buckettools* library, a component of TerraFERMA, contains nested C++ classes that manage the overall problem, the systems, the fields and coefficients, and the non-linear solvers. Buckettools also provides the user with a nested tree of options to control each level of the model (see example, Section 4). Model specific components, like the equations and constitutive relations, are also described in the options tree but compiled into an application specific executable that shares generic model services with other applications through the buckettools library. TerraFERMA is therefore not a single executable. Instead, it is the set of tools that allows the rapid development of models, sharing generic services, through an intuitive, hierarchical user interface.

2.2. Libraries

Three core pieces of functionality in TerraFERMA are provided by external libraries: 1) options management (SPuD), 2) discretized equation assembly (FEniCS) and 3) algebraic solvers (PETSc). Here we discuss each of these components individually before demonstrating their use through TerraFERMA in the next section.

2.2.1. SPuD

The options that can be set in the framework include the actual differential equations to be solved, boundary and initial conditions, solver options etc. To manage these many options we use SPuD [*Ham et al.*, 2009] which is an application-neutral options system that provides both human and machine-readable interfaces based on a single xml (extensible markup language) schema. The schema describes all the available options and their

dependent sub-options and is used to automatically populate a graphical interface. This interface (Diamond) displays the schema in a natural tree-like structure and saves user selections to a problem specific options file. The schema may also be used to validate and automatically update old options files, which in a rapid development environment, helps ensure that previous model input files remain up to date.

2.2.2. FEniCS

The assembly of the discrete equations is performed at a high and symbolic level using FEniCS [*Logg et al.*, 2012], which consists of a group of computational libraries that together allow arbitrary multi-physics problems to be described, assembled and solved using finite elements.

FEniCS provides the Unified Form Language [UFL, *Logg et al.*, 2012, Chapter 12], which is a high level language for describing the weak forms of arbitrarily coupled systems of equations. As an example, consider the Poisson equation

$$-\nabla^2 u = f \quad \text{in } \Omega \tag{1}$$

$$u = u_0 \quad \text{on } \partial\Omega$$

with a prescribed function, $f(\boldsymbol{x})$, and solution, $u(\boldsymbol{x})$, defined on a 2D domain, Ω, with boundary conditions on $\partial\Omega$. To describe this problem in UFL we first multiply (1) by a test function u_t and integrate by parts to transform it into its weak form and restrict u to a suitable discrete function space

$$\int_\Omega \nabla u_t \cdot \nabla u_a d\boldsymbol{x} = \int_\Omega u_t f d\boldsymbol{x}. \tag{2}$$

Here u_t is the test function and u_a is the trial function. In our example we restrict these to being piecewise linear Lagrange functions on a mesh of triangles over Ω. It is convenient

to express linear weak forms like (2) in a unified notation

$$a(u_t, u_a) = L(u_t) \tag{3}$$

where, in the current example

$$a(u_t, u_a) = \int_\Omega \nabla u_t \cdot \nabla u_a d\boldsymbol{x} \tag{4}$$

$$L(u_t) = \int_\Omega u_t f d\boldsymbol{x} \tag{5}$$

are the bilinear and linear forms respectively. The problem may now be described succinctly in UFL as

```
u_e = FiniteElement("Lagrange", triangle, 1)


u_t = TestFunction(u_e)
u_a = TrialFunction(u_e)
f   = Coefficient(u_e)


a   = inner(grad(u_t), grad(u_a))*dx
L   = u_t*f*dx
```

A more complete discussion of this problem and how one sets this up with FEniCS using the python interface is provided in *Alnæs et al.* [2012].

Given a description of the problem in UFL, the FEniCS Form Compiler [FFC, *Kirby and Logg*, 2006; *Ølgaard and Wells*, 2010; *Logg et al.*, 2012, Chapter 11] automatically generates C++ source code from UFL. This code describes how to assemble the discretized problem over a single cell. At this stage, however, the problem as described in UFL lacks any details about the shape or desired discretization of the 2D domain, the boundary conditions or the actual expression of f.

The C++ library DOLFIN [*Logg and Wells*, 2010; *Logg et al.*, 2012] (together with its python bindings) provides the link between the FFC generated code [UFC, *Alnæs et al.*, 2009; *Logg et al.*, 2012, Chapter 16] and the rest of the problem description. In particular, DOLFIN provides classes for describing discrete functions, meshes, and interfaces to assembly routines that produce discrete equations given a description of the geometry and the boundary conditions.

2.2.3. PETSc

The third stage is the solution of the discrete equations. Once a matrix-vector system has been assembled, PETSc [*Balay et al.*, 1997, 2001a, b; *Katz et al.*, 2007b] provides a wide range of scalable linear and non-linear solvers with which to solve the discrete equations. All solvers and preconditioners can be chosen and adjusted at run time, rather than being hard-wired into the code from the beginning. This is important as it provides maximum flexibility to the user and allows them to experiment with choices of solvers and preconditioners that might require significant time to code in applications where these solvers are explicitly provided by the user. Of particular interest is that PETSc provides the ability to compose effective multi-physics preconditioners by applying solvers as preconditioners to sub-components of the system [*Brown et al.*, 2012]. These fieldsplit preconditioners allow complex coupled matrices to be solved using the experiences gained from solving smaller single-physics problems.

3. An Example

To make the organization of TerraFERMA less abstract we will work through the example of infinite Prandtl number thermal convection. Detailed instructions and step-by-step

tutorials for this and other problems are provided with the software. Here we primarily lay out the logic and organizational structure of this problem.

The simplest description of 2D thermal convection of an incompressible Boussinesq fluid can be written in non-dimensional form as

$$-\nabla \cdot \eta \left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T\right) + \nabla p - T\boldsymbol{k} = \boldsymbol{0} \tag{6}$$

$$\nabla \cdot \boldsymbol{v} = 0 \tag{7}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{v} \cdot \nabla T - \frac{1}{\mathrm{Ra}} \nabla^2 T = 0 \tag{8}$$

on the domain $\Omega = [0,1] \times [0,1]$, with boundary conditions

$$T(x,0) = 1 \quad T(x,1) = 0 \tag{9}$$

$$\frac{\partial T}{\partial x}(0,z) = 0 \quad \frac{\partial T}{\partial x}(1,z) = 0 \tag{10}$$

$$\boldsymbol{v} \cdot \boldsymbol{n} = \nabla \boldsymbol{v} \cdot \boldsymbol{n} = 0 \quad \text{on } \partial\Omega \tag{11}$$

$\boldsymbol{v}$, P are the dimensionless velocity and pressure, T is dimensionless temperature, and η is dimensionless viscosity which can be constant ($\eta = 1$) or a function of T, $\boldsymbol{v}$ and/or p.

$$\mathrm{Ra} = \frac{\rho g \alpha \Delta T h^3}{\eta_0 \kappa}$$

is the Rayleigh number for a layer of depth h, temperature difference ΔT and reference viscosity η_0. Note, (6)–(8) scale velocity by $\boldsymbol{v} = v_0 \boldsymbol{v}'$ with

$$v_0 = \frac{\rho g \alpha \Delta T h^2}{\eta_0} \tag{12}$$

which is the rise time of a Stokes blob of size h, rather than the more standard "diffusion velocity" $v_0 = \kappa/h$. This scaling puts $1/\mathrm{Ra}$ in the diffusion term in (8), rather than multiplying T in (6). For large Ra, (8) becomes close to pure advection. We have found that this scaling is numerically more robust when solving (6)–(8) as a coupled system.

Thermal convection is a good example of a multi-physics problem as it describes the coupled interactions of three fields; velocity, pressure and temperature. (6)–(7) are the Stokes equations for viscous flow and are linear in $\boldsymbol{v}$ and p if T is known and η is independent of velocity. Likewise, (8) is an advection-diffusion equation for T and appears linear in T if $\boldsymbol{v}$ is known. When all three fields are unknown then the coupled system $\boldsymbol{u} = (\boldsymbol{v}, p, T)$ is non-linear because the velocity is a function of temperature and vice versa. Adding additional coupling through more complex rheologies, $\eta\,(T, \boldsymbol{v}, p)$, only increases the non-linearity of the system. We consider here two approaches: Newton's method (N) and a Picard splitting scheme (P).

To solve for the full non-linear system using (N), Newton's method, we need to compute a residual for the full system $r(\boldsymbol{u})$ as well as an exact or approximate Jacobian $J(\boldsymbol{u})$ and solve

$$J(\boldsymbol{u}_i)\boldsymbol{\delta u}_i = -r(\boldsymbol{u}_i)$$

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \delta \boldsymbol{u}_i \tag{13}$$

until $||r|| <$ tol for some norm on r and measure of tolerance. $\boldsymbol{u}_i$ is our estimate of the solution at iteration i and $\boldsymbol{\delta u}_i$ is the correction that takes us to an improved estimate at iteration $i + 1$.

Computing both the residual and Jacobian is straightforward in finite elements using the high level language features of FEniCS. We start by discretizing the time derivative using finite differences as $\frac{\partial T}{\partial t} = (T_i - T_n)/\Delta t$ where T_i is our current estimate of the value of temperature at the new time and T_n is temperature at the previous time. Using a θ scheme for time-integration of the advection-diffusion equation, multiplying by appropriate test functions $\boldsymbol{u}_t = (\boldsymbol{v}_t, p_t, T_t)$ and integrating by parts, we can write the weak form of the

residual as

$$\begin{aligned} r_{\boldsymbol{v}} &= \int_\Omega \left[\dot{\epsilon}(\boldsymbol{v}_t) : 2\eta\dot{\epsilon}(\boldsymbol{v}_i) - p_i \nabla\cdot\boldsymbol{v}_t - (\boldsymbol{v}_t)_z T_i\right] d\boldsymbol{x} \\ r_p &= \int_\Omega p_t \nabla\cdot\boldsymbol{v}_i d\boldsymbol{x} \\ r_T &= \int_\Omega \left[T_t\left(T_i - T_n + \Delta t \boldsymbol{v}_\theta\cdot\nabla T_\theta\right) + \frac{\Delta t}{\mathrm{Ra}}\nabla T_t\cdot\nabla T_\theta\right] d\boldsymbol{x} \\ r &= r_{\boldsymbol{v}} + r_p + r_T \end{aligned} \tag{14}$$

where

$$\dot{\epsilon}(\boldsymbol{v}) = \frac{1}{2}\left(\nabla\boldsymbol{v} + \nabla\boldsymbol{v}^T\right)$$

is the strain-rate tensor and

$$T_\theta = (1-\theta)T_n + \theta T_i$$

$$\boldsymbol{v}_\theta = (1-\theta)\boldsymbol{v}_n + \theta\boldsymbol{v}_i$$

are $\theta-$weighted variables such that $\theta = 0$ is a fully explicit first order Euler scheme, $\theta = 1$ is a fully implicit backwards Euler scheme and $\theta = 1/2$ is a second-order, Crank-Nicolson (trapezoidal) scheme. Here we use $\theta = 1/2$. For a specific choice of a mixed finite element space for $\boldsymbol{u}_i$ and $\boldsymbol{u}_t$, (14) assembles into a vector which is the discrete residual of the full system. The Jacobian is the functional derivative of the residual with respect to the unknown function $\boldsymbol{u}_i$

$$J(\boldsymbol{u}_i) = \lim_{\epsilon\to 0}\frac{r(\boldsymbol{u}_i + \epsilon\boldsymbol{\delta u}_i) - r(\boldsymbol{u}_i)}{\epsilon} \tag{15}$$

which assembles into the discrete Jacobian matrix.

For many problems, calculating the exact Jacobian can be tedious and error prone. UFL provides useful automatic differentiation routines for calculating Jacobians symbolically before discretization. A partial UFL description of the above problem is

```
T_theta = theta*T_i + (1.-theta)*T_n

v_theta = theta*v_i + (1.-theta)*v_n


r_v = (inner(sym(grad(v_t)),2.*eta*sym(grad(v_i))) \
    - p_i*div(v_t) - T_i*v_t[1])*dx
r_p = p_t*div(v_i)*dx
r_T = (T_t*(T_i - T_n \
    + dt*inner(v_theta, grad(T_theta))) \
    + dt/Ra*inner(grad(T_t), grad(T_theta)))*dx


r =  r_v + r_p + r_T
J = derivative(r, u_i)
```

The subscripts `_t`, `_i` and `_n` are used consistently in TerraFERMA to describe UFL symbols for the test, new and old time-steps of a function.

Within the context of Newton's method, all of the above problems reduce to the same discrete structure that requires solving large, sparse block-Jacobian matrices. The detailed structure of the matrices and potential choice of solvers for them depend on the degree of coupling.

For the simplest isoviscous problem ($\eta = 1$) the inner linear solve in a Newton solver ($J(\boldsymbol{u}_i)\,\boldsymbol{\delta u}_i = -r(\boldsymbol{u}_i)$) looks like

$$\begin{bmatrix} K & G & M \\ G^T & 0 & 0 \\ 0 & B & A \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta v} \\ \delta p \\ \delta T \end{bmatrix} = - \begin{bmatrix} r_{\boldsymbol{v}} \\ r_p \\ r_T \end{bmatrix} \tag{16}$$

where the upper 2×2 block

$$\begin{bmatrix} K & G \\ G^T & 0 \end{bmatrix}$$

is the discrete form of the Stokes equation, M is a (mixed) mass matrix for buoyancy, $B(T_i)$ contains the coupling between velocity and temperature and $A(\boldsymbol{v}_i)$ is the standard advection-diffusion operator.

In TerraFERMA, PETSc gives us the functionality to solve (16) in a number of ways. For small problems (like this example) a direct solver may be applied to the full system at each Newton iteration (N,i). Alternatively PETSC's *fieldsplit* algorithm allows us to solve individual components of the system sequentially. For example, using the upper-triangular blocks of J; first applying the A block before solving the Stokes sub-system (optionally updating the contributions from M in between). This is equivalent to back-substituting using the upper-triangular approximate Jacobian

$$P = \begin{bmatrix} K & G & M \\ G^T & 0 & 0 \\ 0 & 0 & A \end{bmatrix}. \tag{17}$$

which may either precondition (16) (N,ii) or replace J altogether (N,iii).

In (P), a Picard splitting scheme, (16) is split into two systems by first solving

$$A(\boldsymbol{v}_i)\delta T_i = -r_T(\boldsymbol{u}_i) \tag{18}$$

for the temperature correction, using the previous value for $\boldsymbol{v}_i$, then updating the velocity field by solving the linear Stokes system

$$\begin{bmatrix} K & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta v}_i \\ \delta p_i \end{bmatrix} = - \begin{bmatrix} r_{\boldsymbol{v}}(\boldsymbol{u}_{i+1/2}) \\ r_p(\boldsymbol{u}_{i+1/2}) \end{bmatrix} \tag{19}$$

where $\boldsymbol{u}_{i+1/2} = (\boldsymbol{v}_i, p_i, T_{i+1})$ indicates the the temperature has been updated during the iteration.

Figure 3a shows steady-state, isoviscous solutions for the temperature and velocity fields and the mesh used to discretize the problem. Figure 4 compares the convergence behavior of the various solver strategies discussed above; (N,i) direct Newton scheme, applying a direct solver to the full coupled system at each iteration, (N,ii) iterative Newton scheme applying P as an upper-triangular fieldsplit preconditioner (using direct solves sequentially on each sub-system and a flexible gmres iterative solver to a relative tolerance of 10^{-6}

on J), (N,iii) approximate Newton scheme replacing J with P and applying direct solves sequentially on each sub-system, and (P) Picard scheme splitting the coupled system into separate systems and solving each sequentially, updating inter-dependencies between solves within an iteration.

In this isoviscous problem the approximate Newton scheme is equivalent to the Picard splitting because

$$r_{\boldsymbol{v}}(\boldsymbol{u}_{i+1/2}) = r_{\boldsymbol{v}}(\boldsymbol{u}_i) + M\delta T_i. \tag{20}$$

Both schemes demonstrate linear convergence such that the L2 norm of the full non-linear residual is reduced by 6 orders magnitude (to $||r||_2 < 10^{-12}$) in $4-5$ iterations. This can be compared to a direct Newton scheme, which converges quadratically to $||r||_2 \sim 10^{-14}$ in 2 iterations. While the convergence behavior is better with direct Newton, the computational cost is similar to the Picard scheme because of the expense of the direct solves on the full Jacobian (Figure 4b). However, if we use an iterative Newton scheme we gain nearly quadratic convergence with 1-2 Newton iterations, while the computational work per cycle is reduced.

The example above demonstrates the flexibility of TerraFERMA to compare different solver strategies rapidly. This ability becomes even more important for strongly coupled non-linear problems. For example, Figure 3b shows the solution with a more realistic temperature and strain-rate dependent composite rheology

$$\eta(T,V) = \left[\frac{1}{\eta_{\text{diff}}} + \frac{1}{\eta_{\text{disl}}} + \frac{1}{\eta_{\max}}\right]^{-1} \tag{21}$$

where

$$\eta_{diff} = A_{\text{diff}} \exp\left[\frac{E_{diff}}{RT}\right] \tag{22}$$

$$\eta_{disl} = A_{\text{disl}} \exp\left[\frac{E_{disl}}{nRT}\right] \dot{\tilde{\epsilon}}_{II}^{1/n-1} \tag{23}$$

$$\eta_{max} = 10^{24}\,\text{Pa s} \tag{24}$$

are appropriate (dimensional) rheologies for diffusion creep, dislocation creep and a viscosity cap [for example see *Billen and Hirth*, 2007]. In this case the full problem is considerably more non-linear. In particular, the Stokes sub-problem is now fully non-linear due to the dependence of viscosity on the strain-rate tensor through

$$\dot{\tilde{\epsilon}}_{II} = \sqrt{\frac{\dot{\epsilon}(\boldsymbol{v}) : \dot{\epsilon}(\boldsymbol{v}) + \dot{\epsilon}_{\varepsilon}}{2}} \tag{25}$$

where we have regularized the second invariant (of the dimensionless velocities) by $\dot{\epsilon}_{\varepsilon} = 2 \times 10^{-10}$ to prevent singularities at low strain-rates.

The Jacobian matrix, which may again be calculated by automatic differentiation, is still block structured:

$$J = \begin{bmatrix} [K + K_{\boldsymbol{v}}] & G & [M + K_T] \\ G^T & 0 & 0 \\ 0 & B & A \end{bmatrix} \tag{26}$$

but with additional coupling blocks, $K_{\boldsymbol{v}}(\boldsymbol{u}_i)$ and $K_T(\boldsymbol{u}_i)$ due to the variation of viscosity with velocity and temperature respectively, which also make $K(\boldsymbol{u}_i)$ non-linear.

Figure 5 shows convergence and performance results for the same solver strategies discussed for the isoviscous case; (N,i) direct Newton, (N,ii) iterative Newton, (N,iii) approximate Newton, and (P) Picard splitting.

Unlike the isoviscous case, the approximate Newton scheme and the Picard splitting are no longer equivalent with the first successively solving

$$A(\boldsymbol{v}_i)\delta T_i = -r_T(\boldsymbol{u}_i) \quad \text{then} \quad \begin{bmatrix} [K(\boldsymbol{u}_i) + K_{\boldsymbol{v}}(\boldsymbol{u}_i)] & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta v}_i \\ \delta p_i \end{bmatrix} = - \begin{bmatrix} r_{\boldsymbol{v}}(\boldsymbol{u}_i) + [M + K_T(\boldsymbol{u}_i)]\,\delta T_i \\ r_p(\boldsymbol{u}_i) \end{bmatrix} \tag{27}$$

and the second

$$A(\boldsymbol{v}_i)\delta T_i = -r_T(\boldsymbol{u}_i) \quad \text{then} \quad \begin{bmatrix} K(\boldsymbol{u}_{i+1/2}) & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta v}_i \\ \delta p_i \end{bmatrix} = - \begin{bmatrix} r_{\boldsymbol{v}}(\boldsymbol{u}_{i+1/2}) \\ r_p(\boldsymbol{u}_{i+1/2}) \end{bmatrix}. \tag{28}$$

Due to the non-linearity of $K(\boldsymbol{u}_i)$ and $K_{\boldsymbol{v}}(\boldsymbol{u}_i)$, $r_{\boldsymbol{v}}(\boldsymbol{u}_{i+1/2})$ is no longer equivalent to $r_{\boldsymbol{v}}(\boldsymbol{u}_i) + [M + K_T(\boldsymbol{u}_i)]\,\delta T_i$. Despite this, their convergence behaviors are similar with only marginal gains using the approximate Newton scheme. As before the full benefits of Newton's method are only seen using the direct and iterative Newton schemes, where the residual norm drops 10 orders of magnitude in 3 as opposed to 6 iterations. These higher convergence rates however come at a higher cost than using Picard splitting (see Figure 5a), owing to the increased computational expense of assembling and solving (26) as opposed to the linearized matrices in (28).

The above discussion is not intended to be an exhaustive study of solver strategies for thermal convection, and many more alternatives are possible. For example, Figure 5 also shows results for an iterative Newton scheme that uses an approximate Jacobian (N,iv) that maintains the non-linearity in temperature but doesn't assemble the $K_{\boldsymbol{v}}$ block in Eq. (26). This scheme is comparable in convergence and performance to Picard. It should be noted that, for this particular problem, most of the solution is in the diffusion creep regime, so neglecting the some of the non-linearity due to strain-rate may be an effective strategy. However, as these examples demonstrate, optimal solver strategies can

be problem dependent and the flexibility inherent in TerraFERMA can greatly facilitate exploring and implementing them.

In TerraFERMA, FEniCS allowed the changes to the rheological law and the approximate Jacobian to be implemented by changing approximately five lines of UFL while PETSc gave us a wide range of possible solver combinations to explore. Moreover, when optimal strategies are developed for specific problems, these options are recorded and managed in a single options file that can be verified and used as a starting point for other problems. Example options files for the problems shown in Figures 4 and 5 are provided with the software.

4. Managing options with TerraFERMA

TerraFERMA presents an options system that mirrors the overall structure of computational models and provides a hierarchical tree structure for recording both scientific and computational options that is consistent and reusable from model to model. To manage options we use the functionality of the SPuD library [*Ham et al.*, 2009], which allows a developer to write an easily readable rules file (a schema) that defines all the options and dependencies needed in a model. Diamond automatically generates a graphical user interface from this schema which guides the user in choosing all the required (and optional) options. The options file can be interrogated by an application to implement or manipulate options. SPuD is quite general and has been used, for example, as an options front-end to the CFD code Fluidity [*Davies et al.*, 2011].

Our schema describes a `tfml` file (TerraFERMA markup language) which includes options that are used to compile application specific models (particularly science specific options such as weak forms, constitutive relationships, and boundary conditions) as well

as run-time options, such as the choice of solvers and diagnostics. SPuD and Diamond are also self-documenting, providing a short description of each option to the user. Optional user comments can also be incorporated into the `tfml` file through the Diamond interface.

Figure 6 shows a screenshot of a basic Diamond window with the highest level structure of a `tfml` file (in this case for the isoviscous convection problem shown in Figure 3a). At the highest level, the primary choices are divided into

geometry: Information about dimension and meshes.
io: Information about output files, visualization options, settings for various diagnostics and checkpoint times.
timestepping: Settings for start and finish times, as well as time-stepping and steady-state controls.
global_parameters: global UFL code available to all systems (below).
system: Description of coupled fields, non-linear solvers to be applied to them and related coefficients. There can be multiple systems. The fields in each system can be updated at the start of the simulation, every time step, or during diagnostic output.

Figure 6 includes two systems, one (Convection) for the solution of $\boldsymbol{u} = (\boldsymbol{v}, P, T)$ at every time step and one diagnostic (CourantNumber). The overall schema has a tree structure with relevant sub-options appearing under each heading. Diamond provides visual cues to identify which sub-options are required to completely populate the schema. A complete step-by-step tutorial for solving Poisson's equation with screen dumps is provided in the software tutorials.

It is important to note that every new option added to the schema becomes part of the infrastructure which means that an improvement driven by one model is automatically available for all subsequent models. SPuD provides a mechanism for validating `tfml` files and making sure they are consistent with the current schema.

The description of the differential equations and their discretization is provided within the **system**. Figure 7 shows Diamond with the **system (Convection)** tab unfolded. As described above, each system includes a set of **fields**, identified by name (here Velocity, Pressure and Temperature) which are the coupled variables to be solved for; **coefficients** are constants or known functions that can be evaluated in space and time and **non-linear solvers** that include both the equations and solver choices.

Once we choose a **field**, we need to populate a set of choices to fully characterize it. These include

ufl_symbol: A unique symbol (e.g. T for Temperature) that can be used in any UFL expression throughout the model.

rank: Scalar, Vector or Tensor valued fields.

element: The finite element family for the field (e.g. P1, P2, P2DG).

initial_condition: Initial conditions can be specified as constants, a python or C++ function, or a file (for checkpointing purposes). Initial conditions can be set over the whole mesh or in sub-domains.

boundary_condition: Methods, codes and boundary IDs for setting strong Dirichlet boundary conditions.

diagnostics: Switches for determining what output or monitoring should be performed for a given field, including UFL functionals that should be evaluated.

Given a set of fields and coefficients, the individual **non-linear solvers** provide equations and solver options that operate on the fields to transform initial and boundary conditions into a solution for the coupled fields at the current time. We allow several types of non-linear solver but will concentrate here on PETSc's SNES (Scalable Nonlinear Equations Solvers).

form (Residual): A piece of UFL describing the weak form of the non-linear residual $r(\boldsymbol{u}_i)$ (Figure 7).
form (Jacobian): UFL describing the weak form of the Jacobian (can be specified using automatic differentiation in UFL).
snes_type: the type of PETSc SNES to use.
relative_error: The relative change in $||r||_2$ that terminates the non-linear iterations.
absolute_error: The lower threshold of $||r||_2$ that terminates the non-linear iterations.
max_iterations: The maximum number of non-linear iterations allowed before failing.
monitors: a variety of options that can be optionally turned on to monitor convergence behavior (at additional expense).
linear solvers: A combination of an **iterative_method** and a **preconditioner** to use at each non-linear iteration on the Jacobian.

Figure 8a shows the set up for the direct Newton solve (N,i) with iterative method `preonly` and preconditioner `lu`. Figure 8b shows the same problem but for an iterative Newton solve (N,ii). Here the iterative method is set to `fgmres` (with appropriate convergence and monitoring settings attached) and the preconditioner is set to `fieldsplit`. Within the fieldsplit preconditioner, we define two splits: Temperature and Stokes and assign the appropriate fields to each split. The **composite type** "multiplicative" tells SNES to apply the individual fieldsplit pre-conditioners sequentially and use the updated corrections as soon as they are available. The order of operations is set by the order of the fieldsplits. Here we solve T then $(\boldsymbol{v}, p)$ as discussed in Section 3. Each fieldsplit has a further linear solver and iterative method and preconditioner associated with it, which could in turn have nested fieldsplits. The tree structure of the SPuD schema is particularly useful for describing and composing complicated nested fieldsplit preconditioners.

For example, Figure 9 describes a three layer nested fieldsplit for the same problem that uses a block-preconditioned iterative scheme for the Stokes block.

It is clear that due to the complexity of a working computational model this discussion can really only highlight the structure of TerraFERMA and the options system. Detailed tutorials and example `tfml` files that can be run immediately are provided with the software. Since TerraFERMA inherits both functionality and nomenclature from FEniCS and PETSc, some familiarity with both libraries will aid in the exploration of TerraFERMA. The use of the GUI greatly facilitates the use of TerraFERMA as the options can be explored without a need to directly remember their names or to explicitly type them in a text file (as is common with most standalone codes used in the community).

Once a `tfml` file is fully populated, a custom application code is compiled and run from the options file. We provide a utility function `tfbuild` that uses CMake to construct a build directory from which `make` can be called to compile or run the application. During run time, if requested, diagnostic statistics and convergence files will be written that contain basic information on the state of the run. These can be viewed at any time with the utility command `tfplot`. Figure 10 shows an example that plots a derived quantity (the Nusselt number - a functional of T) as a function of time.

5. Benchmarks

To demonstrate the flexibility and accuracy of TerraFERMA we have implemented a large number of published geodynamics benchmarks as `tfml` files, which provide an extensive testing suite. The files and the scripts to run them are available as a separate git repository at bitbucket.org/tferma/benchmarks. The benchmark suite currently includes over 50 individual test problems from six different publications spanning incompressible

and compressible mantle convection [*Blankenbach et al.*, 1989; *King et al.*, 2010; *Kramer et al.*, 2012], comparison between numerical simulations and laboratory experiments of plumes [*Vatteville et al.*, 2009], subduction zones [*van Keken et al.*, 2008] and magma dynamics [*Simpson and Spiegelman*, 2011]. The supplemental material briefly describes the benchmarks and provides tabular data comparing TerraFERMA results to published results for a wide range of metrics. Two specific examples are shown in Figures 11 and 12.

Figure 11 shows a comparison of numerical models to laboratory experiments for a cylindrical thermal plume [*Vatteville et al.*, 2009]. Figure 12 shows 2D solitary wave solutions and error plots from the benchmarks in [*Simpson and Spiegelman*, 2011]. The supplemental material includes both 2D and 3D results. Figure 13 shows a TerraFERMA calculation of the instability of 1-D solitary waves in 3D.

6. Current Limitations and Future Directions

The fundamental design objective of TerraFERMA is to make modeling coupled systems easier and more accessible to a broader user base. In its initial release, TerraFERMA can successfully solve a wide range of problems in solid Earth geodynamics and we are currently using it to explore new problems such as coupled fluid-solid flow in subduction zones (Figure 14), reactive cracking during serpentinization and other problems. Compared to many open-source geodynamics codes, TerraFERMA is considerably more flexible and (we hope) easier for a user to explore problems beyond traditional single phase flow. Nevertheless, while the infrastructure can tackle many problems, it still inherits much of its functionality from the underlying libraries and there are some limitations that are worth discussing.

First, TerraFERMA solves finite element problems in continuum mechanics using intervals, triangles and tetrahedra with a finite element structure that is inherited from FEniCS. This includes a large range of problems and elements, but obviously does not support finite-volume, finite-difference or spectral discretizations. FEniCS itself is a rapidly evolving project and we expect to track and incorporate some new features from FEniCS as they become available and prove useful.

Second, while the functionality available in existing open-source libraries and their basic design interoperability makes a project such as TerraFERMA practical, the mixing and matching of libraries can lead to some awkward design issues. For example, our code allows three mechanisms for defining functions, expressions and their relations. One can define simple expressions in UFL (which is an extension of python), python functions for initial and boundary conditions and C++ expressions for more complicated functions or those that require better performance. Having the ability to extend code in multiple languages, however, leads to the possibility of multiple namespaces where variables defined in one language are not accessible to the others. This can lead to multiple definitions with inconsistent values, which requires careful programming when expanding TerraFERMA.

We plan to address some other limitations in future releases. For example, we currently only support time-stepping schemes that store two time levels. These are adequate for many geodynamics problems, however, given the wide range of advanced ODE solvers available, particularly in PETSc, we would like to explore extending the framework to incorporate better adaptive time-stepping schemes. We also would like to improve the efficiency and performance of large coupled systems that are built on many component mixed finite elements. Incorporating mixed elements allows clear control over coupled

fields, but, in its simplest form, leads to redundant assembly of sub-blocks with corresponding issues of sparsity. We also note that while both FEniCS and PETSc are fundamentally parallel libraries, not all features/elements in FEniCS are supported in parallel yet. We are working to incorporate those that are into TerraFERMA.

We recognize that successful use of TerraFERMA will require significant training and education. Use of TerraFERMA is not entirely trivial due to the high level of abstraction, the use of multiple libraries, and the large variety of choices that need to be made in setting up strongly coupled multi-physics problems. TerraFERMA also requires a solid understanding of computational math, in particular because it borrows its vocabulary from the computational mathematics based libraries of FEniCS and PETSc. While this language may be less familiar to the broader Earth science community, we believe it is the correct way to discuss these models, which are essentially coupled systems of partial differential equations.

We expect that future enhancements in FEniCS, PETSc and further development of TerraFERMA, aided in part by the Computational Infrastructure for Geodynamics (geodynamics.org) will alleviate some of the current limitations. Distribution through CIG will also aid in the training of new users of TerraFERMA.

Finally, we hope that if TerraFERMA is widely adopted, it can provide both a useful collaborative tool and a growing archive of working example models/`tfml` files that encapsulate computational and model choices that work and can be used as starting points for other users. By including `tfml` files as supplements to publications that use TerraFERMA, published models can become more reproducible and extendable.

Acknowledgments. We thank Mark Behn, Ikuko Wada, and the participants of the 4th Computational Infrastructure for Geodynamics workshop on mantle convection and lithosphere dynamics (July 2012, UC Davis, California) for discussions. Supported by the National Science Foundation MARGINS program grants OCE-0841079 and EAR-1141976 (to CW and MS) and OCE-0841075 (to PvK).

References

Alnæs, M. S., A. Logg, K.-A. Mardal, O. Skavhaug, and H. P. Langtangen (2009), Unified Framework for Finite Element Assembly, *International Journal of Computational Science and Engineering*, *4*(4), 231–244.

Alnæs, M. S., J. Hake, R. C. Kirby, H. P. Langtangen, A. Logg, and G. N. Wells (2012), *The FEniCS Manual*.

Balay, S., W. D. Gropp, L. C. McInnes, and B. F. Smith (1997), Efficient management of parallelism in object oriented numerical software libraries, in *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen, pp. 163–202, Birkhauser Press.

Balay, S., W. D. Gropp, L. C. McInnes, and B. F. Smith (2001a), PETSc users manual, *Tech. Rep. ANL-95/11 - Revision 2.1.0*, Argonne National Laboratory.

Balay, S., K. Buschelman, W. D. Gropp, D. Kaushik, L. C. McInnes, and B. F. Smith (2001b), PETSc home page, http://www.mcs.anl.gov/petsc.

Bangerth, W., R. Hartmann, and G. Kanschat (2007), deal.II—A general-purpose object-oriented finite element library, *ACM Transactions on Mathematical Software*, *33*(4), 24–es, doi:10.1145/1268776.1268779.

Billen, M. I., and G. Hirth (2007), Rheologic controls on slab dynamics, *Geochemistry Geophysics Geosystems*, *8*(8), Q08,012, doi:10.1029/2007GC001597.

Blankenbach, B., et al. (1989), A benchmark comparison for mantle convection codes, *Geophys. J. Int.*, *98*, 23–38, doi:10.1111/j.1365-246X.1989.tb05511.x.

Braeck, S., and Y. Podladchikov (2007), Spontaneous thermal runaway as an ultimate failure mechanism of materials, *Physical Review Letters*, *98*(9), doi:10.1103/PhysRevLett.98.095504.

Braeck, S., Y. Podladchikov, and S. Medvedev (2009), Spontaneous dissipation of elastic energy by self-localizing thermal runaway, *Physical Review E*, *80*(4), doi:10.1103/PhysRevE.80.046105.

Brown, J., M. G. Knepley, D. A. May, L. C. McInnes, and B. Smith (2012), Composable linear solvers for multiphysics, in *ispdc, 11th International Symposium on Parallel and Distributed Computing*, pp. 55–62, IEEE.

Busse, F. H., et al. (1994), 3d convection at infinite prandtl number in cartesian geometry - a benchmark comparison, *Geophysical & Astrophysical Fluid Dynamics*, *75*(1), 39–59, doi:10.1080/03091929408203646.

Cagnioncle, A.-M., E. M. Parmentier, and L. T. Elkins-Tanton (2007), Effect of solid flow above a subducting slab on water distribution and melting at convergent plate boundaries, *Journal of Geophysical Research*, *112*, 19 PP., doi:200710.1029/2007JB004934.

Davies, D. R., C. R. Wilson, and S. C. Kramer (2011), Fluidity: A fully unstructured anisotropic adaptive mesh computational modeling framework for geodynamics, *Geochemistry Geophysics Geosystems*, *12*(6), 20 PP., doi:10.1029/2011GC003551.

Faccenda, M., T. Gerya, N. Manckelow, and L. Moresi (2012), Fluid flow during slab unbending and dehydration: implications for intermediate-depth seismicity, slab weakening and deep water recycling, *Geochemistry Geophysics Geosystems*, *13*.

Gerya, T. V., D. A. May, and T. Duretz (2013), An adaptive staggered grid finite difference method for modeling geodynamic stokes flows with strongly variable viscosity, *Geochem. Geophys. Geosys.*, *14*, 1200–1225.

Ham, D. A., et al. (2009), Spud 1.0: generalising and automating the user interfaces of scientific computer models, *Geoscientific Model Development*, *2*(1), 33–42, doi:10.5194/gmd-2-33-2009.

Hasebe, K., N. Fujii, and S. Uyeda (1970), Thermal processes under island arcs, *Tectonophysics*, *10*, 335–355.

Heroux, M., et al. (2003), An Overview of Trilinos, *Tech. Rep. SAND2003-2927*, Sandia National Laboratories.

Katz, R., M. Knepley, B. Smith, M. Spiegelman, and E. Coon (2007a), Numerical simulation of geodynamic processes with the portable extensible toolkit for scientific computing, *Physics of the Earth and Planetary Interiors*, *163*, 52–68.

Katz, R., M. Knepley, B. Smith, M. Spiegelman, and E. Coon (2007b), Numerical simulation of geodynamic processes with the Portable Extensible Toolkit for Scientific Computation, *Physics of the Earth and Planetary Interiors*, *163*(1-4), 52–68, doi:10.1016/j.pepi.2007.04.016.

Katz, R. F., M. Spiegelman, and B. Holtzman (2006), The dynamics of melt and shear localization in partially molten aggregates., *Nature*, *442*(7103), 676–9, doi:10.1038/nature05039.

King, S. D., C. Lee, P. E. van Keken, W. Leng, S. Zhong, E. Tan, N. Tosi, and M. C. Kameyama (2010), A community benchmark for 2-D Cartesian compressible convection in the Earth's mantle, *Geophysical Journal International*, *180*(1), 73–87, doi:{10.1111/j.1365-246X.2009.04413.x}.

Kirby, R. C., and A. Logg (2006), A compiler for variational forms, *ACM Transactions on Mathematical Software*, *32*(3), 417–444, doi:10.1145/1163641.1163644.

Koglin, D., S. Ghias, S. King, G. Jarvis, and J. Lowman (2005), Mantle convection with reversing mobile plates: a benchmark study, *Geochem., Geophys., Geosyst.*, *6*, doi:10.1029/2005GC000924.

Kramer, S. C., C. R. Wilson, and D. R. Davies (2012), An implicit free surface algorithm for geodynamical simulations, *Physics of the Earth and Planetary Interiors*, *194-195*(0), 25 – 37, doi:10.1016/j.pepi.2012.01.001.

Logg, A., and G. N. Wells (2010), Dolfin: Automated Finite Element Computing, *ACM Transactions on Mathematical Software*, *37*(2), 1–28, doi:10.1145/1731022.1731030.

Logg, A., K.-A. Mardal, and G. N. Wells (2012), *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*, *Lecture Notes in Computational Science and Engineering*, vol. 84, Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-642-23099-8.

May, D., and L. Moresi (2008), Preconditioned iterative methods for stokes flow problems arising in computational geodynamics, *Physics of the Earth and Planetary Interiors*, *171*, 33–47.

McKenzie, D. (1969), Speculations on consequences and causes of plate motions, *Geophys. J. Roy. Astron. Soc.*, *18*, 1–18.

McKenzie, D., J. Roberts, and N. Weiss (1973), Numerical models of convection in the earth's mantle, *Tectonophysics*, *19*, 89–103.

Minear, J., and M. Toksöz (1970), Thermal regime of a downgoing slab and new global tectonics, *J. Geophys. Res.*, *75*, 1397–1419.

Ølgaard, K. B., and G. N. Wells (2010), Optimizations for quadrature representations of finite element tensors through automated code generation, *ACM Transactions on Mathematical Software*, *37*(1), 1–23, doi:10.1145/1644001.1644009.

Schmeling, H., et al. (2008), A benchmark comparison of spontaneous subduction modelsTowards a free surface, *Physics of the Earth and Planetary Interiors*, *171*(1-4), 198–223, doi:10.1016/j.pepi.2008.06.028.

Simpson, G., and M. Spiegelman (2011), Solitary wave benchmarks in magma dynamics, *Journal of Scientific Computing*, doi:10.1007/s10915-011-9461-y.

Stadler, G., M. Gurnis, C. Burstedde, L. Wilcox, L. Alisic, and O. Ghattas (2010), The dynamics of plate tectonics and mantle flow: from local to global scales, *Science*, *329*, 1033–1038.

Tackley, P. (2002), Mantle convection and plate tectonics: toward an integrated physical and chemical theory, *Science*, *288*, 2000–2007.

Torrance, and D. Turcotte (1971), Thermal convection with large viscosity variations, *J. Fluid Mech.*, *47*, 113–125.

van Keken, P., S. King, H. Schmeling, U. Christensen, D. Neumeister, and M.-P. Doin (1997), A comparison of methods for the modeling of thermochemical convection, *J. Geophys. Res.*, *102*, 22,477–22,495.

van Keken, P., et al. (2008), A community benchmark for subduction zone modeling, *Phys. Earth Planet. In.*, *171*(1-4, Sp. Iss. SI), 187–197.

van Keken, P. E., B. R. Hacker, E. M. Syracuse, and G. a. Abers (2011), Subduction factory: 4. Depth-dependent flux of H_2O from subducting slabs worldwide, *Journal of Geophysical Research*, *116*(B1), doi:10.1029/2010JB007922.

Vatteville, J., P. E. van Keken, A. Limare, and A. Davaille (2009), Starting laminar plumes: Comparison of laboratory and numerical modeling, *Geochem. Geophys. Geosyst.*, *10*, Q12,013, doi:10.1029/2009GC002739.

Zhong, S., A. McNamara, E. Tan, L. Moresi, and M. Gurnis (2008), A benchmark study on mantle convection in a 3-d spherical shell using citcoms, *Geochem. Geophys. Geosys.*, *9*.

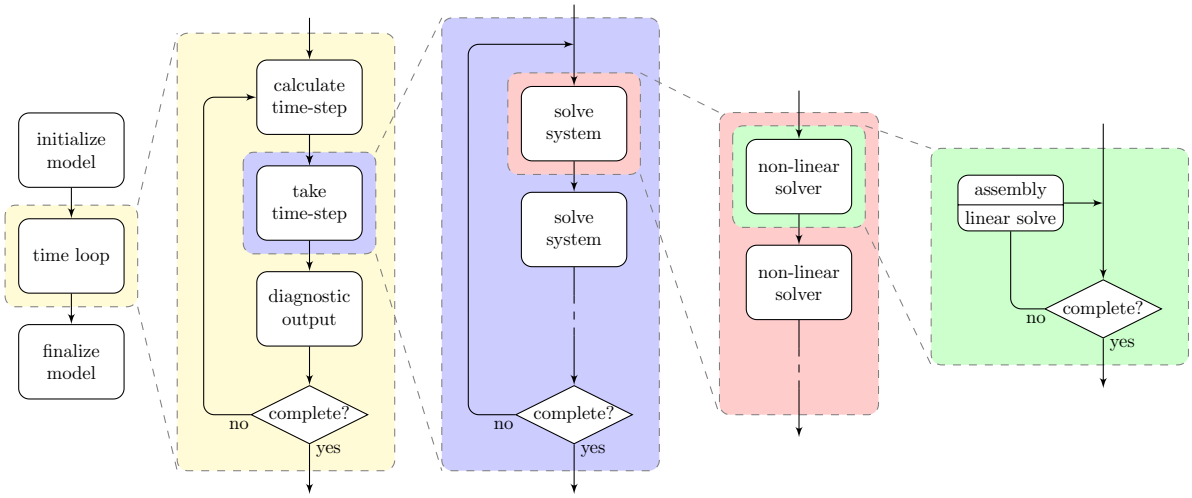

Figure 1. Simplified flow diagram illustrating the key loops of most geodynamic models. The time loop (yellow) is complete when the finish time for a time-dependent simulation is reached or after a single iteration in a steady-state calculation. A time-step itself consists of non-linear iterations over one or more weakly coupled or uncoupled systems of fields (blue). The loop is complete when the global non-linear residual of all systems is minimized to some tolerance. A system consists of a set of strongly coupled fields. A sequence of one or more non-linear solvers (red) may be applied to a system. A non-linear solver loop is generally complete when the system's non-linear residual is minimized to some tolerance.

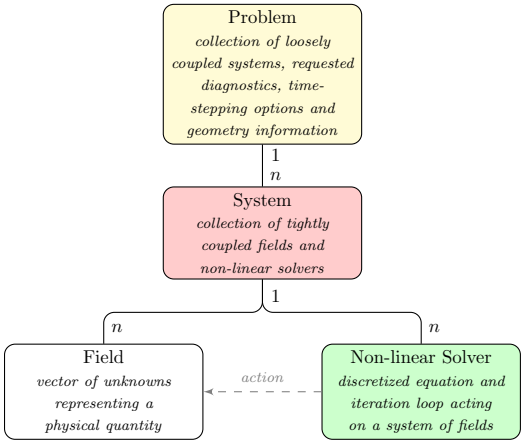

Figure 2. The relationship of various model elements derived from Figure 1. A (single, 1) problem is a collection of (potentially many, n) loosely coupled systems. Systems are collections of (potentially many, n) tightly coupled unknown fields and non-linear solvers which contain the equations that act upon the fields. The underlying C++ class structure of TerraFERMA closely mirrors the model structure outlined here and in Figure 1.

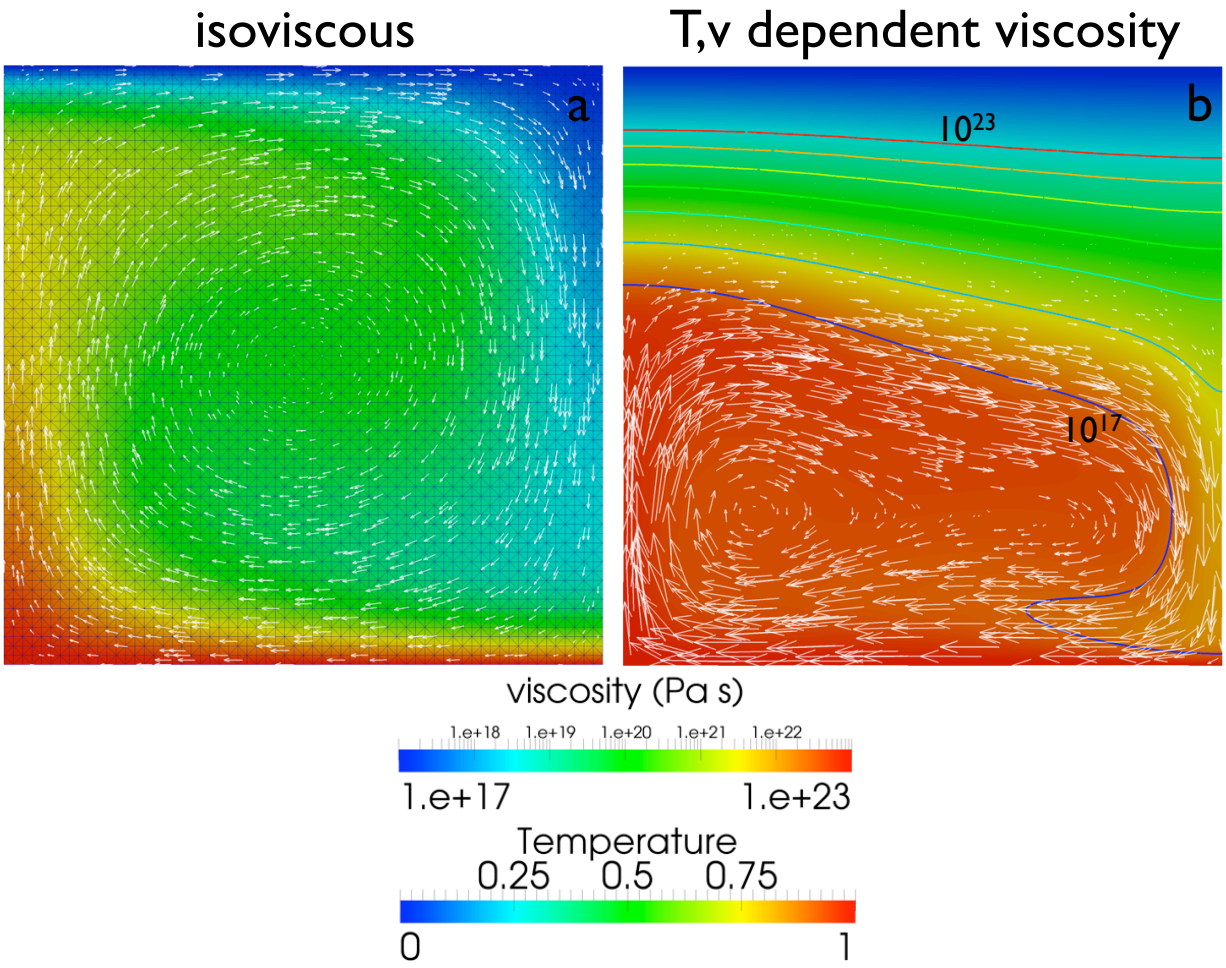

Figure 3. Temperature and velocity fields for two thermal convection simulations. Both solutions are calculated on a mesh of $64 \times 64 \times 2$ regular right triangles (**a**) using a [(P2,P2),P1,P2] mixed element for $\boldsymbol{u} = (\boldsymbol{v}, P, T)$ (i.e. a standard P2-P1 Taylor hood element for Stokes plus P2 quadratic temperature) yielding $54,148$ total degrees of freedom. (**a**) isoviscous convection corresponding to benchmark 1a of *Blankenbach et al.* [1989] at steady-state, showing the mesh. (**b**) The same problem but with a more realistic temperature and strain-rate dependent composite rheology. Contours show log viscosity which varies over ~ 7 orders of magnitude across the domain. Both simulations were run with $\mathrm{Ra} = 10^4$, (**b**) additionally set $A_{\mathrm{diff}} = 1.32043 \times 10^9 \mathrm{Pa\,s}$, $E_{\mathrm{diff}} = 335 \times 10^3 \mathrm{kJ\,mol^{-1}}$, $A_{\mathrm{disl}} = 28968.6 \mathrm{Pa\,s^{1/n}}$, $E_{\mathrm{disl}} = 540 \times 10^3 \mathrm{kJ\,mol^{-1}}$, $n = 3.5$ and $R = 8.3145 \mathrm{J\,mol^{-1}\,K^{-1}}$. A full working description of each problem is contained in the tutorial `tfml` files.

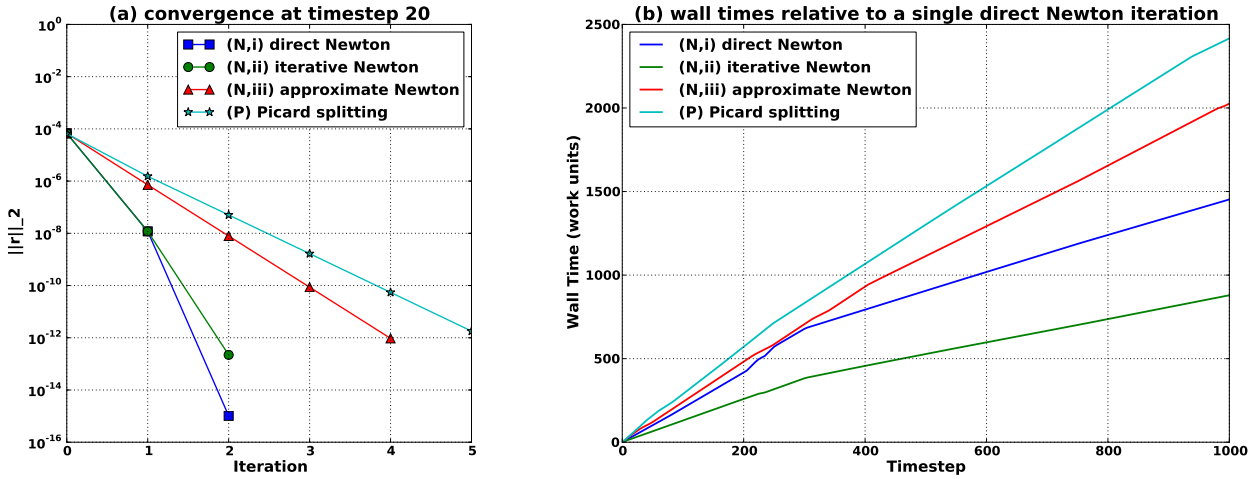

Figure 4. Comparison of convergence behavior and performance for a range of preconditioner-solver options for the isoviscous thermal convection on a unit square (Figure 3a). See Section 3 for a description of the solver strategies. (**a**) convergence behavior of different solver schemes at time-step 20 showing linear convergence for Picard splitting and quadratic for direct and iterative Newton schemes. (**b**) relative performance of different solver schemes. One "work unit" is the time for a single direct Newton iteration which includes assembly of both the residual and Jacobian, and a sparse direct linear solve.

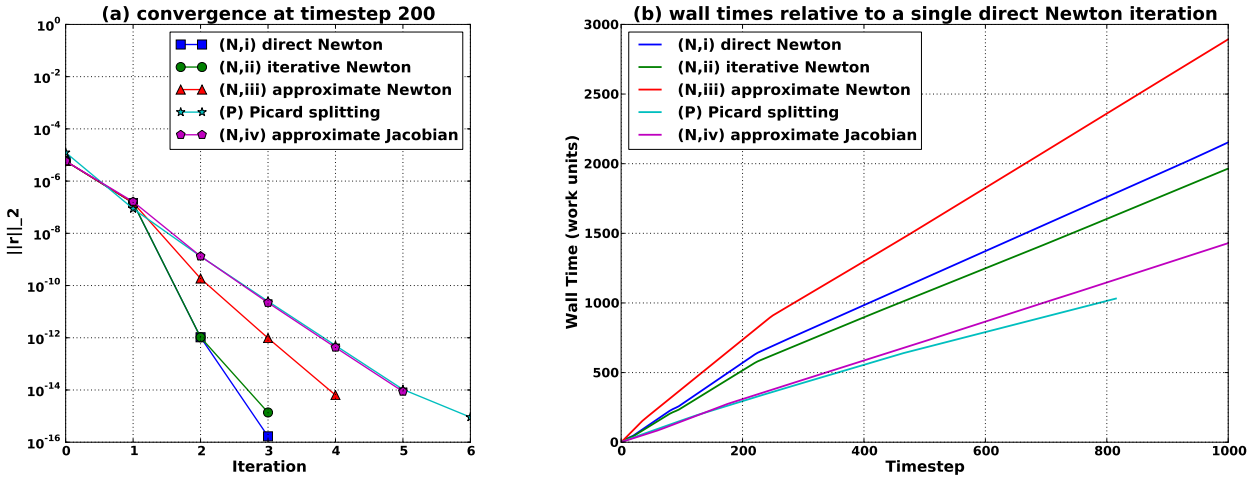

Figure 5. Comparison of convergence behavior and performance for a range of preconditioner-solver options for thermal convection on a unit square with a non-linear rheology (Figure 3b). See Section 3 for a description of the solver strategies. (**a**) convergence behavior of different solver schemes at time-step 200 showing linear convergence for Picard splitting and quadratic for direct and iterative Newton schemes. (**b**) relative performance of different solver schemes. One "work unit" is the time for a single direct Newton iteration which includes assembly of both the residual and Jacobian, and a sparse direct linear solve.

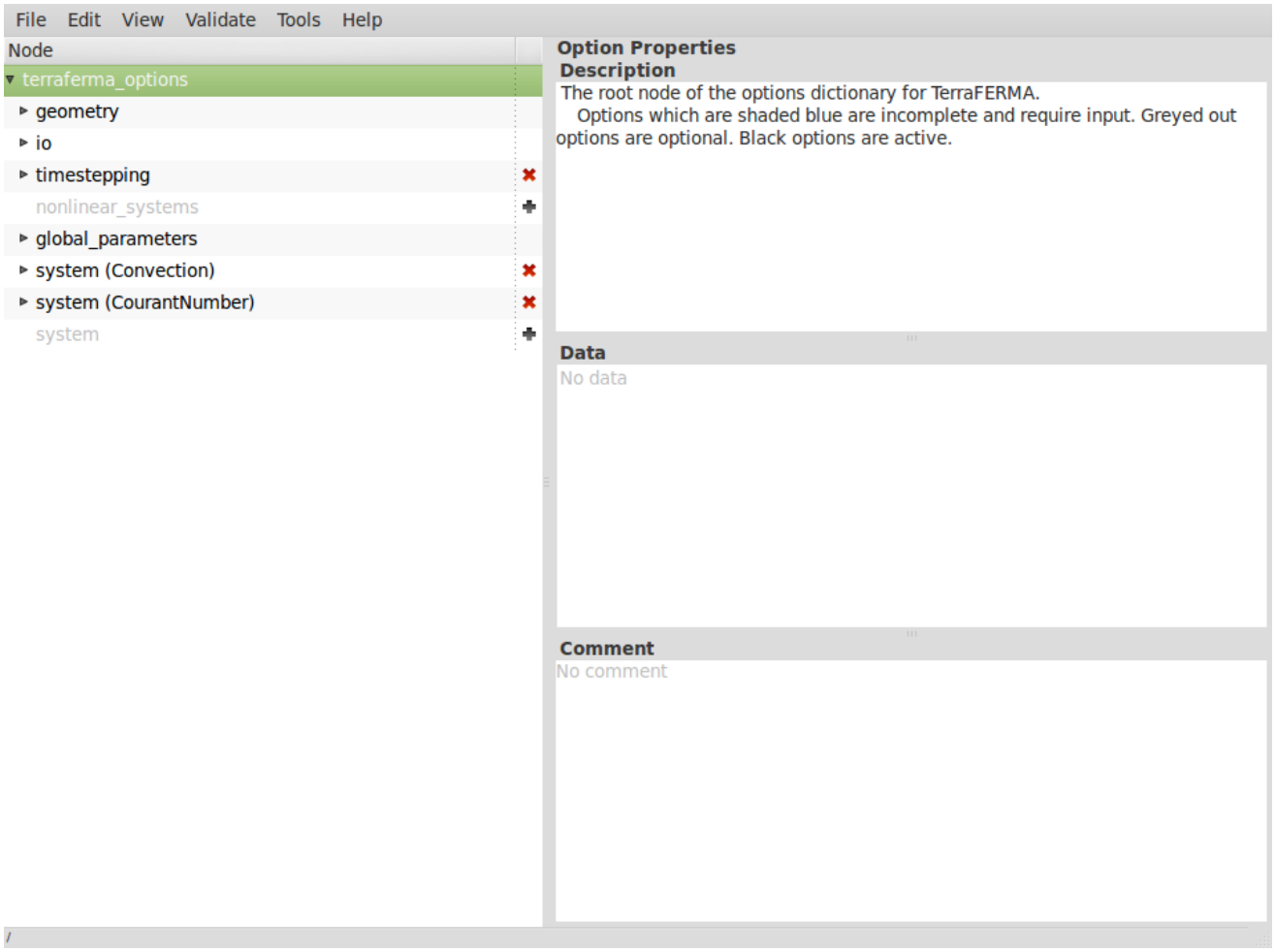

Figure 6. Screenshot of Diamond showing the top-level structure of a `tfml` (TerraFERMA markup language) file demonstrating the major computational choices for controlling meshes (geometry), io, time-stepping and the actual systems of PDEs to be solved. This file shows `convection.tfml` for the isoviscous convection problem and solves two systems, **Convection** for $\boldsymbol{u} = (\boldsymbol{v}, p, T)$ and **CourantNumber** for diagnostic and time-stepping purposes. Note the brief documentation included in the Description window.

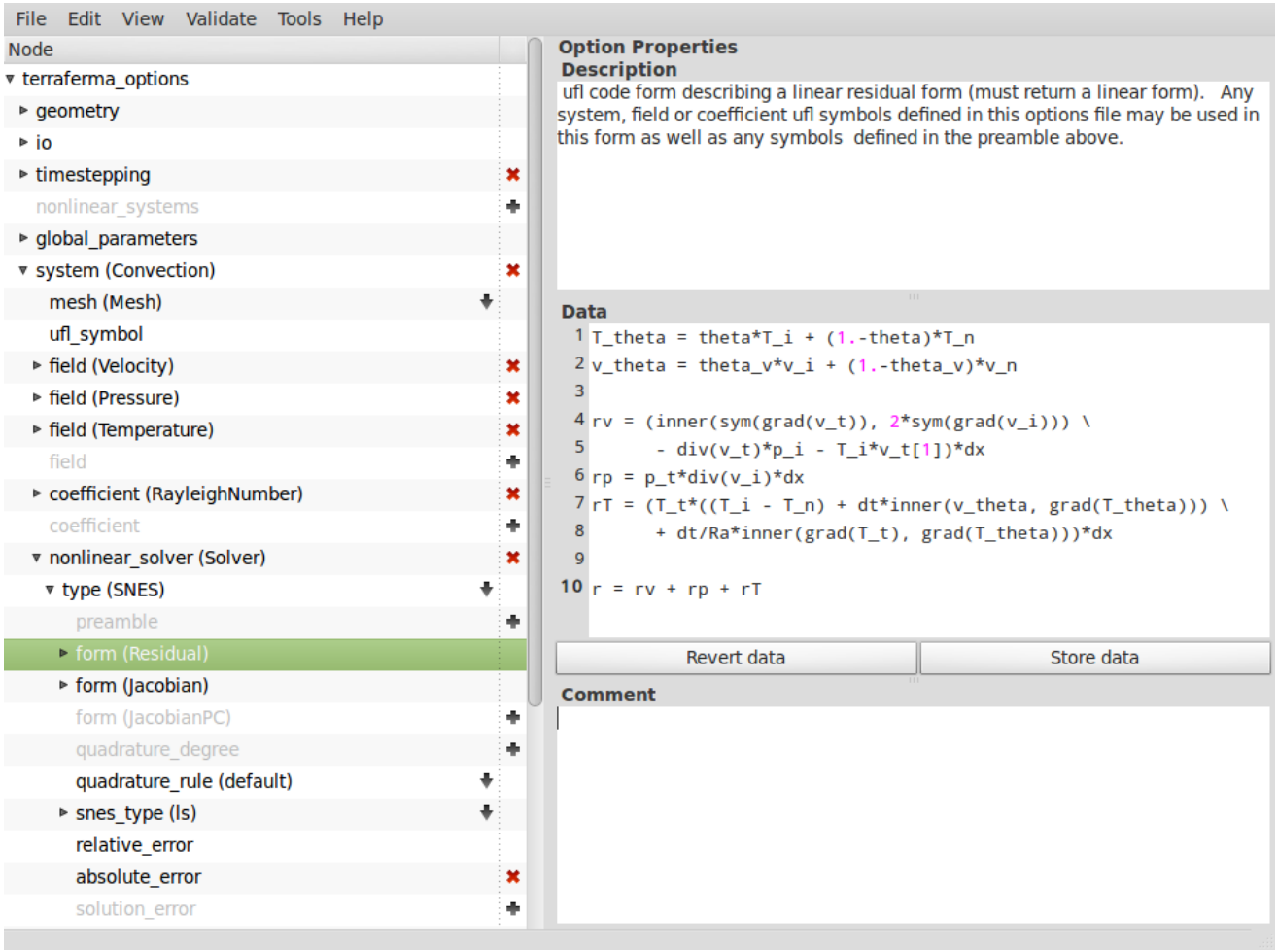

Figure 7. Screenshot showing options for the options for the system **Convection** with three fields $\boldsymbol{u} = (\boldsymbol{v}, P, T)$, a coefficient and a non-linear solver. The non-linear solver contains the UFL describing the weak form of the residual (highlighted, compare with UFL in Section 3). The Jacobian is calculated by automatic differentiation with `J = derivative(r,us_i,us_a)`. Other options control stopping/convergence criteria, monitoring and choice of linear solver for each Newton iteration.

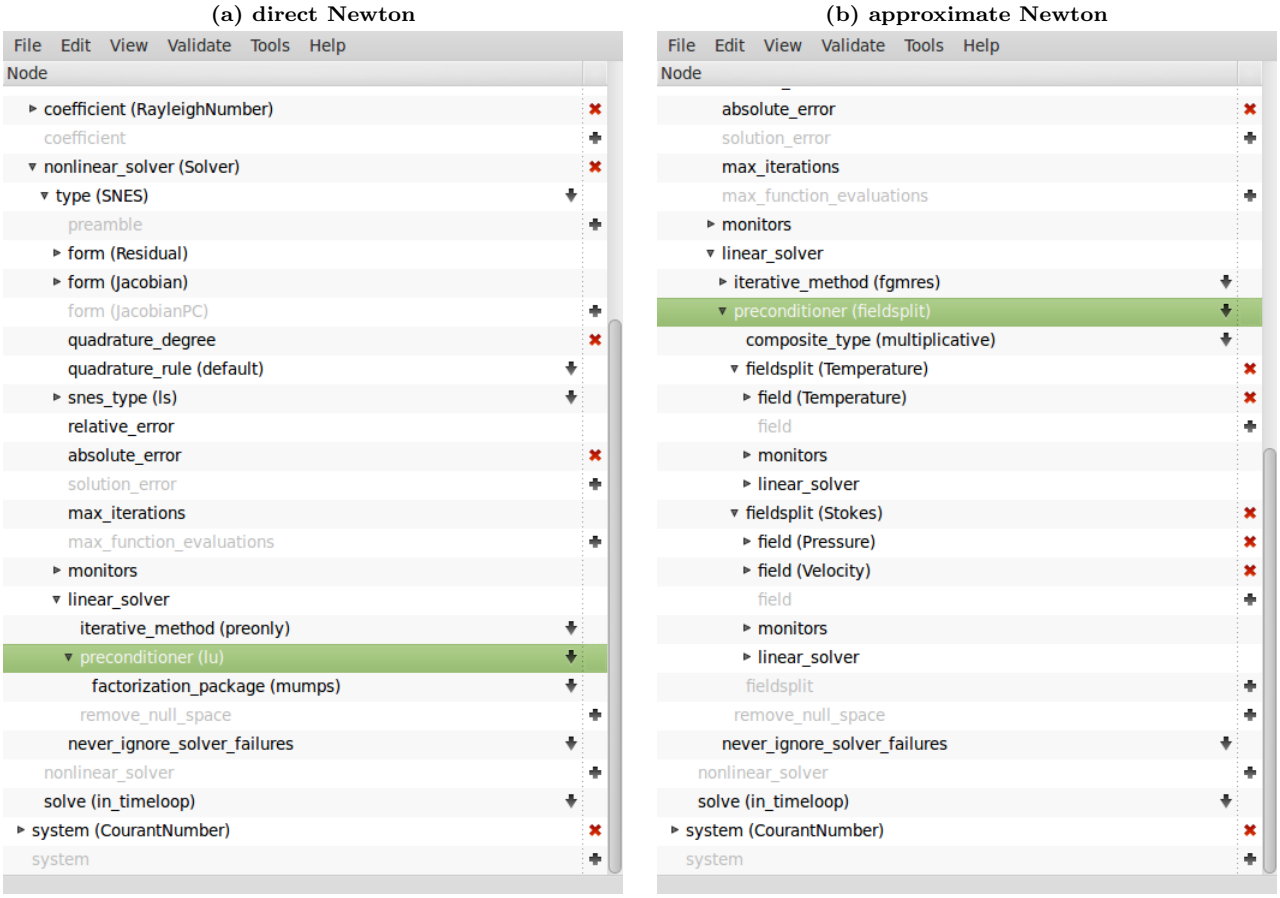

Figure 8. Screenshot comparing linear solver choices for (**a**) direct Newton `(preonly, lu)` versus (**b**) Newton fieldsplit direct `(fgmres, fieldsplit)` with multiplicative fieldsplits for T and Stokes $(\boldsymbol{v}, p)$. Unfolding the linear solver for each split would show a direct solve on each split (i.e. `preonly, lu`).

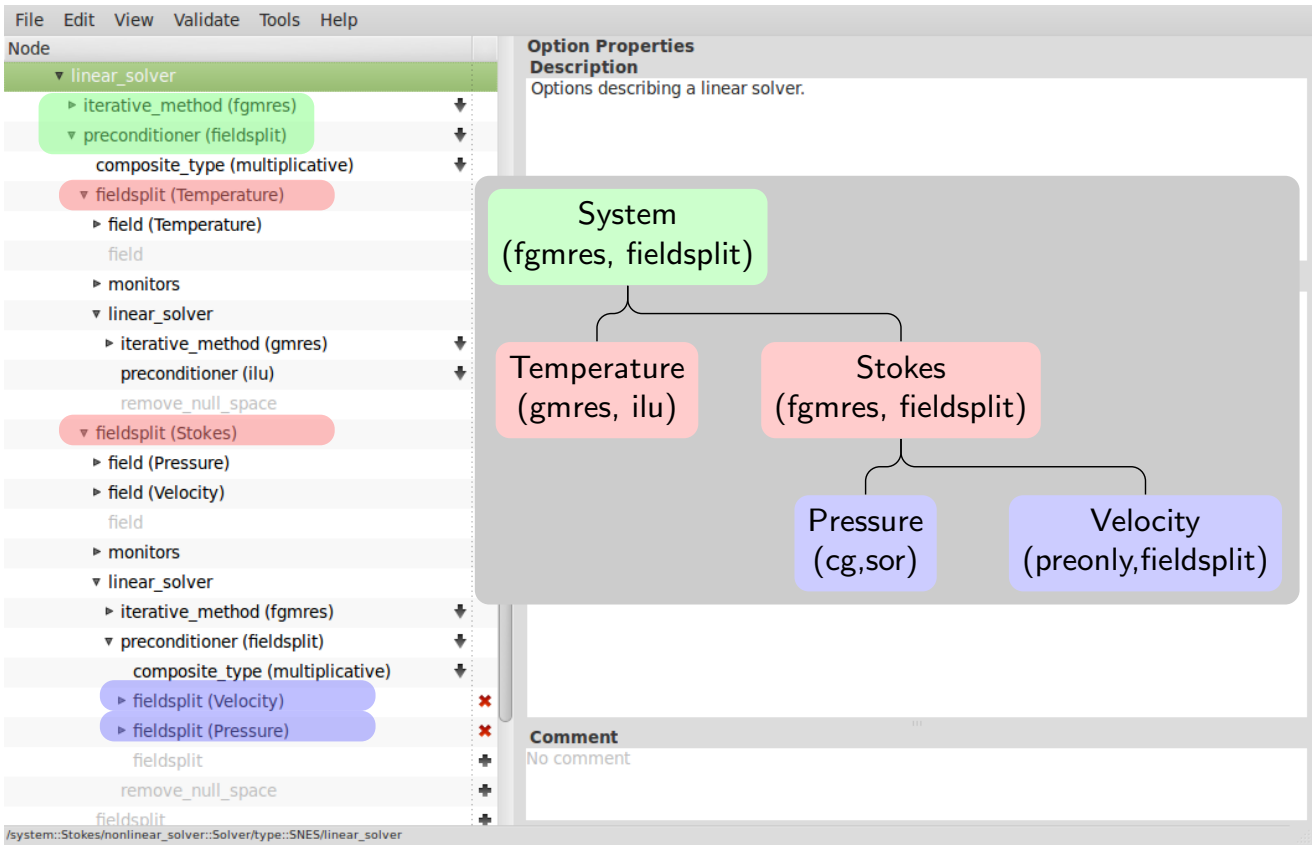

Figure 9. Tree structure (right) and options (left) for describing a three level nested fieldsplit that replaces the direct solver for stokes shown in Figure 8b with an iterative, block-preconditioned solver.

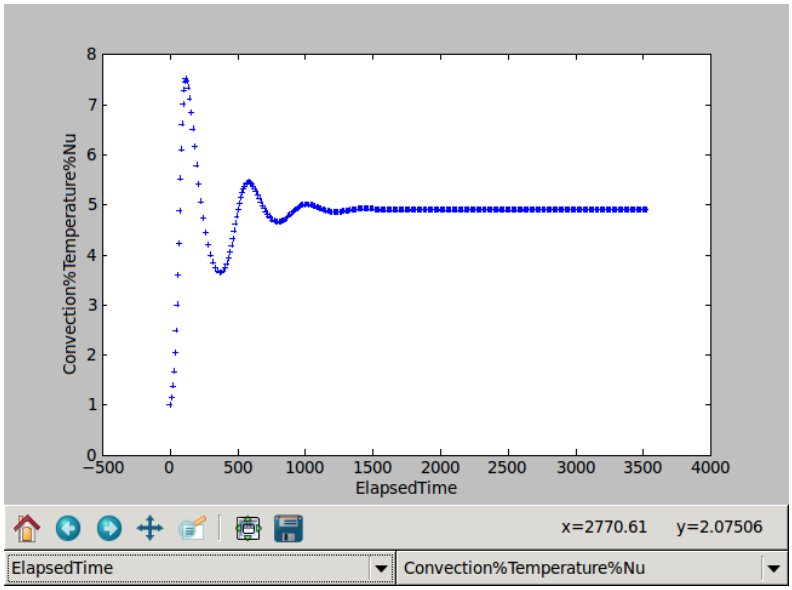

Figure 10. Screenshot of monitoring tool `tfplot`, showing variation in the Nusselt Number as function of simulation time. All variables included for statistics output report max and min values as well as residual, elapsed wall time and other basic statistics.

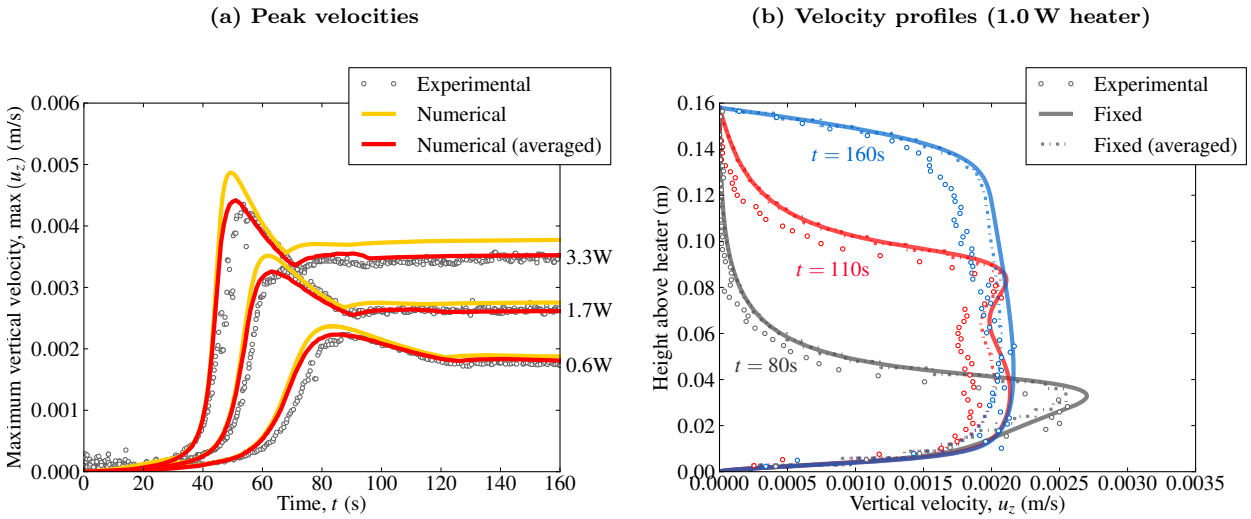

Figure 11. (a) Benchmark results for *Vatteville et al.* [2009] showing maximum velocities vs. time in the laboratory experiments and the numerical simulations for a cylindrical plume. Following *Vatteville et al.* [2009], the numerical data is averaged in 3 mm squares to mimic the effect of laboratory data collection. The effect of this averaging can also be seen in velocity profiles of the numerical and laboratory data above the heater (b, heater power of 1.0 W).

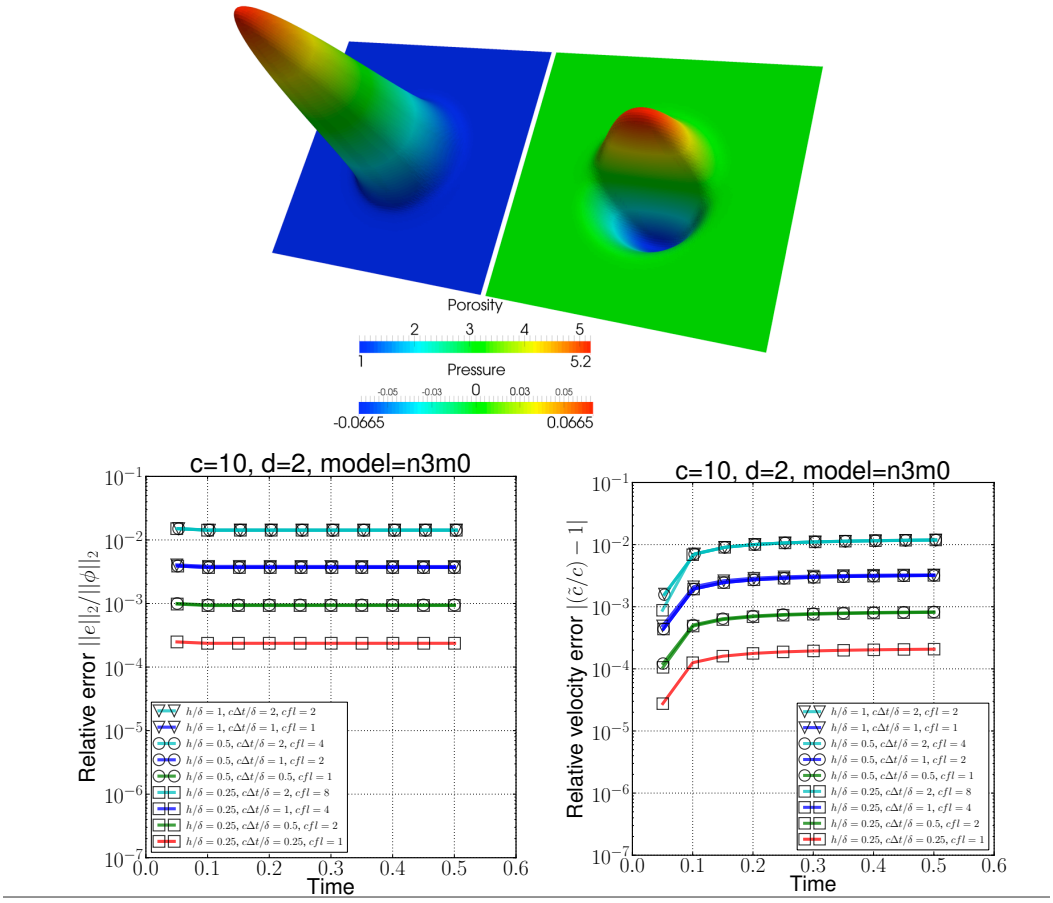

Figure 12. Example magmatic solitary wave benchmarks from *Simpson and Spiegelman* [2011]. Top figure shows porosity and pressure fields for a 2D porosity wave with speed $c = 10$, permeability exponent $n = 3$, and bulk-viscosity exponent $m = 0$. These waves should propagate at constant porosity c without changing form and provides a fully non-linear benchmark problem. Lower figures show shape and velocity errors as a function of time . This benchmark solves the problem in a frame moving with the solitary waves using a semi-Lagrangian advection scheme in TerraFERMA. Additional results for other amplitude waves in 2D and 3D, and other advection schemes are in the supplemental material.

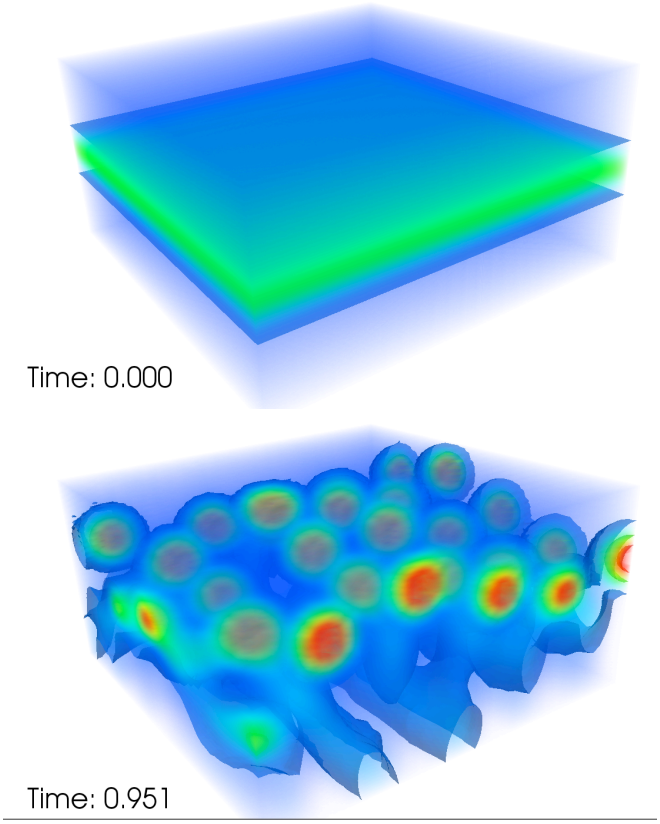

Figure 13. Instability of 1-D solitary wave to spherical 3D solitary waves.

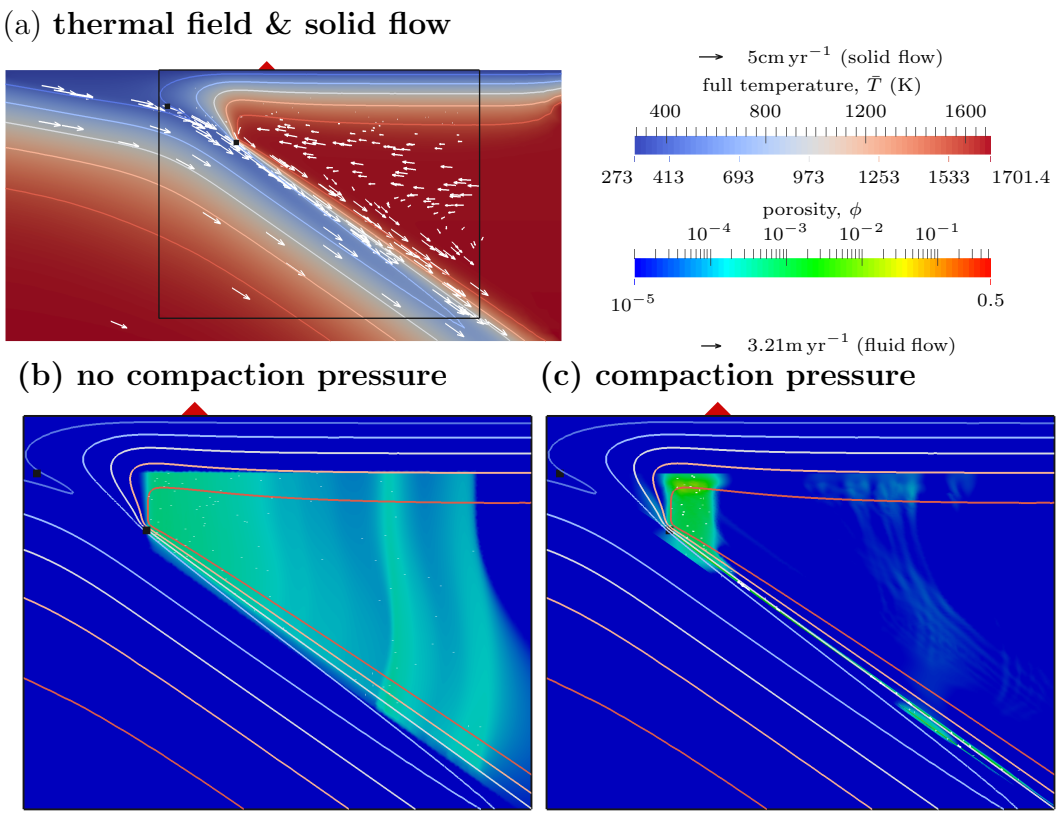

Figure 14. An example subduction zone model for coupled fluid-solid flow beneath the Tohoku arc demonstrating the ability of TerraFERMA to explore different model physics. (**a**) solid flow and thermal structure calculated using parameters from *van Keken et al.* [2011]. (**b,c**) isotherms, fluid fraction (porosity) and fluid flux (arrows) calculated for two different sets of equations used to model fluid flow. Both calculations have the same solid flow, thermal structure and fluid sources calculated using the thermodynamic look up tables for hydrated basalt, gabbro and depleted mantle from *van Keken et al.* [2011]. Domain is black box shown in (**a**). (**b**) Approximate fluid flow calculation similar to *Cagnioncle et al.* [2007] where fluid flow is driven only by gravity and solid advection. For this high-permeability case, fluid flow is primarily vertical. (**c**) full fluid flow calculations including "compaction pressure" gradients developed by fluid interaction with solid rheological and permeability variations. These gradients can lead to significantly different fluid flow paths and concentration of fluids beneath the volcanic arc. Details of these calculations are found Wilson 2013, submitted.