



DOUAR: A new three-dimensional creeping flow numerical model for the solution of geological problems

Jean Braun^{a,*}, Cédric Thieulot^{a,1}, Philippe Fullsack^b, Marthijn DeKool^{c,2}, Christopher Beaumont^b, Ritske Huismans^{b,1}

^a Géosciences Rennes, Université de Rennes 1, Rennes Cedex CS 35042, France

^b Department of Oceanography, Dalhousie University, B3H 4J1 Halifax, N.S., Canada

^c Research School of Earth Sciences, The Australian National University, Canberra, ACT 0200, Australia

ARTICLE INFO

Article history:

Received 29 October 2007

Received in revised form 18 April 2008

Accepted 9 May 2008

Keywords:

Geodynamics

Lithosphere

Deformation

Numerical modelling

Finite elements

Octree

ABSTRACT

We present a new finite element code for the solution of the Stokes and energy (or heat transport) equations that has been purposely designed to address crustal-scale to mantle-scale flow problems in three dimensions. Although it is based on an Eulerian description of deformation and flow, the code, which we named DOUAR ('Earth' in Breton language), has the ability to track interfaces and, in particular, the free surface, by using a dual representation based on a set of particles placed on the interface and the computation of a level set function on the nodes of the finite element grid, thus ensuring accuracy and efficiency. The code also makes use of a new method to compute the dynamic Delaunay triangulation connecting the particles based on non-Euclidian, curvilinear measure of distance, ensuring that the density of particles remains uniform and/or dynamically adapted to the curvature of the interface. The finite element discretization is based on a non-uniform, yet regular octree division of space within a unit cube that allows efficient adaptation of the finite element discretization, i.e. in regions of strong velocity gradient or high interface curvature. The finite elements are cubes (the leaves of the octree) in which a q_1-p_0 interpolation scheme is used. Nodal incompatibilities across faces separating elements of differing size are dealt with by introducing linear constraints among nodal degrees of freedom. Discontinuities in material properties across the interfaces are accommodated by the use of a novel method (which we called divFEM) to integrate the finite element equations in which the elemental volume is divided by a local octree to an appropriate depth (resolution). A variety of rheologies have been implemented including linear, non-linear and thermally activated creep and brittle (or plastic) frictional deformation. A simple smoothing operator has been defined to avoid checkerboard oscillations in pressure that tend to develop when using a highly irregular octree discretization and the tri-linear (or q_1-p_0) finite element. A three-dimensional cloud of particles is used to track material properties that depend on the integrated history of deformation (the integrated strain, for example); its density is variable and dynamically adapted to the computed flow. The large system of algebraic equations that results from the finite element discretization and linearization of the basic partial differential equations is solved using a multi-frontal massively parallel direct solver that can efficiently factorize poorly conditioned systems resulting from the highly non-linear rheology and the presence of the free surface. The code is almost entirely parallelized. We present example results including the onset of a Rayleigh–Taylor instability, the indentation of a rigid-plastic material and the formation of a fold beneath a free eroding surface, that demonstrate the accuracy, efficiency and appropriateness of the new code to solve complex geodynamical problems in three dimensions.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, modelling the Earth's crust and upper mantle deformation has led to increased insight concerning the way the Earth responds to both tectonic and erosional forcing and indirectly to the climate (Willett et al., 1993; Batt and Braun, 1997; Beaumont et al., 2001). Apart from a few exceptions (Braun, 1993, 1994; Braun and Beaumont, 1995), such modelling has been limited

* Corresponding author. Fax: +33 22323 6780.

E-mail address: Jean.Braun@univ-rennes1.fr (J. Braun).

¹ Now at Department of Earth Science, Bergen University, N-5007 Bergen, Norway.

² Now at Geosciences Australia, GPO Box 378, Canberra, ACT 2601, Australia.

to two-dimensional analysis, mostly for computational efficiency reasons. Convection or mantle-scale flow calculations have been more easily performed in three dimensions, in part due to the relatively simple geometry of the problem and the lack of interactions with a free or eroding upper surface (Houseman, 1988; Albers, 2000; Tackley, 1998, among many others).

The need for a three-dimensional model capable of taking into account the large stresses arising from surface topography gradient is however growing (Braun, 2006). Key questions regarding the potential couplings and feedbacks between tectonics, erosion and climate can only be properly addressed using a plan-view surface processes model and, thus a full three-dimensional representation of deformation in the underlying crust (Stolar et al., 2005).

Three dimensional calculations are inherently computationally costly. Using a uniform spatial discretization, most three-dimensional convection models are limited in their spatial resolution to meshes of the order of 100^3 elements or nodes (Tackley, 1998, for example). Owing to the non-linear and localizing nature of lithospheric rheologies, deformation gradients are much greater in the lithosphere than in the convecting parts of the Earth's mantle and a finer spatial discretization is required to capture them. This is the reason why complex meshing algorithms have been developed and are commonly used for the solution of lithospheric-scale deformation or flow problems in two dimensions (Braun and Sambridge, 1994). These are, however, difficult to generalize to three dimensions. Furthermore, the evolving nature of the deformation or velocity field (in part due to the formation of shear zones or faults) requires the use of adaptive meshing techniques in which the numerical spatial discretization evolves with the flow.

Two contrasting methodologies have commonly been used to solve lithospheric-scale deformation problems. Explicit time-stepping methods are based on the dynamical force balance equation ($\mathbf{F} = m\mathbf{y}$) in which a pseudo-mass (m) has been introduced to damp numerical oscillations. These methods require relatively few operations per time steps but a large number of time steps. There have been several implementations of these implicit algorithms to solve problems of lithospheric deformation in two dimensions (Poliakov et al., 1993; Hassani et al., 1997). Remarkably, none of these have been so far ported to three-dimensions. Implicit methods in which the equations of static equilibrium have been linearized to form a large system of algebraic equations require less time steps but become computationally expensive in three dimensions. Multigrid iterative methods are commonly used to solve these large systems of equations but their convergence is poor when dealing with highly non-linear problems or those involving a free surface (Moresi and Solomatov, 1998).

Here we present a newly developed finite element code to solve the three dimensional Navier–Stokes equations that we purposely developed to address $Re = 0$, high Ra and infinite Pr flows characterized by a free and/or eroding surface. The new model, that we called DOUAR, is in principle capable of tracking any interface, such as the Moho or a stratigraphic marker, deforming with the flow. It is based on a multi-scale octree-based discretization method and uses a fast, yet accurate direct solver for the solution of the large system of algebraic equations resulting from the implicit time-stepping, finite element discretization of the static force balance equations. In this paper, we present in detail the various components of this new code that are based on existing and novel algorithms. We also present the results of selected computations that demonstrate the usefulness and accuracy of the methodology.

2. Basic equations

The deformation of the Earth's lithosphere and underlying mantle are commonly regarded as similar to that of a high viscosity,

viscoplastic material deforming at a sufficiently low speed that inertial forces can be neglected (zero Reynolds number flow) and that heat is conducted faster than dissipated by viscous flow (infinite Prandtl number flow). Under such conditions, the velocity field \mathbf{v} and pressure p must obey the following simplified form of the momentum or Navier–Stokes equations, sometimes referred to as the Stokes equations:

$$\nabla \cdot \mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T) - \nabla p = \rho \mathbf{g} \quad (1)$$

where \mathbf{g} is the gravitational acceleration vector. Under the assumption that such a flow is incompressible, the divergence of the velocity must also be nil:

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

Pressure can be eliminated from these equations by making the approximation that the material is nearly incompressible and introducing a so-called penalty or compressibility factor, λ :

$$-\lambda \nabla \cdot \mathbf{v} = p \quad (3)$$

λ has the dimensions of a viscosity (Pa s) and is commonly taken to be eight orders of magnitude larger than the shear viscosity, μ , which ensures a nearly incompressible behaviour for the flow.

At high temperature, rocks deform by creep, a non-linear form of viscous deformation that is commonly approximated by defining a stress or strain rate dependent and thermally activated viscosity in the above equation:

$$\mu = \mu_0 \dot{\epsilon}^{1/(n-1)} e^{Q/nRT} \quad (4)$$

At low temperature, rocks deform by brittle failure that is also approximated by adapting the viscosity to limit the stress that is generated during deformation. This 'cap' on the stress level is parameterized by various failure criteria that have been derived from laboratory experiments. These criteria usually take the form of a yield criterion F that is expressed in terms of the stress tensor σ and of material constants:

$$F(\sigma, \sigma_0, \dots) = 0 \quad (5)$$

As the yield criterion should be independent of the orientation of the coordinate system employed, it should only be a function of stress invariants:

$$\begin{aligned} J_1 &= \sigma_{ii} \\ J'_2 &= \frac{1}{2} s_{ij} s_{ij} \\ J'_3 &= \frac{1}{3} s_{ij} s_{jk} s_{ki} \end{aligned} \quad (6)$$

where \mathbf{s} is the deviatoric stress tensor defined as follows:

$$\mathbf{s} = \sigma - \frac{1}{3} \text{Tr}[\sigma] \mathbf{1}. \quad (7)$$

The Mohr–Coulomb criterion is commonly used to represent the behaviour of rocks and requires two rheological parameters, ϕ the dimensionless internal angle of friction and c the cohesion that has units of pressure:

$$\tau = c - \sigma_n \tan \phi \quad (8)$$

where τ is the magnitude of the shearing stress and σ_n is the normal stress. It can also be expressed in terms of stress invariants:

$$\sqrt{J'_2} = \frac{m(\theta_l, \phi) \sin \phi}{3} J_1 + m(\theta_l, \phi) c \cos \phi \quad (9)$$

where:

$$m(\theta_l, \phi) = \frac{\sqrt{3}}{\sqrt{3} \cos \theta_l + \sin \theta_l \sin \phi} \quad (10)$$

and θ_l is the Lode angle, defined as:

$$\theta_l = -\frac{1}{3} \sin^{-1} \left[\frac{3\sqrt{3} J_3^{3/2}}{2 J_2'} \right] \quad (11)$$

Under some circumstances or in different materials, this pressure dependence can be neglected and the equation is simplified to become the von Mises criterion:

$$F_{VM} = J_2' - \sigma_0 = 0 \quad (12)$$

which depends on a single parameter σ_0 , the yield constant.

Because rock material properties such as density and viscosity depend on temperature, it is also necessary to compute the temperature within the deforming system. This is done by solving the energy or heat transport equation which has temperature T as an unknown:

$$\rho c \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) = \nabla \cdot k \nabla T + \rho H \quad (13)$$

k is the thermal conductivity, ρ is density, c is heat capacity and H is heat production per unit mass. The relative importance of the advective term with respect to the conductive term is measured by the value of the dimensionless Peclet number, $Pe = v_0 L / \kappa$ where v_0 and L are typical velocity and length characterizing the system and $\kappa = k / \rho c$ is the thermal diffusivity. In most active tectonic systems, Pe is large ($1 < Pe < 100$) and, therefore, the advective term must be included.

The density ρ varies as a function of temperature according to:

$$\rho = \rho_0 (1 - \alpha(T - T_0)) \quad (14)$$

where α is the coefficient of thermal expansion and ρ_0 is the value of the density at $T = T_0$.

3. Finite element discretization

Among the many methods that have been devised to solve this set of partial differential equations, the finite element method is one of the most commonly used, mainly because of its geometrical flexibility, i.e. how it can solve problems with complex (non-rectangular) geometries or those requiring a non-uniform discretization to represent efficiently localized flow/deformation. It is based on the assumption that the solution of the PDEs, in our case, the components of the velocity field, the pressure and the temperature, can be approximated by their values at a finite number of points or nodes and, between these points, by a set of piecewise interpolation functions (or shape functions) defined inside *finite elements* connecting the nodes. Under this set of assumptions, a good approximation of the solution to the PDEs can be obtained by solving the following set of integral equations, obtained by the so-called Galerkin method or approximation:

$$\left[\int_V \mathbf{B}_v^T \mu \mathbf{B}_v dV + \int_V \mathbf{B}_v^T \lambda \mathbf{B}_v dV \right] \mathbf{v} = \int_V \mathbf{N}_v^T \rho \mathbf{g} dV$$

$$\left[\int_V \mathbf{N}_t^T \rho c \mathbf{N}_t dV \right] \dot{T} + \left[\int_V \mathbf{N}_t^* \mathbf{v} \mathbf{B}_t dV + \int_V \mathbf{B}_t^T k \mathbf{B}_t dV \right] T = \int_V \mathbf{N}_t^T \rho H dV \quad (15)$$

where V is the problem domain over which the solution is sought and $\mathbf{N}_{v,t}$ and $\mathbf{B}_{v,t}$ are the shape function matrix and the shape function derivative matrix, approximating the velocity and temperature and their spatial derivatives within each element from their values at the nodes connected by the elements. This is a standard approximation used in problems involving visco-plastic rheologies (Hughes et al., 1979) and has been used to study mantle flow

problems (Tackley, 2000a), but also to represent the non-linear, brittle and viscous behaviours of the Earth's crust (Fullsack, 1995), as well as strain localization in the mantle (Tackley, 2000b) and in the lithosphere (Huismans and Beaumont, 2002; Huismans et al., 2005). Note that to ensure stability of the solution of the heat equation in cases where advection dominates over conduction (large Pe number cases), a modified version of the shape function matrix $\mathbf{N}^* = \mathbf{N} + \tau \mathbf{v} \mathbf{B}$ is used where $\tau = \Delta l \sqrt{15} / \|\mathbf{v}\|$ and Δl is a length representing the linear dimension of the finite element (Hughes and Brooks, 1982).

In a classical finite element implementation, the integrals in the above equations are evaluated element by element and using an approximate integration scheme (Gauss–Legendre) that requires estimating the integrand at a finite number of points within each finite element. Note that the integration of the compressibility term has to be performed using a lower order integration scheme (1 point integration) than for the other terms (8 point integration) to avoid 'locking' of the solution. We will show how this 'mixed-order' integration scheme can be improved upon when dealing with problems where material properties, such as the viscosity or the density, vary within an element.

These finite element equations are then transformed into a set of linear algebraic equations having the degrees of freedom (or 'dofs') of the problem, i.e. the nodal velocities and temperatures, as unknowns. This linearization involves decoupling the two sets of equations (momentum and energy), linearizing the viscosity (resulting from the implementation of Eqs. (4), (12) and (8)) and performing a series of iterations, or successive solutions of the equations in which the non-linear parameters/terms are updated. These equations are usually written as:

$$\mathbf{A}_v \mathbf{v} = \mathbf{b}_v$$

$$\mathbf{A}_t \mathbf{T} = \mathbf{b}_t \quad (16)$$

where \mathbf{v} and \mathbf{T} are the vectors of the nodal velocities and temperatures, which are regarded here as the unknowns. Note that the use of a penalty method to eliminate the pressure from the momentum/incompressibility equations will affect the conditioning of the resulting finite element matrix \mathbf{A}_v , whereas the advection terms in the energy equation renders the resulting finite element matrix \mathbf{A}_t non-symmetrical; both of these characteristics will determine the choice of a particular method of solution of these very large sets of equations.

4. Octree division of space

As stated above, many problems require a non-uniform discretization of space. The use of irregular triangular meshes is common in two-dimensional analyses but becomes relatively impractical in three dimensions. For this reason, we chose an octree-based discretization of space (Cheng et al., 1986) which combines the flexibility of a non-uniform discretization while being regular. An octree is a geometrical construct that divides three-dimensional space in a space-filling set of cubes of varying size that are used here as basic eight-noded or q_1 – p_0 finite elements (Fig. 1).

The unit cube is said to be of level zero as it counts only $(2^0)^3 = 1$ leaf. After one division, the octree comprises $(2^1)^3 = 8$ leaves and is of level $L = 1$. Performing another subdivision of each leaf leads to a regular level $L = 2$ octree of $(2^2)^3 = 64$ leaves. Consequently, a $32 \times 32 \times 32$ grid, which is a standard grid for most three-dimensional finite element codes, is a level 5 octree with 32,768 leaves.

In the simple q_1 – p_0 finite element, the basis functions for the velocity are tri-linear whereas the pressure is assumed uniform within each element (Cheng et al., 1986). Where cubes of different size share a common face, some of the nodes that are at the corners

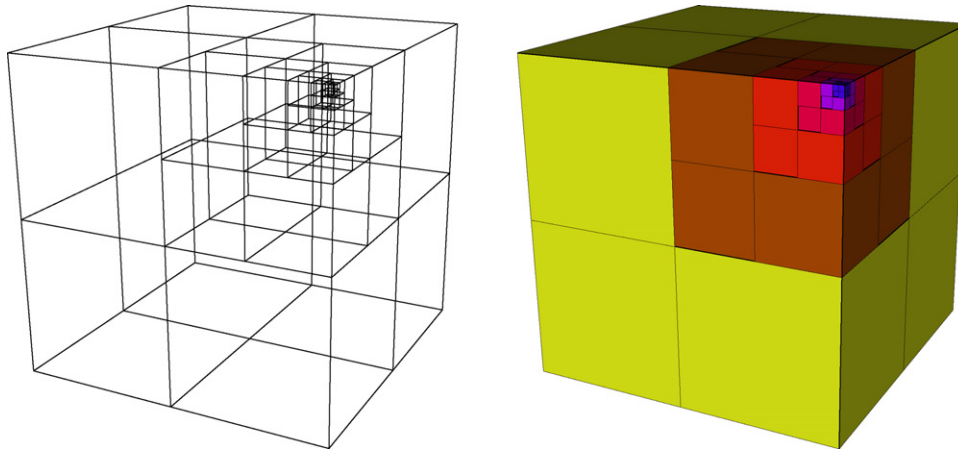


Fig. 1. Example of a simple octree discretization of the unit cube. The unit cube is divided in eight sub-cubes, which can be arbitrarily divided into eight sub-sub-cubes, and so on. The sub-cubes that remain undivided at the end of the construction of the octree are called leaves which are used here as finite elements with which the partial differential equations are solved.

of the small elements do not exist in the adjacent large elements. These are called ‘bad faces’ that are dealt with by imposing linear constraints (Webb, 1990) as shown in Fig. 2.

The q_1 – p_0 elements are known to be affected by the presence of a “checkerboard” mode in the pressure field (Bathe, 1982). The introduction of the linear constraints into the set of finite element equations may also contribute to these unwanted oscillations. To minimize them, we “smooth” the pressure field by performing a double interpolation of the elemental pressure onto the nodes, and then back onto the elements. The element-to-node interpolation is performed by averaging the elemental values from elements common to each node; the node-to-element interpolation is performed by averaging the nodal values element-by-element. This method is not only very efficient but produces a smoothing of the pressure that is adapted to the local density of the octree and can be shown to be equivalent to a least-square smoothing of the pressure.

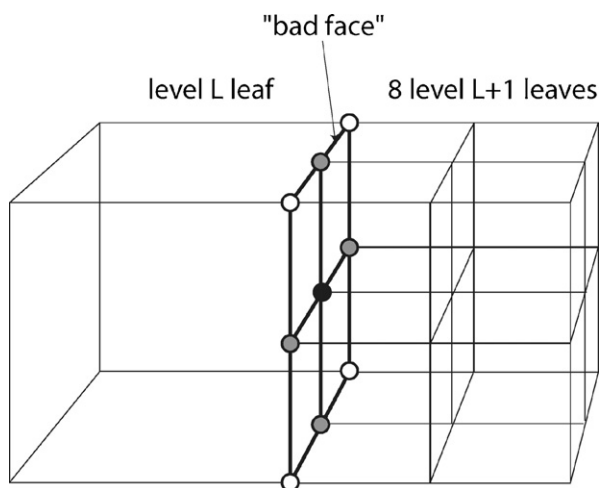


Fig. 2. The interface between two sets of leaves of different level is called a ‘bad face’. These bad faces contain nodes that belong to the smaller elements on one side of the face and not to the larger element on the other side of the face. These nodes are filled in black and grey on the figure. Using the finite element method, one can only solve for velocity components and temperature on nodes that belong to elements on both sides of the face (the white nodes). To obtain values at the incompatible nodes, one needs to impose additional linear constraints that constrain the solution at the grey mid-side nodes to be the mean of the two adjacent red nodes, and the solution at the central black node to be the mean of the four corners nodes.

Octrees are very simple and memory-efficient entities that can be built as a single integer array containing, for each cube of the octree, the address in the array of the first of its eight ‘children cubes’. In the scheme we have developed here, when a cube is not divided, it becomes a leaf to which a name/number is associated and is stored in the octree integer array as a negative number (to indicate that it corresponds to a leaf number and not a child’s address). This scheme is memory efficient (most octrees are only a few kilobytes in size) but requires additional operations when performing operations on the octree. However, most of the operations commonly needed in the construction of a finite element problem are done with great efficiency when using the octree storage scheme described above. Most global operations (i.e. those affecting all the leaves of an octree) require only $\sim N$ arithmetic or conditional operations. For example, for an octree of maximum depth L_{\max} (level of the smallest leaf or finite element in the octree) and made of N leaves:

- to create a leaf at level L around a point of known coordinates; this requires L conditional statements;
- to locate a point of known coordinates, i.e. to find the name/number of the leaf it belongs to; we will call this a ‘location’ operation and is achieved through L_{\max} conditional statements for each of the point coordinates;
- to determine the size of all leaves/elements; this operation involves $\sim N$ conditional statements
- to find the list of neighbouring leaves/elements; this operation involves $\sim 26N$ location operations;
- to interpolate a field known at the nodes of an octree; this operation involves a location and a trilinear interpolation operation per interpolation;
- to unite two octrees; this involves checking the depth of one octree for each leaf of the other octree and, if necessary, creating a leaf;
- to smooth an octree (see Fig. 3); this requires $\sim 6N$ location and conditional operations.

When using the simple scheme described above to store the octree, one operation becomes however relatively costly: to find the connectivity matrix between nodes and leaves/elements. This is done here by first numbering the nodes in a redundant manner, i.e. by giving to each element a set of eight nodes, regardless of connectivities between the elements and the possibility that a single node is given many different names/numbers in the sequence, i.e.

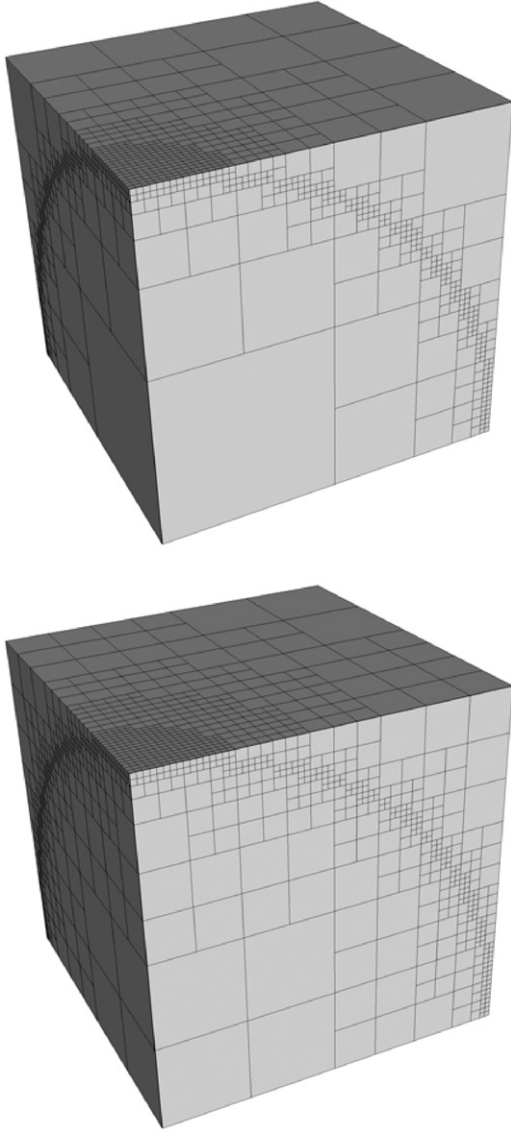


Fig. 3. Example of an octree designed to represent a spherical shell of unit radius. The octree in the top panel has been constructed so that the region surrounding the shell is discretized with ‘leaves’ of level 6. The octree depicted in the bottom panel has been “smoothed”, i.e. the condition that no two adjacent leaves (or elements) can vary in size by more than one level of octree division has been applied.

if it belongs to more than one element. Then the node numbers are ordered (Press et al., 1992) according to their x , y and z positions and checked for common values of one of their coordinates at a time. Redundant nodes are removed and the connectivity matrix is modified accordingly. Note that because nodal coordinates are multiples of $2^{L_{\max}}$, where L_{\max} is the maximum level of any leaf of the octree, it is a well-posed problem to rank and compare them. Once the connectivity matrix is known, the node numbers are ordered to minimize the bandwidth of the resulting finite element matrix using the method developed by Sloan (1989). See Samet (1989), for example, for more detailed information on octree structures.

5. Interface tracking

Material interfaces can be numerous in large-scale tectonic problems. Chief among them is the upper free surface. Its deformation generates large differential stresses that can influence

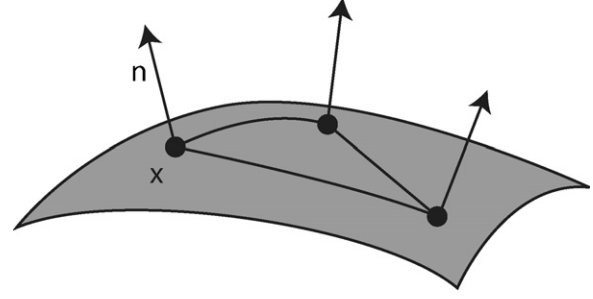


Fig. 4. One of DOUAR’s main feature is its ability to track interfaces. All interfaces are tracked through a set of particles defined by their coordinates \mathbf{x} and their normal to the interface \mathbf{n} . The particles are connected by triangles.

and potentially drive crustal-scale deformation and flow (Braun, 2006). A wide range of tectonic problems, including those in which erosional (and thus potentially climatic) feedback is addressed, require the accurate tracking of the deforming free surface. To track any interface, a dual approach is used combining particle tracking and the definition of a level set function, similar to that of Enright et al. (2005); this ensures both accuracy and efficiency.

5.1. Interface particles

Each interface is first defined by a set of particles of coordinates $\mathbf{x} = (x, y, z)$ and to which unit vectors pointing in the direction normal to the interface $\mathbf{n} = (n_x, n_y, n_z)$ are attached (see Fig. 4).

At each time step, the global coordinates of the interface particles, \mathbf{x} , are advected by interpolation of the computed flow velocities at the mid-point configuration, i.e. the locations half-way between their original and final positions for this time step:

$$\mathbf{x}' = \mathbf{x} + \mathbf{v} \left(\frac{\mathbf{x} + \mathbf{x}'}{2} \right) \Delta t \quad (17)$$

This method is second-order accurate. It does not require the use of the velocity field at the previous time step. Note that during the non-linear iterations, each interface (its defining particles and normals) is advected to its mid-point position, i.e. its position at time $t + \Delta t/2$. This leads to a more accurate (and stable) solution because interfaces usually define regions differing by their material properties (density, viscosity, etc.). Advecting the interfaces therefore ensures that the equations of static equilibrium (Eq. (15)) are solved in the mid-point configuration.

To advect the normals, we devised a simple, yet accurate algorithm: one first needs to compute two orthogonal directions, \mathbf{n}_1 and \mathbf{n}_2 located in the tangential plane to the interface. Two such directions are given by:

$$\begin{aligned} \mathbf{n}_1 &= (\cos \theta \cos \phi, \cos \theta \sin \phi, -\sin \theta) \\ \mathbf{n}_2 &= (-\sin \theta, \cos \theta, 0) \end{aligned} \quad (18)$$

where:

$$\begin{aligned} \theta &= \tan^{-1} \frac{\sqrt{n_x^2 + n_y^2}}{n_z} \\ \phi &= \tan^{-1} \frac{n_y}{n_x} \end{aligned} \quad (19)$$

The normals are then advected using a second-order scheme obtained by Taylor expansion of the velocity field:

$$\mathbf{n}' = \mathbf{n} + (\mathbf{L}\mathbf{n}_1 \times \mathbf{n}_2 + \mathbf{n}_1 \times \mathbf{L}\mathbf{n}_2)\Delta t + (\mathbf{L}\mathbf{n}_1 \times \mathbf{L}\mathbf{n}_2)\Delta t^2 \quad (20)$$

where \mathbf{L} , defined as:

$$\mathbf{L} = \begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{pmatrix} \quad (21)$$

is the velocity gradient matrix computed inside each element using the shape function derivative matrix \mathbf{B} .

The time step is chosen to be smaller than that imposed by the Courant condition:

$$\Delta t < \min_{\text{all elements}} \left(\frac{\Delta x}{v_x}, \frac{\Delta y}{v_y}, \frac{\Delta z}{v_z} \right) \quad (22)$$

5.2. Interface triangulation

The particle coordinates are used to define a triangulation on the interface that is used to add or remove particles according to a set of rules/criteria to ensure that the interface geometry is properly tracked, and to compute a level set function at each node of the octree as shown below. Many criteria can be used to maintain the appropriateness of the particle set to represent the surface with accuracy. For example, on highly deformed interfaces, a condition is imposed that the normals to two particles connected by an edge of the triangulation do not diverge by more than a prescribed angle or that the length of any edge is smaller than a set distance; if they do, particles are injected to reduce the local curvature between adjacent particles or the length of an edge.

Ideally, we would like to use the Delaunay triangulation because it produces relatively evenly shaped triangles, i.e. maximizing the smallest internal angle made by any two sides of any triangle (Sambridge et al., 1995). However, the Delaunay triangulation is commonly defined on a planar surface. To generalize its use to curved, arbitrary surfaces (such as the interfaces tracked in DOUAR), we introduced a new measure of distance between two particles, defined as the ratio of the Euclidian distance between the two particles to the dot product of the two normals to the interface raised to a set power, m :

$$\bar{d}_{12} = \frac{d_{12}}{(\mathbf{n}_1 \cdot \mathbf{n}_2)^m} \quad (23)$$

m is a free parameter that can be adjusted to adapt the method to various types of surfaces (smooth versus creased surfaces for example). For a given Euclidian distance between two points, this distance grows with the local curvature of the surface; it becomes equivalent to the Euclidian distance on a flat surface.

To construct this pseudo-Delaunay triangulation, we first compute any arbitrary triangulation and we update it by performing the so-called ‘in-circle test’ between any two pairs of triangles sharing

an edge (Sambridge et al., 1995). If the third vertex (i.e. not belonging to the common edge) of any of the two triangles lies within the circumcircle constructed from the three nodes of the other triangle, the common edge has to be ‘flipped’, as shown in Fig. 5. We do this for every pair of triangles (or every edge common to two triangles) to obtain the pseudo-Delaunay triangulation connecting the particles. This algorithm is also used to update the triangulation from step to step, as it is deformed by the computed velocity field.

On a planar surface, the in-circle test is equivalent to solving a set of 4 algebraic equations to compute the centre of the circle and its radius. The distance between the fourth point and the centre of the circle is then computed and compared to the radius of the circle. In our case, we wish to perform the in-circle test by using the generalized measure of distance between the particles only (because they are the only locations where the normal to the surface is known). As shown by Pritchard (2005), the two sums of alternate angles in a convex, cyclic $2n$ -gon are equal to $(n-1)\pi$. By definition, the corners of cyclic polygons lie on a circle. Thus, the sum of two alternate angles of any quadrilateral inscribed on a circle is π . Consequently, one can easily show that if a point, X lies inside/outside the circumcircle of three other points A, B, C , the sum of the angles $\hat{A}BC$ and $\hat{A}XC$ is greater/smaller than π . By analogy to the 2D (planar) situation, in our construction of the pseudo-Delaunay triangulation on a curved interface, we have thus replaced the in-circle test by this test on angles $\hat{A}BC$ and $\hat{A}XC$ that can be easily computed from the distances between the points according to:

$$\begin{aligned} \bar{d}_{AC}^2 &= \bar{d}_{AB}^2 + \bar{d}_{BC}^2 - 2\bar{d}_{AB}\bar{d}_{BC} \cos \hat{A}BC \\ \bar{d}_{AC}^2 &= \bar{d}_{AX}^2 + \bar{d}_{XC}^2 - 2\bar{d}_{AX}\bar{d}_{XC} \cos \hat{A}XC \end{aligned} \quad (24)$$

Note that our algorithm is not completely internally consistent. Indeed, Pritchard (2005)’s property cannot be generalized to any non-Euclidian metric. Our experience shows however that the construction of the pseudo-Delaunay triangulation we propose here is accurate when the particle density is relatively high, especially in regions where the surface curvature is high. We therefore caution against using this algorithm for poorly sampled surfaces and acknowledge that further work needs to be done to improve the method we propose. We have also tested this algorithm for constructing a pseudo-Delaunay triangulation by using another measure of distance, namely the distance measured at the surface of the sphere defined by the four points of two adjacent triangles. When the point density remains relatively high, especially in regions of high surface curvature, the two methods converge.

As stated above, during deformation of the interface, particles are injected when the distance between two particles belonging to an edge of the triangulation becomes larger than a prescribed value or when the dot product of the normals at the ends of any given edge becomes smaller than a set value.

To illustrate this algorithm, we have computed the triangulation of an initial set of $51^2 = 2601$ particles regularly spaced on a plane

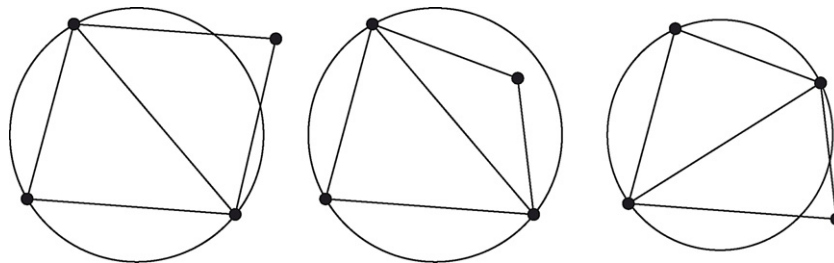


Fig. 5. After deformation of the surface by advection of the particles, each edge of the triangulation is considered for the ‘in-circle’ test/property of the Delaunay triangulation, as illustrated here. The two adjacent triangles shown in the left panel are ‘Delaunay’ as the particle on one side of the edge is not contained within the circle defined by the other three particles; the triangles in the central panel are not Delaunay and their common edge must be ‘flipped’ as indicated in the right panel.

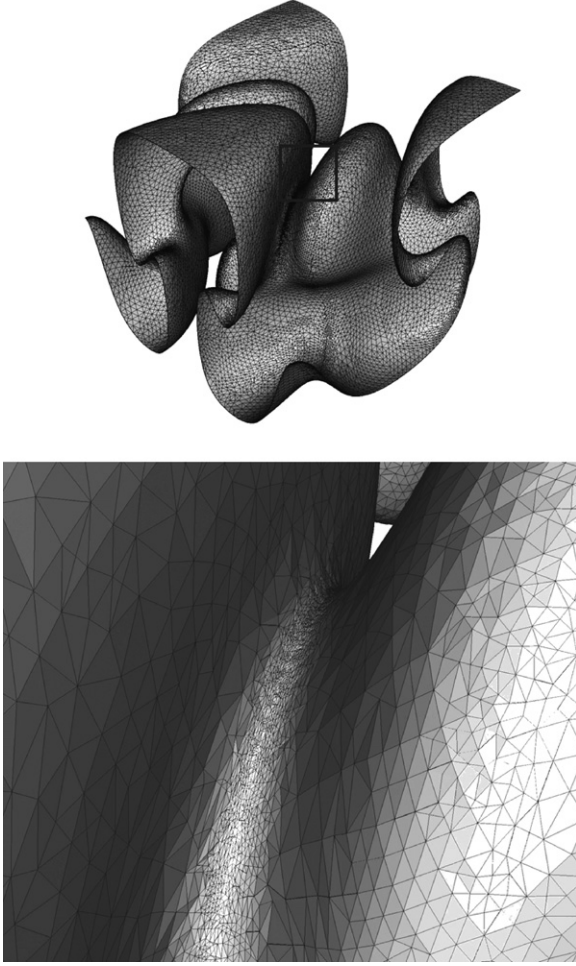


Fig. 6. Geometry and triangulation of an originally flat surface deformed by a prescribed periodic and incompressible velocity field. The bottom panel is a close-up of a region (indicated on the top panel) where the surface curvature has become very large.

of dimension 1×1 , defining $2 \times 50^2 = 5000$ triangles, and located at $z = 0$ and subjected to an imposed velocity field given by:

$$\begin{aligned} v_x &= \frac{\sin(2\pi x) \cos(4\pi y) \sin(\pi z)}{2} + \sin(\pi x) \cos(2\pi y) \sin(\pi z) \\ v_y &= \frac{\cos(2\pi x) \sin(4\pi y) \sin(\pi z)}{4} + \frac{\cos(\pi x) \sin(2\pi y) \sin(\pi z)}{2} \\ v_z &= \cos(2\pi x) \cos(4\pi y) \cos(\pi z) + \cos(\pi x) \cos(2\pi y) \cos(\pi z) \end{aligned} \quad (25)$$

After 1200 steps, the number of particles has grown to 48,038 and the number of triangles to 95,172. The deformed interface and the updated triangulation are shown in Fig. 6.

6. Level set functions

Knowing the geometry of interfaces, we need to pass that information onto the finite element grid (here an octree) to build the finite element equations that are functions of material properties (such as density or viscosity) which vary strongly across interfaces. For each interface, we first build an octree that has high resolution, i.e. small leaves, in the vicinity of the particles defining the interface.

We calculate the signed distance between the interface and the subset of nodes of the octree located in the vicinity of the interface. This subset is built by determining the leaves of the octree that contain the particles used to track the interface deformation.

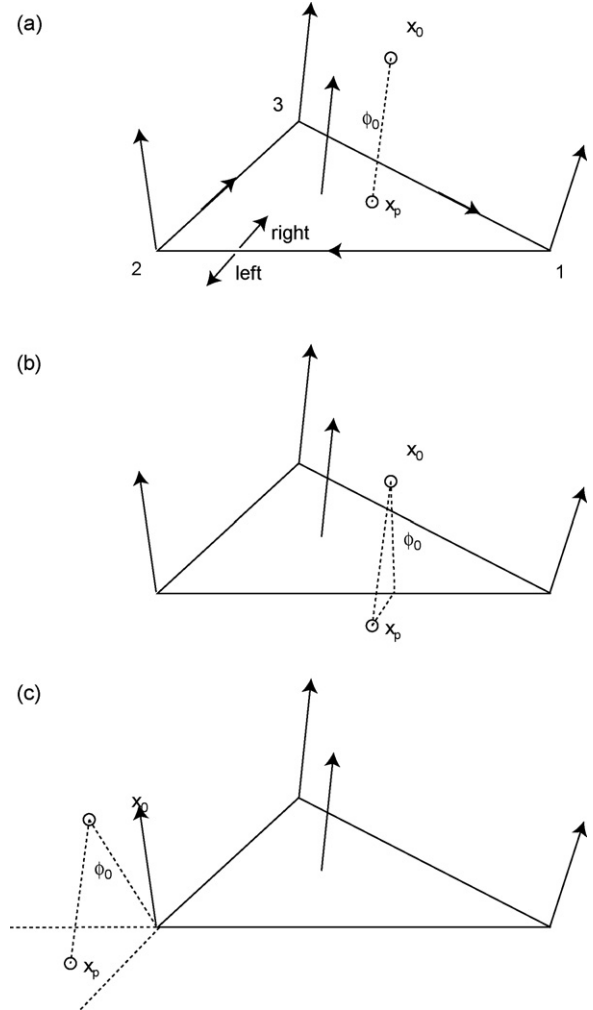


Fig. 7. Graphical description of how the level set function is estimated by projection of the nodes (x_0) of the octree onto the plane of the triangles connecting the particles located on the interface (x_1, x_2, x_3). Three cases are considered depending on the location of the projection (x_p). See text for details of the method.

The function is called a level set function; its sign (+1 or -1) is propagated to the other nodes of the octree. Level set functions are widely used in finite element analysis of two phase flows for example (Sussman et al., 1994). To each interface are then associated an octree and a level set function. All the octrees and their level set functions are merged to form a global octree with a collection of level set functions, one per interface. This global octree forms the basis of the finite element grid (the solve octree or \mathcal{O}_s) on which the Stokes and temperature equations are solved. The values of the level set functions are used to estimate on which side of the surface lies any node of the octree and thus whether an element (a leaf of the octree) is cut by any or several of the surfaces.

To calculate the value of the level set function corresponding to a given interface on the nodes of a given octree, we use the particles representing the surface, their normals and the associated Delaunay triangulation. For each triangle connecting three particles, we first compute the normal to the triangle, (n_x, n_y, n_z) at the centre of the triangle (x_c, y_c, z_c); we then find the leaves of the octree cut by the triangle; for each node, (x_0, y_0, z_0) of each leaf, we call (x_p, y_p, z_p) the orthogonal projection of the node (x_0, y_0, z_0) onto the plane of the triangle; we then consider three cases, depicted in Fig. 7:

1. (x_p, y_p, z_p) falls within the triangle, i.e. it is to the right of the planes defined by each of the sides of the triangle and its normal; in this case the level set function, ϕ_0 , is the signed distance to the plane of the triangle given by

$$\phi = (x_0 - x_c)n_x + (y_0 - y_c)n_y + (z_0 - y_c)n_z \quad (26)$$

2. (x_p, y_p, z_p) lies to the left of one of the edges of the triangle and its orthogonal projection onto that side lies between the two particles defining the edge; in this case the level set function is the distance to the edge, its sign is determined by the dot product of the mean of the normals attached to the two particles and the projection direction;
3. (x_p, y_p, z_p) lies outside at least one of the edges of the triangle and its orthogonal projection onto any of the two sides lies outside the two particles defining any of the two edges; in this case the level set function is the distance to the closest particle defining the triangle and its sign is determined by the dot product of the normal attached to the particle and the projection direction.

Note that in order to define a consistent ‘left’ and ‘right’ side to a plane defined by a side of the triangle and its normal, one needs to be consistent in ordering the particles within each triangle. We also wish to point out that our method differs from those based on updating the level set function nodal values directly by advection of the values computed at the previous time step, such as in so-called ‘fast-marching algorithms’ (Adalsteinsson and Sethian, 1995). In our algorithm, the level set function is not updated but computed from the surface geometry at each time step.

7. Octree refinement

7.1. Grid adaptivity

The solve octree constructed from the union of the individual interface octrees is further refined according to a set of criteria:

- refinement based on the previous time step/iteration solution;
- imposed refinement in a series of rectangular boxes defined by the user;
- imposed refinement on the faces of the unit cube to accurately represent complex boundary conditions for example.

which leads to the creation of the solve octree \mathcal{O}_s , i.e. the octree whose leaves are used to perform the finite element discretization of the equations to be solved.

In what follows we have used functions of the velocity field known on the previous solve octree \mathcal{O}_{ps} (corresponding to the last time step/iteration) to improve the resolution of \mathcal{O}_s . In particular, the second invariant E'_2 of the strain-rate tensor $\dot{\epsilon}$ is measured for each leaf i of \mathcal{O}_{ps} , and E^{\max} is its maximum on the whole octree. Having defined a tolerance τ (typically of the order of a few percents), the center of each leaf i of \mathcal{O}_{ps} that satisfies $(E'_2)_i \geq \tau E_2^{\max}$ defines a location in space where the solve octree is further refined to a prescribed level L .

7.2. Progressive adaptive grid

Were we to use the solution obtained at uniform level (typically 5) to refine the grid to a much larger level (let's say 9) of discretization, the resulting grid would more than likely exceed the memory limit on most computers. Instead the refinement has to be done in a more progressive manner. We use the following algorithm. The solve octree is first initialised at a uniform level L_u (typically 5), so that the octree counts $(2^5)^3 = 32,768$ leaves. According to the refinement criterion presented in the previous paragraph, only a

subset of leaves will be refined to a given level $L > L_u$. The value of L progressively increases one unit at a time to reach the authorised maximum level L_{\max} . This increase takes place when both following statements are true:

- for a given grid the non-linear iterations performed on this grid have converged, i.e. the L_2 -norm of the velocity field difference between two consecutive iterations k and $k + 1$ is less than a given parameter η .

$$\|\{\mathbf{v}\}^{k+1} - \{\mathbf{v}\}^k\|_2 < \eta \quad (27)$$

- the refinement based on \mathcal{O}_{ps} has lead to a solve octree \mathcal{O}_s whose number of leaves N_s is close to the number of leaves N_{ps} of \mathcal{O}_{ps} , i.e.

$$\left| \frac{N_s - N_{ps}}{N_s + N_{ps}} \right| < \chi \quad (28)$$

where χ is a user supplied parameter; if this condition is not fulfilled it means that the octree \mathcal{O} is not yet at equilibrium with the calculated strain rate field; it will be modified accordingly and a further iteration performed.

Schematically, the code structure is built upon three nested loops, as sketched on Fig. 8: the outer one is the timestepping, the second one is the progressive adaptive grid construction, and the inner third one is the non-linear iterations.

8. The free surface

In problems involving a free surface, one of the interfaces (the one representing the free surface) is given special properties. All degrees of freedom corresponding to nodes belonging to elements completely contained in the ‘void’, i.e. the space above the interface, are removed from the equation set to be solved. The finite elements (or cells) that are cut by the free surface are named ‘cut cells’. Material properties for the parts of the cut cells above the free surface are set to values approximating ‘void behaviour’: extremely low viscosity ($10^{-8} \times$ smallest rock viscosity), nil density, and extremely high thermal diffusivity ($10^6 \times$ rock diffusivity). Zero stress boundary conditions are imposed on the top surface of the model.

Furthermore, the geometry of this special interface can be modified to represent erosional processes, i.e. the transport of mass at the surface of the Earth by processes such as fluvial incision, transport and deposition, hill-slope processes and/or ice erosion. At the end of each time step, the position of the particles on the interface corresponding to the free surface is thus modified to take these processes into account. The level set function of the free surface is constructed first and used to adjust the position of the particles on all other interfaces in case they are located above the free surface. This ensures that internal interfaces are also affected by erosion if they are advected towards the free surface by tectonic or erosional processes. Any erosional model can be used to compute the geometry of the free surface; in the current version of the code, the simplest erosion model only has been implemented: a reference level, z_r , is set above which erosion/deposition is instantaneous. We are currently coupling DOUAR to a much more sophisticated surface processes model, CASCADE (Braun and Sambridge, 1997).

9. DivFEM

From the values of the level set functions, the position of each element with respect to each interface is known as well as possible intersections between the element and the interfaces. This information is used to determine the material making up the element,

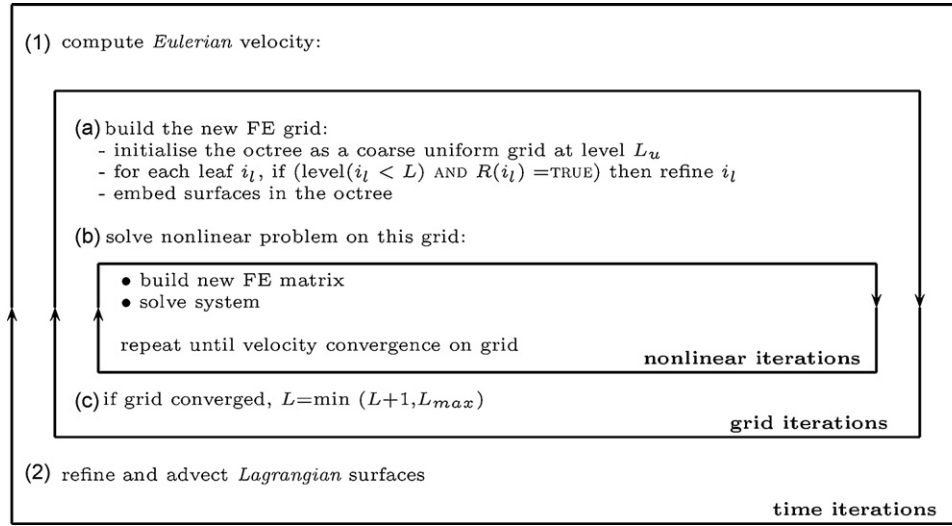


Fig. 8. Schematic overview of DOUAR structure for the mechanical part: the three nested loops are represented, as well as the ALE character of the code.

assuming that interfaces are material boundaries. When an element is intersected by one or several interfaces, the values of the level set functions at the nodes of the elements are used to compute the part or volume of the element that is occupied by each of the materials. These volumes are used to perform the volume integration of the finite element equations (Eq. (15)). We call this procedure the divFEM (division of Finite Element Method).

To determine the volume that is on the positive side of the interface cutting a given element (the cut cell), an octree division of cut cells is performed down to level l . Within each of the resulting leaves (cubes) an 8-point Gauss integration scheme is used to perform the required volume integrals. The level set function is interpolated to the internal nodes and used to determine which part of the volume (positive or negative) each sub-cell belongs to. The relative positive volumes, α , in the remaining cut cells are estimated using the following approximate formula:

$$\alpha = \frac{1}{2} \left(\frac{\sum_{i=1}^8 \phi_i}{\sum_{i=1}^8 \|\phi_i\|} + 1 \right) \quad (29)$$

In those cut cells, material properties are averaged if possible, otherwise the property corresponding to material representing the largest volume in the cut cell is used.

To test the accuracy of the divFEM, we perform simple 2D Rayleigh–Taylor instability tests for which an analytical solution exists (Van Keken, 1993). We computed the growth rate of a small periodic perturbation ($10^{-4} \times$ the vertical size of the problem) of the interface separating two fluids of equal viscosity (1) and differing densities, the lower fluid being lighter than the upper one. The relative error between the analytical solution and that obtained with the divFEM method for increasing levels of octree division within the element is shown in Fig. 9. In practice, we limit the elemental octree division to level $l = 3$, thus minimizing the relative error introduced by divFEM to less than 10^{-3} .

10. Material memory

In many geodynamical problems, we need to track dynamic material properties, i.e. those that are derived from the solution of the equations but need either to be integrated over time, such

as accumulated strain, or simply to be stored for computation of geological observables in a post-processing stage, such as the pressure–temperature–time paths of particles reaching the surface at the end of the model run. This is performed by injecting a large cloud of particles inside the computational domain, the density of which is dynamically adjusted to achieve a balance between accuracy and efficiency.

At every time step, the number of particles in any leaf of the solve octree is imposed to remain between two set limits (commonly 8 and 27). Injection of each new particle is performed by randomly injecting 10 test particles inside the element/leaf among which one only is kept, i.e. the one furthest away from the preexisting particles and the boundaries of the element. Local particle density is also used to perform the removal operation. At the end of each time step, the particles are advected by interpolation of the computed nodal velocities, using the same algorithm as used for the particles defining the interfaces; the strain (second invariant of the strain tensor) is updated by interpolation of the derivatives of the velocity field. For consistency, both interpolations are performed using the finite element interpolation functions.

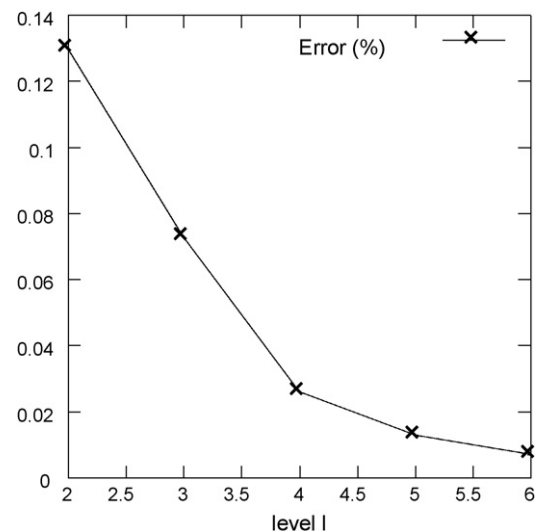


Fig. 9. Error in calculating the growth rate of a Rayleigh–Taylor instability in 2D using the divFEM. The problem was solved on a regular $32 \times 32 \times 32$ mesh. The error decreases strongly with the level of octree division l , within each element.

When new particles are injected, the strain (or any other field carried by the particles) is interpolated from the existing particles onto the nodes of the solve octree and interpolated back onto the new particles. This method may lead to some numerical diffusion which is however limited by the high density of the solve octree leaves (and thus high density of particles) in regions of diverging flow/deformation. Further improvement of this method is however required.

11. Direct solver

The most time consuming part of a large-scale three-dimensional finite element computation is the solution of the large set of coupled algebraic equations generated by the finite element discretization of the partial differential equations and, in the case of non-linear problems, their linearization. This system is represented in matrix notation as $\mathbf{Ax} = \mathbf{b}$ where \mathbf{x} is the solution vector, \mathbf{b} is the right-hand side vector and \mathbf{A} is a matrix containing the coefficients multiplying the unknowns in each of the coupled equations.

It is common practice to use iterative methods based on a spectral acceleration called multigrid to obtain an approximate solution to these equations. Such solvers are tuned to the nature of the set of equations to solve as their convergence rate is affected by the nature and conditioning of the system to be solved or matrix \mathbf{A} to be inverted. For non-linear problems or those characterized by large changes in material properties across interfaces, and, in particular, those characterized by a free surface, convergence may be slow.

A direct solver is based on the factorization of the matrix \mathbf{A} in either of two forms, depending on its symmetry: $\mathbf{A} = \mathbf{L}^T \mathbf{L}$ for symmetric matrices and $\mathbf{A} = \mathbf{LU}$ for non-symmetric matrices where \mathbf{L} is a lower triangular matrix and \mathbf{U} the corresponding upper triangular matrix. After factorization, two successive backsubstitutions are performed to obtain the solution vector: $\mathbf{L}^T \mathbf{y} = \mathbf{b}$ (or $\mathbf{Uy} = \mathbf{b}$ in the case of a non-symmetrical system) and $\mathbf{Lx} = \mathbf{y}$. These require a relatively small number of arithmetic operations compared to the factorization step.

Direct solvers have the following advantages:

1. They can solve ill-conditioned matrices (robustness to ill-conditioning).
2. They can reuse the factorized matrix and apply it to the solutions of multiple right-hand sides.
3. They can be used as black boxes with little or no need for tuning by users.
4. They are versatile and application independent, being based on algebra and graph theory rather than on any specific construction of the system of equations. For example, they can handle broad classes of systems (e.g. symmetric definite positive systems). Any grid or method of connecting equations can be used.
5. They have a high computational intensity and can execute well in hierarchical computer memories. Computational intensity is loosely defined here as the ratio between the time spent to perform arithmetic operations and the time spent to bring data into and out of the registers.

Direct methods have the following drawbacks:

1. They typically need to build the entire matrix of the system, which means that big systems may not fit in memory and not be usable.
2. Memory requirements for the storage of the numerical factor (number of nonzeros in the Cholesky matrix) grow very fast as the number of equations/ grid size increases, especially in 3D.
3. Same observation for the operation count.

4. Their abstraction ignores/sacrifices the specifics of the problem.
5. They are harder to parallelize efficiently on a large number of processors
6. The solution has to be completely recomputed in non-incremental methods (i.e. those requiring the computation of the matrix \mathbf{A} at each iteration, as is the case in DOUAR).

We elected to use a direct solver mostly because these are well suited for highly non-linear problems or those characterized by ill-conditioned matrices. In DOUAR, the ill-conditioning of the matrix arises from the presence of the free surface and the incompressibility. This choice limits us in the size of the problems that can be solved. For instance, our 256GB RAM cluster is limited to the solution of approximately 2,000,000 simultaneous equations.

Two such direct solvers have been used, WSMP (Watson Sparse Matrix Package)¹(Gupta, 2000) and MUMPS (MULTifrontal Massively Parallel sparse direct Solver)²(Amestoy et al., 2001). For reason of ease of access to the source code, we selected MUMPS but did not perform extensive tests to determine which of the two solvers leads to better performances in our problems. We noted however that such solvers make extensive use of basic scalar and parallel algebraic functions (BLAS, BLACS and SCALAPACK) and that their efficiency is thus mostly affected by the quality of the implementation of such low level routines on the computer used for the computations. MUMPS can be used to solve symmetrical and non-symmetrical systems and is thus used in its symmetrical form to solve the Stokes equations and its non-symmetrical form to solve the energy equation.

12. Parallelization

The construction of the finite element matrix is fully parallelizable because all the information required to build the finite element matrix is known at the nodes of the element (such as the level set function values). The advection of the particles used for interface tracking as well as those carrying the material memory is also fully parallelizable. The non-uniform octree discretization limits the number of particles in the 3D cloud which, in turn, allows for its complete storage in all processors. This greatly simplifies the inter-processor communication required to perform operations on the cloud particles (such as their advection).

Calculating the value of the level set functions in the vicinity of the interface they represent is also fully parallelizable as each leaf of the octree cutting a given interface is treated separately and independently. The propagation of the sign of the level set function to the other nodes of the octree cannot be parallelized as it is, by nature, a sequential operation. Consequently, it has to be performed by all processors; this operation can be sped up by performing a series of tests through the entire collection of elements in alternating orders:

```

START = 1 and END = nleaves
BEGIN_LOOP
Do LEAF = START to END
If (any LSF(LEAF) ≠ 0) then
Where LSF(LEAF) = 0 LSF(LEAF) = sign (any non nil LSF(LEAF))
End if
End do
swap START with END
If (LSF (any node) = 0) goto BEGIN_LOOP
FINISH

```

¹ <http://www-users.cs.umn.edu/~agupta/wsmp.html>.

² <http://graal.ens-lyon.fr/MUMPS/>.

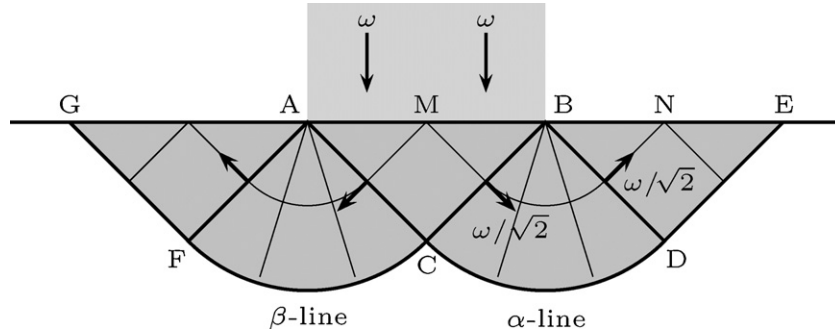


Fig. 10. Prandtl's rigid-plastic solution. The indenter is shown in light grey while the material that has a non-zero velocity is shown in dark grey. α and β slip lines intersect each other at $\pi/2$ angles.

13. Rectangular punch problem

In order to illustrate how the octree refinement algorithm works practically, to justify its implementation in DOUAR and to demonstrate its efficiency, we have carried out numerical experiments of two-dimensional and three-dimensional punches indenting a rigid, perfectly plastic von Mises half-space. The analytical solution to the two-dimensional problem is to be found in many textbooks (Hill, 1950; Kachanov, 2004 or Freudenthal and Geiringer, 1958) for a more mathematical approach.

Moreover, since the late 1970s, a simple analogy has been made between the tectonics of Asia and the deformation of a rigidly indented rigid-plastic solid: India is analogous to the indenter and

the great strike-slip faults, such as the Kunlun Fault System, correspond to slip lines (Molnar and Tapponier, 1975). This analogy has been the subject of physical (Davy and Cobbold, 1988) and numerical modelling (Houseman and England, 1993, among many others) and has given rise to an abundant literature. More recently, GPS data seem to indicate that the velocity measurements fit to a certain degree an indenter type of velocity field (Zhang et al., 2004).

In the two-dimensional case, the analytical solution is known, and the slip-line solution is shown on Fig. 10. The slip lines consist of a curvilinear mesh of two families of lines, which always cross each other at right angles. The velocity distribution and stress state can be determined from the geometry of these lines. An undeformed wedge of material forms an active so-called "Rankine zone"

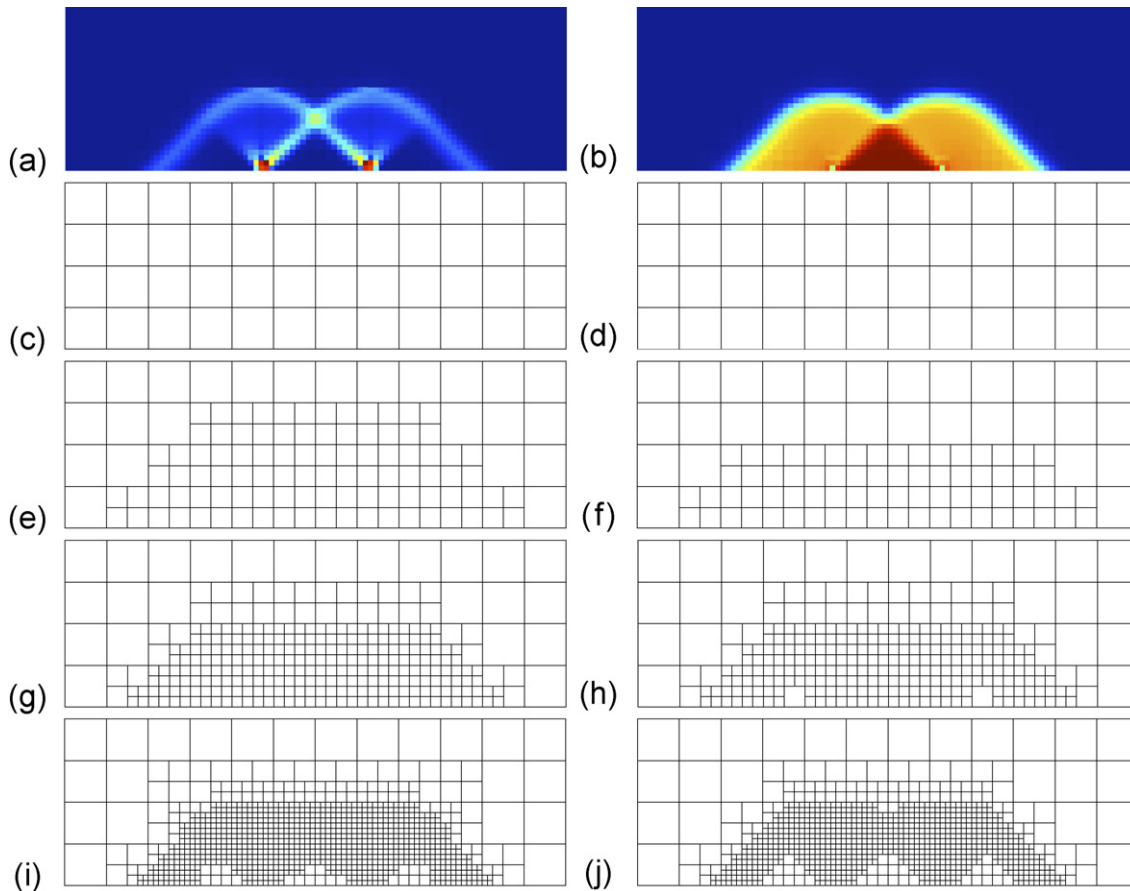


Fig. 11. Cross-section of the octree at $x = 0.5$ and for $z < 0.2$. (a–b) E_2' field and velocity norm field computed on finest grid; (c–d) first and last generated grid at level 5; (e–f) first and last generated grid at level 6; (g–h) at level 7; (i–j) at level 8.

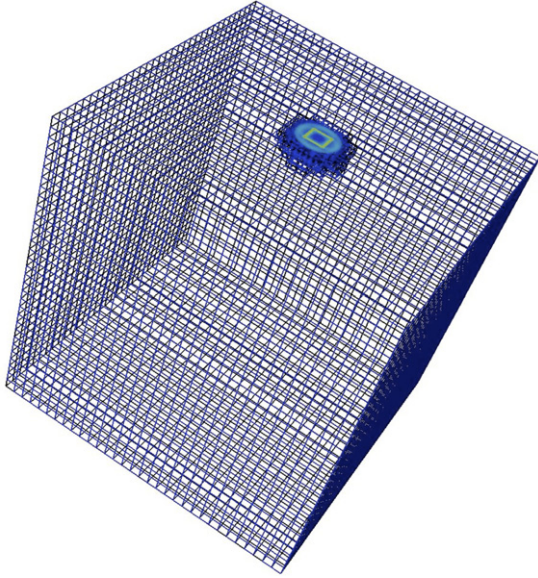


Fig. 12. Final octree ($L_u = 5$, $L_{\max} = 9$) generated by DOUAR for the square punch computations. Colour (from blue to red) indicates E'_2 . (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

below the punch with angles $\pi/4$. This wedge pushes material outwards, causing passive Rankine zones to form with angles $\pi/4$. The transition zones are circular.

In order to carry out two-dimensional simulations with our 3D code, we have set no-slip boundary conditions on the walls of the cube so that no flux of material is allowed outside the unit cube, and we have imposed a velocity $\mathbf{v} = (0, 0, -w)$ for nodes whose coordinates (x, y, z) verify $x \in [0 : 1]$, $y \in [-\Delta y/2, \Delta y/2]$ and $z = 1$. In fact, this corresponds to replicating the 2D punch problem (plane Oyz) in the third dimension (along Ox).

The (dimensionless) parameters used to run the simulations are: gravity $g = 0$, punch width $\Delta y = 0.08$, viscosity $\mu_0 = 10^4$, imposed velocity $w = 1.0$, penalty $\lambda = 10^8$, refinement ratio $\tau = 0.06$, von Mises yield $\sigma_0 = 1$, convergence criterions $\eta = 10^{-5}$ (Eq. (27)), and $\chi = 0.025$ (Eq. (28)). In the absence of gravity, the value of the density ρ is meaningless.

In Fig. 11a and b are presented the solutions obtained on the final grid of maximum level $L_{\max} = 8$. One sees that the code has captured the slip-lines reasonably well: the lightblue colour indicates regions where E'_2 is maximal, the velocity norm shows a rigid wedge and two regions on each side of constant velocity, and the velocity field displays three regions of apparent rigid movement and two of rotation, as expected.

In Fig. 11c–j are shown the succession of increasing level grids that were built in order to reach the final grid. Fig. 11c represents the portion of the initial uniform grid of interest. The solution is first computed on this grid, and is used to refine a new grid down to level 6 (Fig. 11d). Once the solution is obtained on this grid, and as long as the refinement based on this solution leads to an octree that does not comply with the C_2 criterion, the process is iterated, until we reach a stable grid (Fig. 11e). It is then used to generate a level 7 grid (Fig. 11f). After some grid iterations, the algorithm carries on to level $L_{\max} = 8$ (Fig. 11i and j).

It should be that this algorithm allows for grids to evolve to a given maximum level of refinement, and that, given the set of refinement parameters input by the user, it computes the best/smallest grid satisfying the conditions, hence ensuring that no memory is wasted. The transition from level 7 to level 8 grids is a good illustration of this process: the refinement criterion based on the velocity field computed on grid (h) overestimates the grid structure (i) that ultimately evolves into (j). Similar qualitative results have already been obtained previously by Zienkiewicz et al. (1995) with mainly adaptive triangular meshes.

While these simulations prove that the code captures the physics of such a classical problem, they are quite heavy in terms of cpu time and memory usage because of the redundant nature of the setup (we are in fact performing what others may call a pseudo-2D punch). No exact solution exists for the three-dimensional square punch, even in the simple case of an isotropic homogeneous weightless plastic material. Many authors have tackled this problem numerically since it is encountered in bearing capacity calculations which are an important part of the design of foundations (Taiebat and Carter, 2000, 2002; Salgado et al., 2004; Gourvenec and Randolph, 2003; Gourvenec et al., 2006). However, it is important to notice that most of these three-dimensional experiments have been performed on carefully chosen grids, giving rise to different sets of tailored grids for square, rectangular and circular footings.

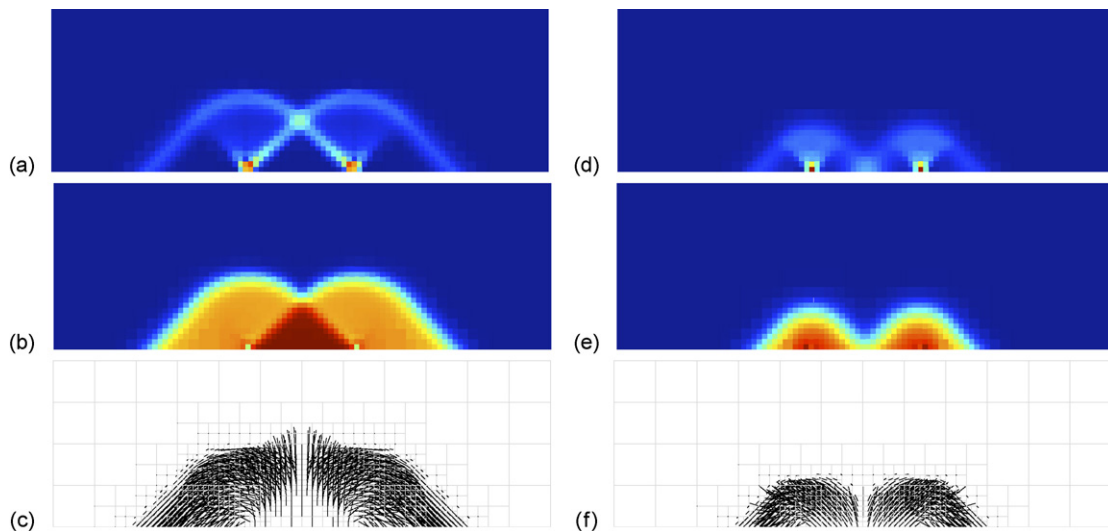


Fig. 13. Cross-section of the octree at $x = 0.5$ and for $z < 0.2$. (a and d) E'_2 , (b and e) velocity norm and (c and f) velocity direction. (a–c) Rough interface; (d–f) smooth interface. See comments in the text for further description of these results.

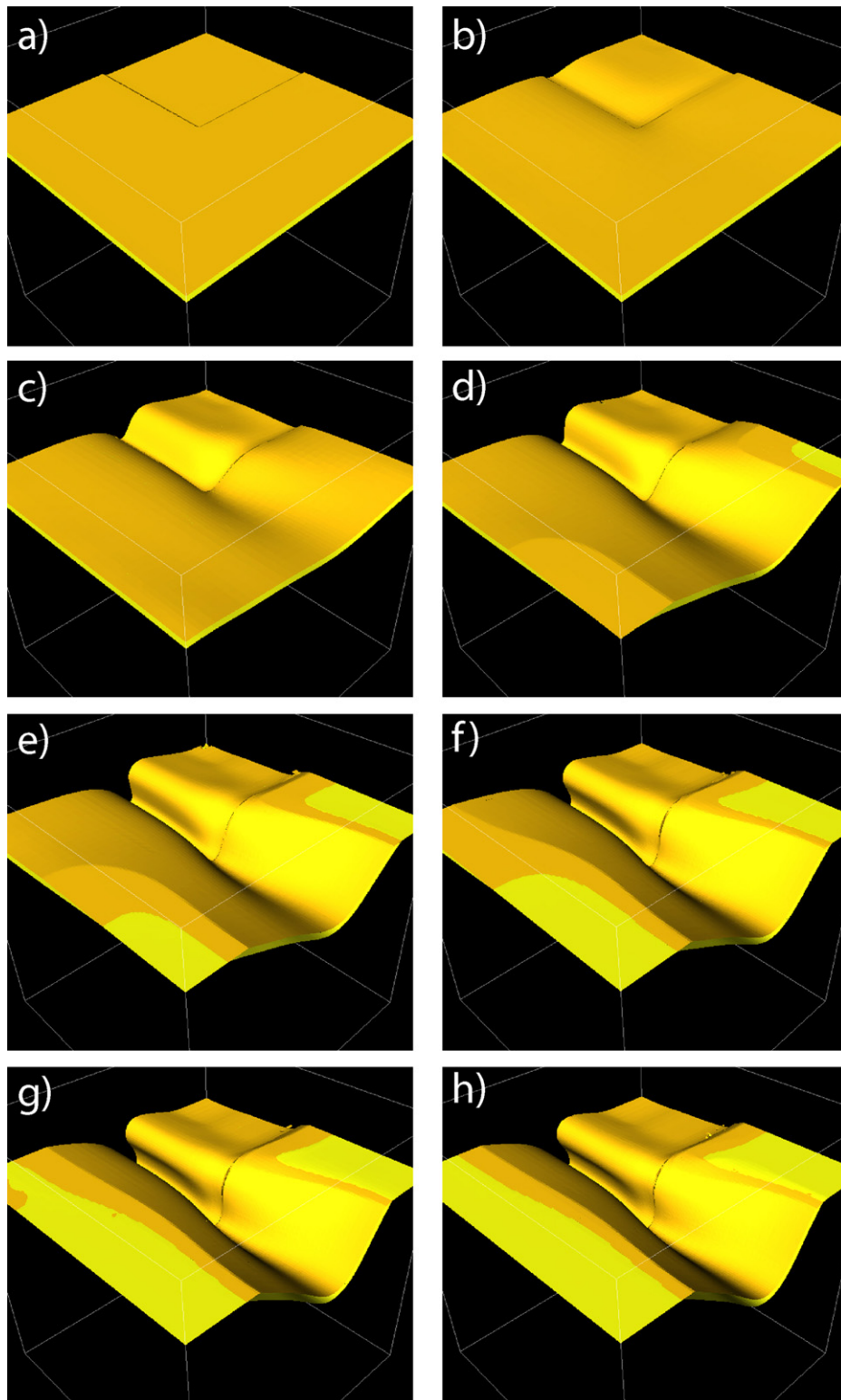


Fig. 14. Results of a 3D folding experiment in a material subjected to surface erosion/sedimentation. (a–h) Equally spaced time steps in the evolution of the system. The folding layer is progressively exhumed at both ends of the model leading to change in shortening mechanism from folding and growth of a mechanical instability (a–e) to simple pure-shear shortening (f–h). The free surface is not represented but corresponds to regions where the two surfaces of the folding layer are intersected by an horizontal surface.

In Fig. 12 is shown the full octree at $L_{\max} = 9$, in the case of a square punch. One clearly sees that the structure is highly refined close to the punch area, while a vast proportion of the unit cube is remained at the uniform level $L_u = 5$.

Using the same simulation parameters as in the two-dimensional case, we have performed two square punch experiments. The first one is usually called *smooth* as only the z -component of the velocity is imposed on the nodes under

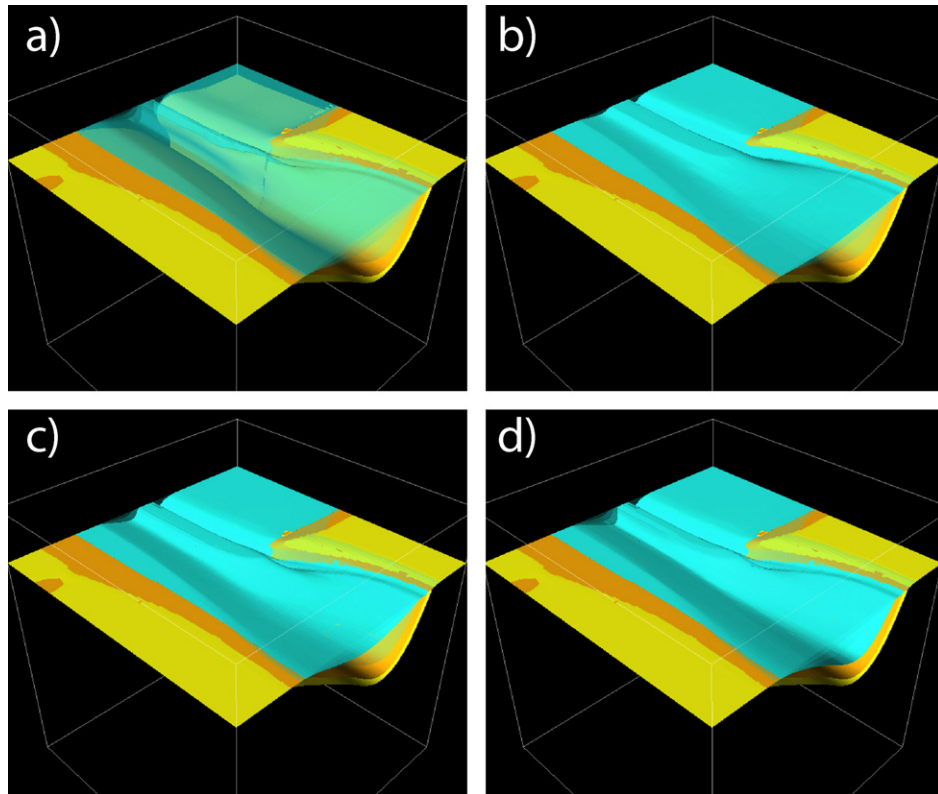


Fig. 15. Final stratigraphy of the basin formed during the folding experiment described in Fig. 14. (a) The three stratigraphic markers are shown as semi-transparent cyan surfaces. (b–d) Each of the three stratigraphic markers is shown as an opaque cyan surface.

the punch, while the second one is called *rough* since all three components are assigned (the x - and y -components are set to zero).

In both cases, the boundary conditions imposes fourfold symmetry, with displacements orthogonal to the edges but also symmetric about the diagonals. In Fig. 13a and b are shown the material displacement (blue colour gradient) as well as the velocity field on a plane perpendicular to the edges and passing through the center of the square.

As observed already by Gourvenec et al. (2006), a double-wedge Hill-type deformation pattern is observed in the smooth case, while a simple-wedge deformation pattern appears in the rough case. The obtained velocity fields clearly indicate the different direction of material displacement below the punch. The material displacement is also much shallower in the smooth case than in the rough case, while its extent on the sides of the punch is larger in the latter case.

14. Folding experiments with a free eroding surface

To demonstrate the surface tracking capabilities of DOUAR, we have performed a folding experiment in which a thin layer of non-linear ($n = 3$) viscous material is embedded in another, less viscous, non-linear fluid. The thickness of the layer ($h = 0.02$) and the viscosity ratio ($r = 192$) are chosen such that a folding instability develops at a wavelength ($\lambda \approx 0.4$) that is fully contained within the unit box in which the experiment is performed. Similar experiments have already been performed in three dimensions to investigate the importance of an imposed velocity in a direction perpendicular to the shortening (folding) direction (Kaus and Schmalholz, 2006). Here, the system is further characterized by a free surface located at distance $\Delta z = 0.05$ from the top of the competent layer and subjected to very efficient erosion, i.e. regions of positive topography are instantaneously eroded and the resulting

mass is instantaneously and uniformly deposited in the regions of negative topography simulating surface processes and assuming mass conservation. An initial small perturbation in the layer thickness is imposed in the upper quarter of the model (x and $y < 0.5$) to force the folding to initiate near the center of the model as well as to introduce an asymmetry in the system and produce a non-cylindrical fold. The evolution of the model is shown in Fig. 14.

The non-linear viscosity leads to the formation of a single, asymmetrical fold which grows rapidly and leads to the exhumation of the folding layer, first in the right corner of the experiment (as seen from the reader's point of view), then in the front corner. Further shortening leads to complete erosion of the folding layer along the left boundary of the model and the concomitant filling of the depression created by the down-going limb of the fold near the center of the model. The folding instability arises from the presence of the more competent layer; as deformation progresses and the layer is exhumed, the instability stops (Fig. 14e) and shortening is accommodated by pure shear of the entire model leading to a progressive tightening of the fold (Fig. 14f–h).

Tracking interfaces can also be used to compute the stratigraphy within the evolving basin by injecting interfaces at regular time intervals that have the same geometry as the free surface. These surfaces are then passively advected like any other interface and eroded with the free surface, if necessary. The resulting stratigraphy is shown in Fig. 15. In the bottom part of the basin (closest to the reader), a broad depression has formed and evolved monotonically with time. In the upper part of the basin (furthest from the reader) the situation is drastically different: the depression is much narrower and the shortening has led to the formation of a central ridge separating two sub-basins. This asymmetry results from the initial perturbation introduced in the model and the non-linear viscosity of the viscous material involved which has led

to the localization of the deformation in the upper part of the model.

15. Discussion and conclusions

We have presented here a new three-dimensional finite element code for solving Stokes equations in the context of geological problems that are characterized by a free surface. The design of this tool was driven by the need to track complex interfaces as well as material properties. Efficiency is achieved through the use of a regular but non-uniform octree division of space and a direct solver. We have shown the results of some computations for problems which either have an analytical solution against which the accuracy and resolution of the new code can be tested or which demonstrate the flexibility and suitability of the code to be used to solve interesting geological problems.

Further development and testing is however required, as well as improvements to algorithms that have already been implemented in the code. In particular, the following are required

- A more robust algorithm to triangulate the particles on curved interfaces; the current method is efficient and adapted to many problems but breaks down where the surface geometry is complex, i.e. characterized by extreme curvature. Note that the triangulation of the particles on the surface is only required to compute the value of the level set function associated with the surface onto the nodes of the octree attached to the surface. We have already tested meshless methods that compute the level set function values from the position of and normal attached to a finite (reduced) set of particles. This requires solving a local minimization problem to find, for example, the best fitting quadratic surface representing the geometry of the interface.
- A more precise algorithm for the advection of the particles that would allow the use of longer and thus fewer time steps;
- An implicit or semi-implicit algorithm for the advection of the free surface. In problems characterized by a free surface, the time step is limited by the stresses generated by the deformation of the free surface in comparison to the viscous stresses produced in the fluid by the deformation/flow. In cases where the internal density differences driving the flow are much smaller than the density difference across the free surface, this restriction may lead to small steps that are impractical. This implies, in turn, that the size of the problems characterized by a free surface is currently limited.

Finally, further work is also required to improve the predictive capabilities of DOUAR, and, in particular, to use the accurate P–T and deformation paths derived from the tracking of Lagrangian, material particles, to produce estimates of thermochronological ages, metamorphic grade distribution, integrated strain patterns, or other geological observables.

Yet, as we have shown in this paper, DOUAR currently represents an efficient and reasonably accurate method to solve various large-scale problems, as presented here, and holds promise for the solution of more demanding fully 3D thermo-mechanical problems in geodynamics.

Acknowledgments

The work described in this paper has been supported by an ARC (Australian Research Council) Discovery grant. C. Thieulot was supported by a postdoctoral fellowship from the Canadian Institute for Advanced Research. Computations were performed on a cluster co-financed by Rennes-Metropole, the Centre National de la Recherche Scientifique (CNRS), the Agence Nationale de la Recherche (ANR)

and a Marie Curie International Reintegration Grant of the European Union to J. Braun. Dalhousie University authors were funded in part by an Atlantic Innovation Fund contract and preliminary calculations at Dalhousie University were supported by CFI and IBM-SUR grants.

References

- Adalsteinsson, D., Sethian, J., 1995. A fast level set method for propagating interfaces. *J. Comput. Phys.* 118, 269–277.
- Albers, M., 2000. A local mesh refinement multigrid method for 3-d convection problems with strongly variable viscosity. *J. Comput. Phys.* 160, 126–150.
- Amestoy, P., Duff, I., Koster, J., L'Excellent, A., 2001. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.* 23 (1), 15–41.
- Bathe, K.-J., 1982. *Finite Element Procedures in Engineering Analysis*, first ed. Prentice-Hall, Englewood Cliffs, NJ.
- Batt, G., Braun, J., 1997. On the thermomechanical evolution of compressional orogens. *Geophys. J. Int.* 128, 364–382.
- Beaumont, C., Jamieson, R., Nguyen, M., Lee, B., 2001. Himalayan tectonics explained by extrusion of a low-viscosity crustal channel coupled to focused surface denudation. *Nature* 414, 738–742.
- Braun, J., 1993. Three-dimensional numerical modelling of compressional orogens: thrust geometry and oblique convergence. *Geology* 21, 153–156.
- Braun, J., 1994. Three-dimensional numerical simulations of crustal-scale wrenching using a non-linear failure criterion. *J. Struct. Geol.* 16, 1173–1186.
- Braun, J., 2006. Recent advances and current problems in modelling surface processes and their interactions with crustal deformation. In: Buitert, S., Schreurs, G. (Eds.), *Analogue and Numerical Modelling of Crustal-scale Processes*, Volume 253 of Special Publications. The Geological Society of London, London, pp. 307–325.
- Braun, J., Beaumont, C., 1995. Three-dimensional numerical experiments of strain partitioning at oblique plate boundaries: implications for contrasting tectonic styles in the southern Coast Ranges, California and central South Island, New Zealand. *J. Geophys. Res.* 100, 18,059–18,074.
- Braun, J., Sambridge, M., 1994. Dynamical Lagrangian Remeshing (DLR): a new algorithm for solving large strain deformation problems and its application to fault-propagation folding. *Earth Planet. Sci. Lett.* 124, 211–220.
- Braun, J., Sambridge, M., 1997. Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization. *Basin Res.* 9, 27–52.
- Cheng, J., Finnigan, P., Hathaway, A., Kela, A., Schroeder, W., 1986. Quadtree/Octree Meshing with Adaptive Analysis. In: Sengupta, S., Hauser, J., Eisman, P., Thompson, J. (Eds.), *Numerical Grid Generation in Computational Fluid Mechanics* 88. Pineridge Press, Swansea, UK, pp. 633–642.
- Davy, P., Cobbold, P., 1988. Indentation tectonics in nature and experiment. 1: Experiments scaled for gravity. *Bull. Geol. Inst. Uppsala* 14, 129–141.
- Enright, D., Losasso, F., Fedkiw, R., 2005. A fast and accurate semi-Lagrangian particle level set method. *Comput. Struct.* 83, 479–490.
- Freudenthal, A., Geiringer, H., 1958. The mathematical theory of the inelastic continuum. In: *Handbuch der Physik, Encyclopedia of Physics*, Volume VI. Springer-Verlag.
- Fullsack, P., 1995. An arbitrary lagrangian-eulerian formulation for creeping flows and its application in tectonic models. *Geophys. J. Int.* 120, 1–23.
- Gourvenec, S., Randolph, M., 2003. Effect of strength non-homogeneity on the shape of failure envelopes for combined loading of strip and circular foundations on clay. *Géotechnique* 53, 575–586.
- Gourvenec, S., Randolph, M., Kingsnorth, O., 2006. Undrained bearing capacity of square and rectangular footings. *Int. J. Geomech.* 6, 147–157.
- Gupta, A., 2000. WSMP: Watson sparse matrix package (Part-II: direct solution of general sparse systems). Technical Report RC 21888 (98472), IBM T.J. Watson Research Center, Yorktown Heights, NY.
- Hassani, R., Jongmans, D., Chery, J., 1997. Study of plate deformation and stress in subduction processes using two-dimensional numerical models. *J. Geophys. Res.* 102, 17951–17965.
- Hill, R., 1950. *The Mathematical Theory of Plasticity*. Oxford University Press.
- Houseman, G., 1988. The dependence of convection planform on mode of heating. *Nature* 332, 346–349.
- Houseman, G., England, P., 1993. Crustal thickening versus lateral expulsion in the Indian–Asian continental collision. *J. Geophys. Res.* 98, 12,333–12,349.
- Hughes, T., Brooks, A., 1982. A theoretical framework for Petrov–Galerkin methods with discontinuous weighting functions: applications to the streamline-upwind procedure. In: Gallagher, R., Norrie, D.H., Oden, J., Zienkiewicz, O. (Eds.), *Finite Elements in Fluids*. John Wiley & Sons, New York, pp. 47–65.
- Hughes, T., Liu, W., Books, A., 1979. Finite element analysis of incompressible flows by the penalty function formulation. *J. Comput. Phys.* 30, 1–60.
- Huisman, R., Beaumont, C., 2002. Asymmetric lithospheric extension: the role of frictional plastic strain softening inferred from numerical experiments. *Geology* 30 (3), 211–214.
- Huisman, R., Buitert, S., Beaumont, C., 2005. Effect of plastic-viscous layering and strain softening on mode selection during lithospheric extension. *J. Geophys. Res.* 110 (B02406), 2004JB003114.

- Kachanov, L., 2004. *Fundamentals of the Theory of Plasticity*. Dover Publications, Inc.
- Kaus, B., Schmalholz, S., 2006. 3D finite amplitude folding: implications for stress evolution during crustal and lithospheric deformation. *Geophys. Res. Lett.* 33 (doi:10.1029/2006GL026341).
- Molnar, P., Tapponier, P., 1975. Cenozoic tectonics of Asia: effects of a continental collision. *Science* 189, 419–426.
- Moresi, L., Solomatov, V., 1998. Mantle convection with a brittle lithosphere: thoughts on the global tectonics styles of the Earth and Venus. *Geophys. J. Int.* 133, 669–682.
- Poliakov, A., Podladchikov, Y., Talbot, C., 1993. Initiation of salt diapirs with frictional overburden: numerical experiments. *Tectonophysics* 228, 199–210.
- Press, W., Teukolky, S., Vetterling, W., Flannery, B., 1992. *Numerical Recipes, The Art of Scientific Computing*, first Ed. Cambridge University Press, Cambridge, England.
- Pritchard, C., 2005. A cyclical property of cyclic polygons. *Math. School* 34 (5), 14.
- Salgado, R., Lyamin, A., Sloan, S., Yu, H., 2004. Two- and three- dimensional bearing capacity of foundations in clay. *Géotechnique* 54, 297–306.
- Sambridge, M., Braun, J., McQueen, H., 1995. Geophysical parameterization and interpolation of irregular data using natural neighbours. *Geophys. J. Int.* 122, 837–857.
- Samet, H., 1989. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley.
- Sloan, S., 1989. A FORTRAN program for profile and wavefront reduction. *Int. J. Num. Methods Eng.* 28, 2651–2679.
- Stolar, D., Roe, G., Willett, S., (GSA special publication) 2005. Evolution of a critical orogen under various forcing scenarios: findings from a numerical sandbox. In: Willett, S., Hovius, N., Brandon, M., Fisher, D. (Eds.), *Tectonics, Climate, and Landscape Evolution*.
- Sussman, M., Smereka, P., Osher, S., 1994. A level set approach for computing solutions to incompressible two-phase flows. *J. Comput. Phys.* 114, 146–159.
- Tackley, P., 1998. Self-consistent generation of tectonic plates in three-dimensional mantle convection. *Earth Planet. Sci. Lett.* 157, 9–22.
- Tackley, P., 2000a. Self-consistent generation of tectonic plates in time-dependent, three-dimensional mantle convection. 1. Pseudoplastic yielding. *Geochem. Geophys. Geosyst.* 1, 2000GC000036.
- Tackley, P., 2000b. Self-consistent generation of tectonic plates in time-dependent, three-dimensional mantle convection. 2. Strain weakening and asthenosphere. *Geochem. Geophys. Geosyst.* 1, 2000GC000043.
- Taiebat, H., Carter, J., 2000. Numerical studies of bearing capacity of shallow foundations on cohesive soil subject to combined loading. *Géotechnique* 50, 409–418.
- Taiebat, H., Carter, J., 2002. Bearing capacity of strip and circular foundations on undrained clay subjected to eccentric loads. *Géotechnique* 52, 61–64.
- Van Keken, P., 1993. Numerical modelling of thermochemically driven fluid flow with non-Newtonian rheology: applied to the Earth's lithosphere and mantle. Ph.D. Thesis. Faculteit Aardwetenschappen der Rijksuniversiteit, Utrecht.
- Webb, J., 1990. Imposing linear constraints in finite-element analysis. *Comm. Appl. Num. Methods* 6 (6), 471–475.
- Willett, S., Beaumont, C., Fullsack, P., 1993. Mechanical model for the tectonics of doubly-vergent compressional orogens. *Geology* 21, 371–374.
- Zhang, P.-Z., Shen, Z., Wang, M., Gan, W., Bürgman, R., Molnar, P., Wang, Q., Niu, Z., Sun, J., Wu, J., Hanrong, S., Xinzhaio, Y., 2004. Continuous deformation of the tibetan plateau from global positioning system data. *Geology* 32, 809–812.
- Zienkiewicz, O., Pastor, M., Huang, M., 1995. Softening, localisation and adaptive remeshing. capture of discontinuous solutions. *Comput. Mech.* 17, 98–106.