Generation of unstructured meshes in 2-D, 3-D, and spherical geometries with embedded high-resolution sub-regions

Jorge M. Taramón[1], Jason P. Morgan[1,2], Chao Shi[2], Jörg Hasenclever[3]

[1]Department of Earth Sciences, Royal Holloway University of London, Egham, Surrey, UK

[2]EAS Dept., Cornell University, Itaca, NY, USA

[3]Institute of Geophysics, Hamburg University, Hamburg, Germany

JMT and JPM designed the research. JMT, CS, JPM and JH programmed the 2-D rectangular mesh generator. JMT programmed the 2-D cylindrical mesh generator and 3-D spherical mesh generator in discussion with JPM and JH. JMT designed the tests and analysed the results in discussion with JPM and JH. JMT wrote the manuscript in collaboration with JPM and JH.

Corresponding author: Jorge M. Taramón, `jorge.taramongomez.2014@live.rhul.ac.uk`

Abstract

We present 2-D, 3-D, and spherical mesh generators for triangular and tetrahedral elements. The mesh nodes are treated as if they were linked by virtual springs that obey Hooke's law. Given the desired lengths for the springs, a finite element problem is solved for optimal (static equilibrium) nodal positions. A 'guide-mesh' approach allows the user to define embedded high-resolution sub-regions within a coarser mesh. The method converges rapidly. For example, the algorithm is able to refine within a few iterations a specific region embedded in an unstructured tetrahedral spherical shell so that the edge-length factor $l_{0r}/l_{0c} = 1/33$ where l_{0r} and l_{0c} are the desired spring length for elements inside the refined and coarse regions respectively. The algorithm also includes routines to locally improve the quality of the mesh and to avoid ill-shaped 'sliver-like' tetrahedra. We include a geodynamic modelling example as a direct application of the mesh generator.

1 Introduction

Mesh generation and (adaptive) refinement are essential ingredients for computational modelling in science and industry. During modelling, low-quality meshes can potentially lead to larger numerical approximation errors. A high-quality mesh would consist of elements with aspect ratios near 1, i.e. with similar edge lengths. There are three main techniques to generate meshes: (1) The advancing front method [*Löhner and Parikh*, 1988; *Schöberl*, 1997; *Choi et al.*, 2003; *Ito et al.*, 2004] starts from the boundary of the domain. New elements are created one-by-one from an existing front of elements towards the interior until the region is filled. This method generally creates high-quality meshes close to the domain boundaries but can have difficulties in regions where advancing fronts merge. (2) Octree-based methods [*Mitchell and Vavasis*, 1992; *Labelle and Shewchuk*, 2007; *Ito et al.*, 2009] produce graded meshes through recursive subdivision of the domain. The simplicity of these methods makes them very efficient. However, poorly shaped elements can form near region boundaries. (3) Delaunay Triangulation ensures that the circumcircle/circumsphere associated to each triangle/tetrahedron does not contain any other point in its interior. This feature makes Delaunay-based methods [*Chew*, 1989; *Ruppert*, 1995; *Chew*, 1997; *Shewchuk*, 1998] robust and efficient. However, in 3-D they can generate very poorly shaped 'sliver' tetrahedra with four almost coplanar vertex nodes and a near zero volume. Several techniques to remove slivers have been proposed [*Cheng et al.*, 2000; *Li and Teng*, 2001; *Cheng and Dey*, 2002], although near boundaries slivers can typically persist [*Edelsbrunner and Guoy*, 2002]. Any 'good' mesh should meet the following requirements [*Bern et al.*, 1994]: (1) It conforms to the boundary; (2) It is fine enough in those regions where the problem to be solved demands higher accuracy; (3) Its total number of elements is as small as possible to reduce the computational costs to solve the problem; (4) It has well-shaped elements that improve the performance of iterative methods.

Current mesh-generation algorithms for engineering such as Netgen [*Schöberl*, 1997], GiD (`https://www.gidhome.com`) or TetGen [*Si*, 2015] are based on the above methods. Variational methods [*Alliez et al.*, 2005] rely on energy minimization to optimize the mesh during the generation procedure. A widely used open access community-code for 2-D mesh generation is Triangle [*Shewchuk*, 1996]. DistMesh [*Persson and Strang*, 2004] is a spring-based method that allows the user to create 2D and 3D unstructured meshes based on the distance from any point to the boundary of the domain. However, this algorithm is typically slow.

Frequently used mesh generators in 3-D geodynamic problems are the ones included in the ASPECT [*Kronbichler et al.*, 2012], Rhea [*Burstedde et al.*, 2008] and Fluidity [*Davies et al.*, 2011] codes. ASPECT and Rhea are written in C++ with adaptive mesh refinement (AMR). However, their regular hexahedral elements create so-called "hanging nodes" in regions where the resolution changes and cannot be directly applied to create well-formed tetrahedral elements. Fluidity is another example of AMR for tetrahedral meshes, with very limited mesh-generation capabilities.

Here we present a new unstructured mesh generator that is based on a finite element implementation of the DistMesh approach, using virtual springs between nodes and solving for the static equilibrium positions of the nodes. This makes it considerably faster than the DistMesh algorithm. The user can create tetrahedral meshes without hanging nodes and also create embedded high-resolution sub-regions within a much coarser mesh. Throughout the algorithm, MATLAB's (`http://www.mathworks.com`) 'delaunay' function is called to generate the spring connectivity matrix that relates nodes to triangles or tetrahedra. We have also developed and tested techniques for adding or rejecting nodes in regions where mesh resolution is too high or too low respectively. Smooth variations in element size between high-resolution and low-resolution regions are achieved by using a guide-mesh defining the desired resolution in space. It defines the mesh sizing function to be a linearly varying function over its element size. The mesh-generation code is written in vectorized MATLAB. We will show that its speed is faster than competing compiled algorithms.

The motivation to build this computational tool was twofold: (1) We wished to perform numerical experiments on 3-D spherical shell meshes with embedded high-resolution regions, and found no available open source code that could readily and efficiently construct this type of mesh; (2) We wished to add the capability to adaptively remesh high-resolution regions during a time-dependent solution, and wanted an easy-to-modify mesh-generation code that could be further customized for this task.

We present this approach by showing its methodology for rectangular, cylindrical annulus, and spherical shell meshes (Section 2). In Section 3 we show the results for these meshes. Section 4 presents a geodynamic modelling example where a 3-D spherical shell mesh is generated. In Section 5 we compare the performance of our algorithm to other algorithms.

2 Methods

2.1 Spring-based solver

Inspired by *Persson and Strang* [2004], we treat mesh points as the locations of finite element nodes linked by virtual elastic springs. Spring length is used to define the desired nodal distance within any mesh region. Nodal positions are solved for so that the global network of virtual springs is in static equilibrium. The flowchart of the entire algorithm is presented later in Section 2.5 and Figure 9.

The behaviour of each fictitious spring is described by Hooke's law

$$F = -k\delta s\,, \tag{1}$$

where F is the force acting at each end of spring, k is its stiffness, and δs is the distance the spring is stretched or compressed from its equilibrium length l_0. Forces and nodal positions are expressed in x, y coordinates in 2-D (Figure 1a). Because Hooke's law is formulated along the spring direction, it is necessary to introduce the X' axis as the local 1-D reference system. Hooke's law for each spring in its local 1-D reference system is given by

$$\begin{aligned} f_1{}' &= k\delta s = k(x_2{}' - x_1{}' - l_0)\,, && (2a)\\ f_2{}' &= -k\delta s = -k(x_2{}' - x_1{}' - l_0)\,, && (2b) \end{aligned}$$

where f' and x' are the force and position of the ends of the spring (subscripts 1 and 2). Writing equations (2a) and (2b) in matrix form, and moving the force terms to the left-hand side yields

$$\begin{pmatrix} f_1{}' \\ f_2{}' \end{pmatrix} + k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 0 \\ l_0 \end{pmatrix} = k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} x_1{}' \\ x_2{}' \end{pmatrix}. \tag{3}$$

To solve for the nodal positions in 2-D, we change from local coordinates ($x_1{}'$, 0; $x_2{}'$, 0) to global coordinates (x_1, y_1; x_2, y_2). This change of coordinates is described in matrix form as

$$\boldsymbol{R_{2D}} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ 0 & 0 & \cos\alpha & \sin\alpha \end{bmatrix}, \tag{4}$$

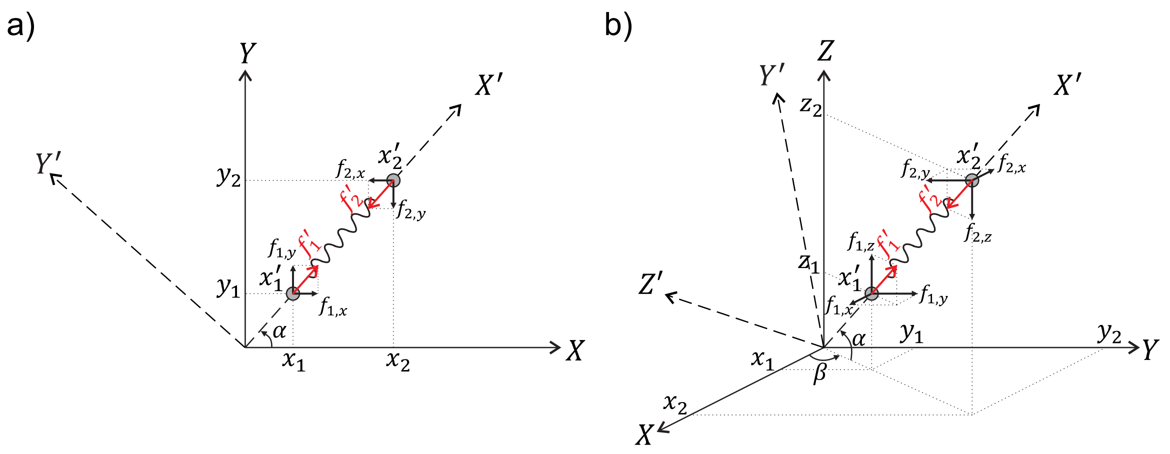

Figure 1: (a) Virtual spring in the 2-D space. The global (X, Y) and local (X', Y') reference systems are shown. (b) Virtual spring in the 3-D space. The global (X, Y, Z) and local (X', Y', Z') reference systems are shown. Grey dots represent two nodes linked by the virtual spring. Red arrows represent the forces acting at each end of the spring.

where α is the angle of the X' axis measured from the X axis in the counterclockwise direction (Figure 1a). Applying equation (4) to equation (3) (see Appendix A: for further details), equation (3) becomes

$$k \begin{bmatrix} -c_\alpha{}^2 & -s_\alpha c_\alpha & c_\alpha{}^2 & s_\alpha c_\alpha \\ -s_\alpha c_\alpha & -s_\alpha{}^2 & s_\alpha c_\alpha & s_\alpha{}^2 \\ c_\alpha{}^2 & s_\alpha c_\alpha & -c_\alpha{}^2 & -s_\alpha c_\alpha \\ s_\alpha c_\alpha & s_\alpha{}^2 & -s_\alpha c_\alpha & -s_\alpha{}^2 \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_{1,x} \\ f_{1,y} \\ f_{2,x} \\ f_{2,y} \end{pmatrix} + kl_0 \begin{pmatrix} c_\alpha \\ s_\alpha \\ -c_\alpha \\ -s_\alpha \end{pmatrix}, \tag{5}$$

where $s_\alpha \equiv \sin\alpha$ and $c_\alpha \equiv \cos\alpha$. Equation (5) can be written in the matrix form as

$$\boldsymbol{Kx} = \boldsymbol{f_r} + \boldsymbol{f_{l_0}}, \tag{6}$$

where $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{x}$ is the nodal displacement vector, $\boldsymbol{f_r}$ is the residual force and $\boldsymbol{f_{l_0}}$ is the force-term created by the fact that the springs would have zero-force at their desired length. In equilibrium, $\boldsymbol{f_r} = 0$. The solution is the 'optimal' position of each node obtained by solving

$$\boldsymbol{x} = \boldsymbol{K}^{-1}\boldsymbol{f_{l_0}}. \tag{7}$$

In 3-D, forces and nodal positions are expressed in x, y and z coordinates (Figure 1b). Here, a change from local coordinates ($x_1{}'$, 0, 0; $x_2{}'$, 0, 0) to global coordinates (x_1, y_1, z_1; x_2, y_2, z_2) is needed. This change consists of the 3-D rotation described by

$$\boldsymbol{R_{3D}} = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta & \sin\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\alpha\cos\beta & \cos\alpha\sin\beta & \sin\alpha \end{bmatrix}, \tag{8}$$

where α and β are angles equivalents to latitude and longitude, respectively (Figure 1b). Applying equation (8) to equation (3) (see Appendix B: for details), equation (3) becomes

$$k\begin{bmatrix} -{c_\alpha}^2{c_\beta}^2 & -{c_\alpha}^2 s_\beta c_\beta & -s_\alpha c_\alpha c_\beta & {c_\alpha}^2{c_\beta}^2 & {c_\alpha}^2 s_\beta c_\beta & s_\alpha c_\alpha c_\beta \\ -{c_\alpha}^2 s_\beta c_\beta & -{c_\alpha}^2{s_\beta}^2 & -s_\alpha c_\alpha s_\beta & {c_\alpha}^2 s_\beta c_\beta & {c_\alpha}^2{s_\beta}^2 & s_\alpha c_\alpha s_\beta \\ -s_\alpha c_\alpha c_\beta & -s_\alpha c_\alpha s_\beta & -{s_\alpha}^2 & s_\alpha c_\alpha c_\beta & s_\alpha c_\alpha s_\beta & {s_\alpha}^2 \\ {c_\alpha}^2{c_\beta}^2 & {c_\alpha}^2 s_\beta c_\beta & s_\alpha c_\alpha c_\beta & -{c_\alpha}^2{c_\beta}^2 & -{c_\alpha}^2 s_\beta c_\beta & -s_\alpha c_\alpha c_\beta \\ {c_\alpha}^2 s_\beta c_\beta & {c_\alpha}^2{s_\beta}^2 & s_\alpha c_\alpha s_\beta & -{c_\alpha}^2 s_\beta c_\beta & -{c_\alpha}^2{s_\beta}^2 & -s_\alpha c_\alpha s_\beta \\ s_\alpha c_\alpha c_\beta & s_\alpha c_\alpha s_\beta & {s_\alpha}^2 & -s_\alpha c_\alpha c_\beta & -s_\alpha c_\alpha s_\beta & -{s_\alpha}^2 \end{bmatrix}\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} f_{1,x} \\ f_{1,y} \\ f_{1,z} \\ f_{2,x} \\ f_{2,y} \\ f_{2,z} \end{pmatrix} + kl_0 \begin{pmatrix} c_\alpha c_\beta \\ c_\alpha s_\beta \\ s_\alpha \\ -c_\alpha c_\beta \\ -c_\alpha s_\beta \\ -s_\alpha \end{pmatrix}, \tag{9}$$

where $s_\alpha \equiv \sin\alpha$, $c_\alpha \equiv \cos\alpha$, $s_\beta \equiv \sin\beta$ and $c_\beta \equiv \cos\beta$. The system of equations is solved as described above (see equation (7)).

2.2 Boundary Conditions

Boundary conditions are necessary to constrain the mesh to the desired domain boundaries. In the simple case of a rectangular mesh, a boundary node is free to slide along a domain edges parallel to the X- or Y-axis. This is done by fixing one of its y_i or x_i values and letting the other value vary so that the node is free to move along the boundary. In the general case of a boundary oblique to the X- or Y-axes, this requires a transformation to a new local coordinate system in which the constraint direction is parallel to a local coordinate axis. This is sketched in Figure 2 where node 2 is free to slide along the tilted segment (yellow dashed line in Figure 2) since $y_2{}' = 0$ defines the boundary constraint. The boundary condition is imposed by the rotation of coordinate system for node 2 given by the transformation matrix $\boldsymbol{T}$ relating $\boldsymbol{x}$ to $\boldsymbol{x}'$ by

$$\underbrace{\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{pmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cos\alpha_2 & -\sin\alpha_2 & & \\ & & \sin\alpha_2 & \cos\alpha_2 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}}_{\boldsymbol{T}} \underbrace{\begin{pmatrix} x_1 \\ y_1 \\ x_2{}' \\ 0 \\ x_3 \\ y_3 \end{pmatrix}}_{\boldsymbol{x}'}. \tag{10}$$

Applying $\boldsymbol{T}$ to the stiffness matrix and force vector

$$\boldsymbol{K}' = \boldsymbol{T}^\mathsf{T}\boldsymbol{K}\boldsymbol{T}, \tag{11}$$

$$\boldsymbol{f_{l_0}}' = \boldsymbol{T}^\mathsf{T}\boldsymbol{f_{l_0}}, \tag{12}$$

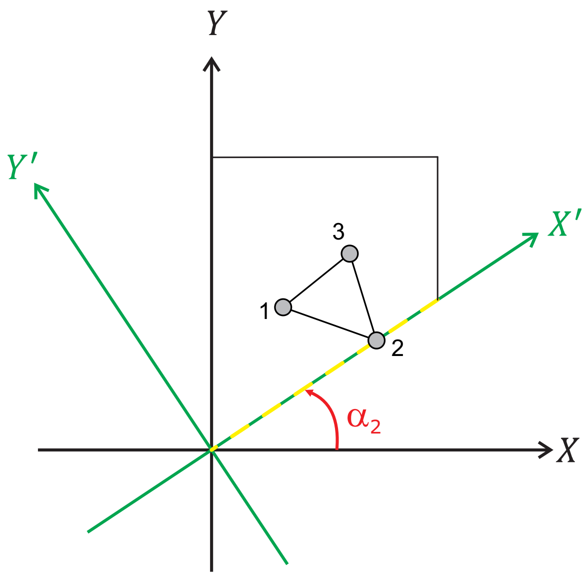

Figure 2: Implementation of boundary conditions along a tilted boundary segment (yellow dashed line), showing one triangle of the mesh. A rotation is needed for the node 2 in order to pass from the global reference system (X, Y) to the local reference system (X', Y') where $y_2' = 0$ is the constrained boundary condition.

with the system of equations transformed into

$$\boldsymbol{K}'\boldsymbol{x}' = \boldsymbol{f}_{\boldsymbol{l}_0}', \tag{13}$$

which is solved for $\boldsymbol{x}'$. Original global coordinates are recovered through the back-transformation

$$\boldsymbol{x} = \boldsymbol{T}\boldsymbol{x}'. \tag{14}$$

Boundary conditions for a cylindrical annulus mesh further generalize the treatment for straight-sided boundary line-segments. This boundary condition prescribes boundary nodes that freely move along a local tangent to the boundary. Nodal motion involves two independent steps (Figure 3a): 1) The node moves along the tangent line to the circle at its current location, and 2) the new node location is projected back to the circle in the radial direction. This approximation assumes that the radial distance needed to put the node back onto the circle is small compared to the distance moved along the tangent line. The mathematical implementation is sketched in Figure 3b. The boundary condition for node 2 is that it can move along its tangent line (dashed line in Figure 3b) since $y_2' = |r|$, where r is the radial distance from the centre of the cylindrical annulus mesh to the boundary. This boundary condition is imposed by the coordinate rotation given by the transformation

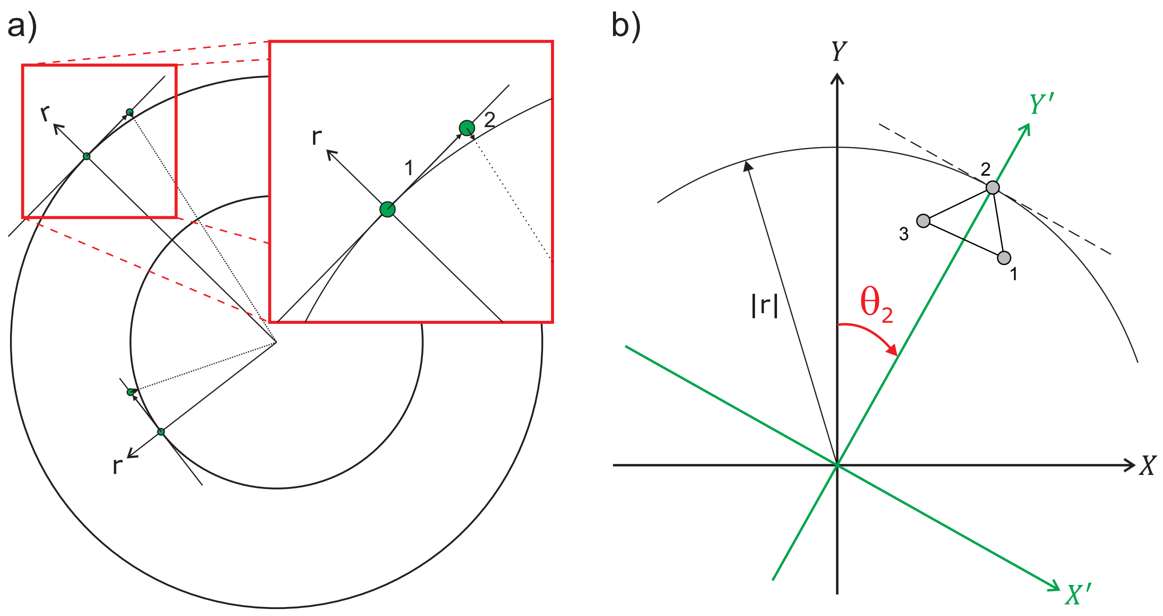

Figure 3: (a) Conceptual diagram for circular boundary conditions. The motion of boundary nodes is first restricted to be along a tangent line to the circle. Nodes are then 'pulled back' to the circle by projecting in the radial direction. (b) Implementation of circular boundary conditions for one triangle. A rotation is needed for the node 2 in order to pass from the global reference system (X, Y) to the local surface-parallel reference system (X', Y') where $y_2{}' = |r|$ is the constrained boundary condition.

matrix $\boldsymbol{T}$ relating $\boldsymbol{x}$ to $\boldsymbol{x}'$ (local surface-parallel reference system (X', Y') in green in Figure 3b) as

$$\underbrace{\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{pmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cos\theta_2 & \sin\theta_2 & & \\ & & -\sin\theta_2 & \cos\theta_2 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}}_{\boldsymbol{T}} \underbrace{\begin{pmatrix} x_1 \\ y_1 \\ x_2{}' \\ |r| \\ x_3 \\ y_3 \end{pmatrix}}_{\boldsymbol{x}'}, \tag{15}$$

where θ_2 is the angle of the node 2 measured from the Y axis in the clockwise direction.

For 3-D applications, we currently focus on developing unstructured spherical meshes. In this case, a useful boundary condition consists in prescribing boundary nodes free to move along a local tangent plane to the spherical surface. Nodal sliding again involves two independent steps (Figure 4a): 1) The node is allowed to move along the local tangent plane to the sphere, and 2) the node is returned to the sphere's surface by projecting in the radial direction. This is sketched in Figure 4b. Node 2 is free to slide along the tangent plane

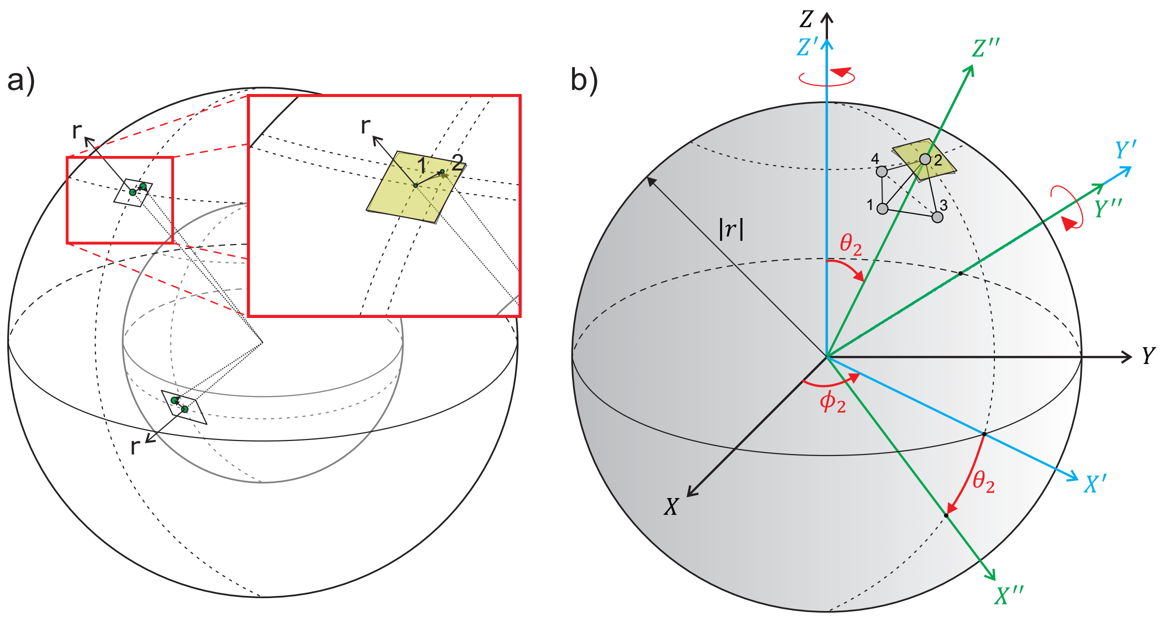

Figure 4: (a) Conceptual diagram for spherical boundary conditions. The motion of boundary nodes is first restricted to be along the tangent plane to the sphere. Then, they are 'pulled back' to the sphere's surface by projecting in the radial direction. (b) Implementation of spherical boundary conditions for one tetrahedron. Two rotations are needed for node 2 to pass from the global reference system (X, Y, Z) to the local reference system (X'', Y'', Z''), where $z_2'' = |r|$ is the boundary condition.

since the boundary condition is $z_2'' = |r|$, where r is sphere radius. The boundary condition is imposed by two rotations of the coordinate system for node 2. The first rotation is around the Z axis by an angle ϕ_2, which is the longitude of node 2 (local reference system (X', Y', Z') in blue in Figure 4b). The second rotation is around the Y' axis by an angle θ_2, which is the colatitude for node 2 (local reference system (X'', Y'', Z'') in green in Figure 4b). The complete rotation is given by the transformation matrix $\boldsymbol{T}$ relating global coordinates $\boldsymbol{x}$ to local coordinates $\boldsymbol{x}''$

$$\underbrace{\begin{pmatrix} x_1\\ y_1\\ z_1\\ x_2\\ y_2\\ z_2\\ x_3\\ y_3\\ z_3\\ x_4\\ y_4\\ z_4 \end{pmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} 1 &&&&&&&&&&& \\ & 1 &&&&&&&&&& \\ && 1 &&&&&&&&& \\ &&& \cos\phi_2\cos\theta_2 & -\sin\phi_2 & \cos\phi_2\sin\theta_2 &&&&&& \\ &&& \sin\phi_2\cos\theta_2 & \cos\theta_2 & \sin\phi_2\sin\theta_2 &&&&&& \\ &&& -\sin\theta_2 & 0 & \cos\theta_2 &&&&&& \\ &&&&&& 1 &&&&& \\ &&&&&&& 1 &&&& \\ &&&&&&&& 1 &&& \\ &&&&&&&&& 1 && \\ &&&&&&&&&& 1 & \\ &&&&&&&&&&& 1 \end{bmatrix}}_{\boldsymbol{T}} \underbrace{\begin{pmatrix} x_1\\ y_1\\ z_1\\ x_2''\\ y_2''\\ |r|\\ x_3\\ y_3\\ z_3\\ x_4\\ y_4\\ z_4 \end{pmatrix}}_{\boldsymbol{x}''} . \tag{16}$$

This transformation matrix contains a θ and ϕ angle for each node on the spherical boundary.

2.3 Mesh refinement

In this algorithm we refine a mesh by decreasing element sizes in the region of interest. One common issue in the refinement process arises when the size contrast between large and small elements occurs over a short spatial interval so that poorly-shaped elements may form. To mitigate this issue a transition region is defined to surround the refined region by use of a guide-mesh approach.

2.3.1 Definition of preferred nodal distances (background guide-mesh)

The first task is to define the preferred nodal distances within the refined (l_{0r}) and coarse (l_{0c}) regions and their dimensions. To avoid poor quality elements, an appropriately smooth transition for mesh refinement should be specified. We choose a preferred spring-length function that is defined on a background 'guide-mesh'. This approach is very similar to the background grid approach described by *Löhner and Parikh* [1988].

The generation of a refined rectangular mesh using the guide-mesh approach involves the following steps. First, create a (coarse) mesh to serve as a guide-mesh, typically using a small number of nodes to define the boundaries of the domain and the internal boundaries of the embedded high-resolution and transition sub-regions. Second, create the design function $l_0(x, y)$ for each node of the guide-mesh. This function defines the desired length for the springs around those points. Third, the function $l_0(x, y)$ is evaluated at the midpoint of all springs by interpolation with linear Finite Element shape functions. We find that this coarse guide-mesh is a simple and flexible way to impose smoothly varying nodal spacing during mesh generation. Figure 5a shows the guide-mesh for the rectangular mesh example whose parameters are listed in Table 1.

The generation of a refined cylindrical annulus mesh using the guide-mesh is similar except that the function $l_0(x, y)$ becomes $l_0(\theta, r)$. In this case the guide-mesh is a coarse cylindrical annulus mesh defined in polar coordinates. Figure 6a shows the guide-mesh (white dashed lines) defining the refined (red), transition (green) and coarse (blue) regions and the parameters are listed in Table 1. Figure 6c shows a zoom of the guide-mesh defined in polar coordinates. The use of a guide-mesh defined in polar coordinates (white dashed lines in Figure 6a and Figure 6c) instead of Cartesian coordinates (white dashed lines in Figure 6b and Figure 6d) takes advantage of higher precision when l_0 values are interpolated in points both close and on the boundaries (green dots in Figure 6c).

The generation of a refined spherical shell mesh using the guide-mesh involves similar steps except that the preferred length function $l_0(\theta, r)$ is now $l_0(\theta, \phi, r)$. In this case the guide-mesh is a coarse spherical shell mesh defined in spherical coordinates (Figure 7a).

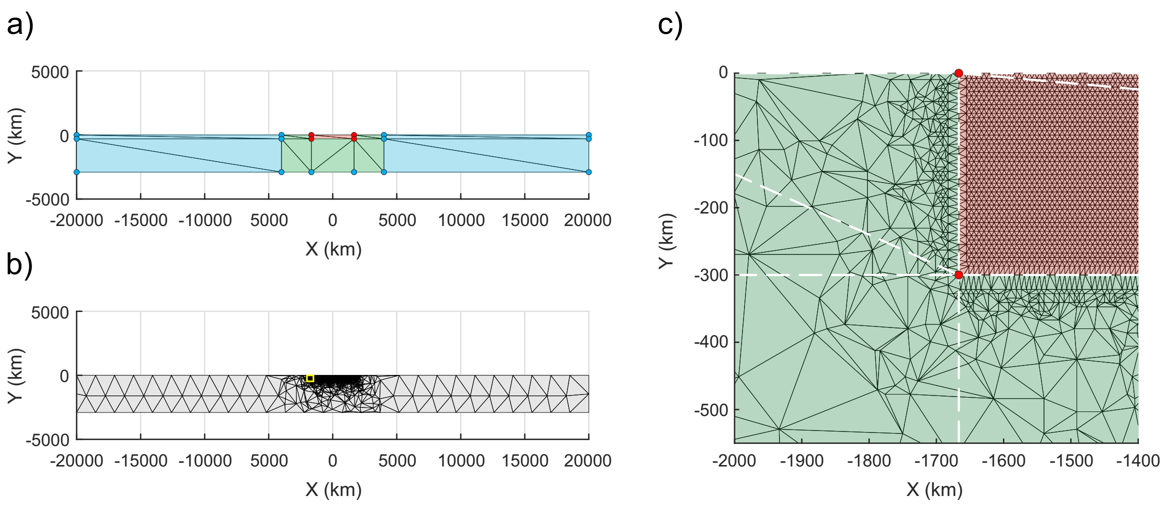

Figure 5: (a) Guide-mesh for a rectangular mesh defined by a few nodes in Cartesian coordinates. The parameters for this mesh are listed in Table 1. Each node is assigned a value for the desired spring length, being l_{0r} for red dots and l_{0c} for blue dots. The length of the springs within the refined region (in red) is approximately equal to l_{0r}. The length of the springs within the transition region (in green) varies smoothly from l_{0r} to l_{0c}. The length of springs within the coarse region (in blue) is approximately equal to l_{0c}. (b) Initial guess for the rectangular mesh. (c) Zoom around the left boundary of the refined region for the initial guess (yellow line in (b)). Corresponding pictures of the final mesh are shown in Figure 11a. The guide-mesh defining refined (red) and transition (green) regions is shown with white dashed lines.

2.3.2 Initial placement of the nodes

The next step is to create a starting guess for node locations (computational work is significantly reduced with a good initial guess for nodal positions). Nodes on the boundary and within the domain are created considering both the location of the refined region and the desired spring lengths for elements inside the refined and coarse regions. In 2-D, the boundary nodes in the refined and coarse regions are created using l_{0r} and l_{0c} respectively for the spacing between the nodes. Interior nodes within the refined and coarse regions are created using a circle packing lattice with radii equal to $l_{0r}/2$ and $l_{0c}/2$ respectively. This fills each region with an equilateral triangular tiling. In the transition region the size of the elements is expected to change smoothly between l_{0r} and l_{0c}. The initial placement for boundary and interior nodes in the transition region is created using l_{0r} as explained above. After this step, the rejection method described in *Persson and Strang* [2004] is used to discard points and create a 'balanced' initial distribution of nodes. After performing a Delaunay triangulation, a quasi-regular mesh of triangles is created within the refined and coarse regions, with a poorly structured transition region between them (see Figure 5b for an example of a rectangular mesh). Figure 5c

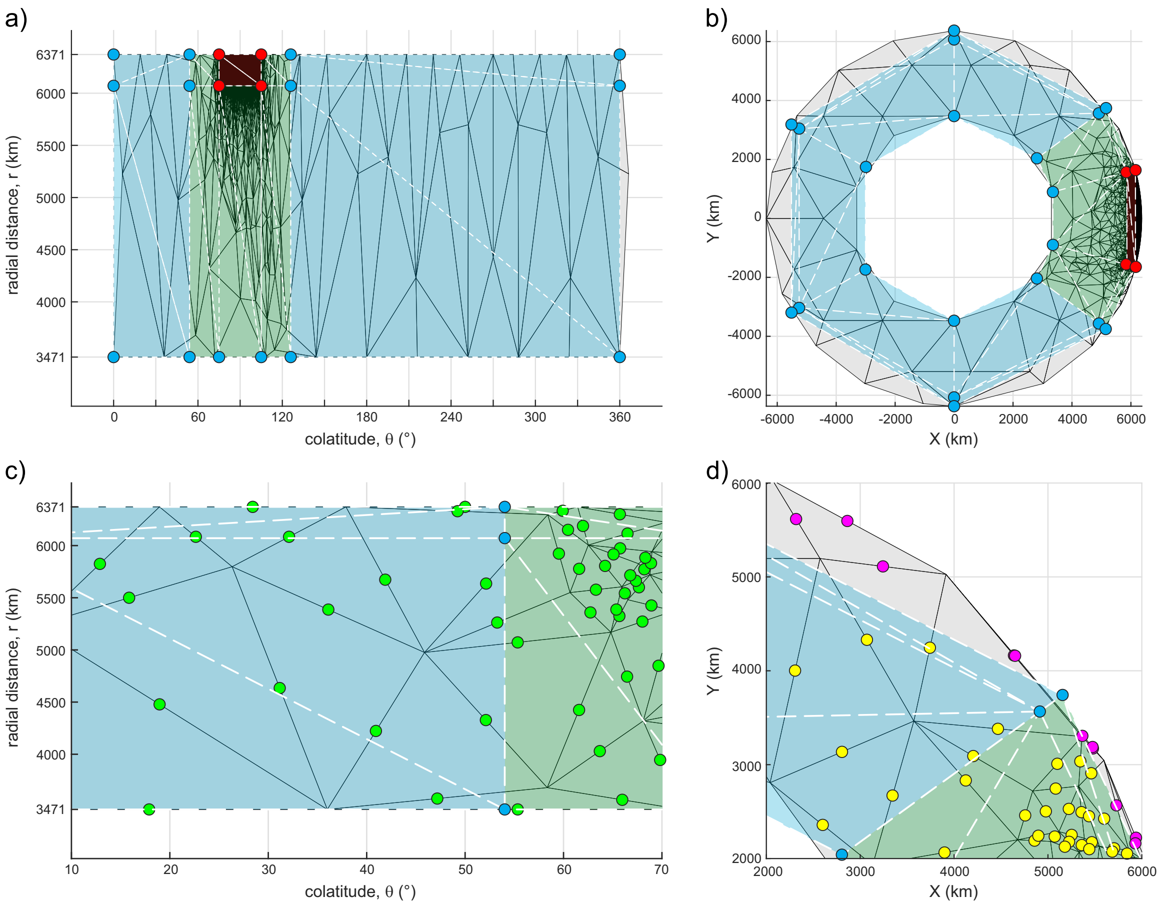

Figure 6: (a) Guide-mesh (white dashed lines) defined by a few nodes (red and blue dots represent l_{0r} and l_{0c} respectively) in polar coordinates for a cylindrical annulus mesh (initial guess is shown in black solid lines). Red, green and blue colours represent the refined, transition and coarse regions respectively. (b) Guide-mesh defined in Cartesian coordinates. Same colours as in (a). (c) Zoom around an edge of the transition region in polar coordinates. The function $l_0(\theta, r)$ can be interpolated at green dots with maximum precision since both boundaries – the cylindrical annulus mesh and its guide-mesh – overlap. (d) Zoom around an edge of the transition region in Cartesian coordinates. The function $l_0(x, y)$ cannot be interpolated at magenta dots since they lay outside of the outer boundary of a Cartesian guide-mesh. The precision of the interpolated l_0 values at yellow dots is reduced since both boundaries – the cylindrical annulus mesh and its guide-mesh – do not overlap.

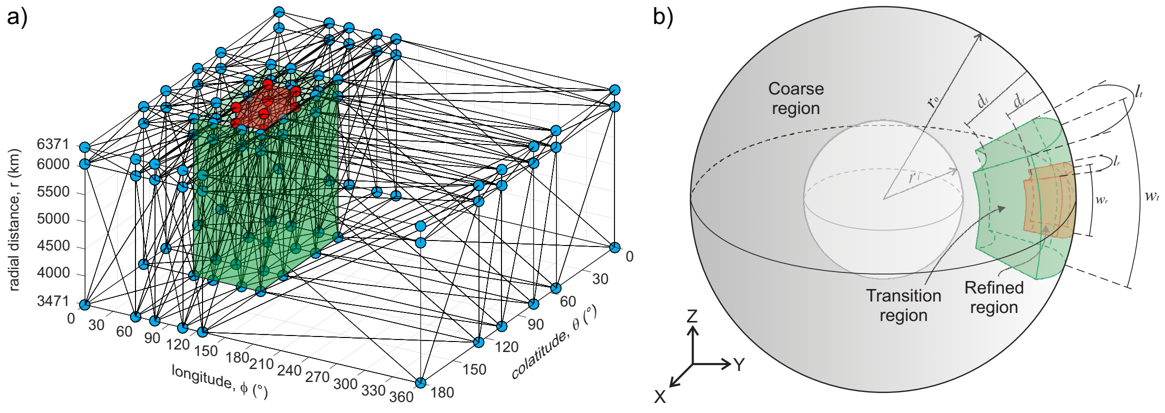

Figure 7: (a) Guide-mesh defined by a few nodes (red and blue dots represent l_{0r} and l_{0c} respectively) in spherical coordinates for a spherical shell. The length of the springs within the refined region (red) is approximately equal to l_{0r}. The length of the springs within the transition region (green) smoothly varies from l_{0r} to l_{0c}. Outside the transition region the length of the springs is approximately equal to l_{0c}. (b) Model domain representing a 3-D spherical shell with an embedded high-resolution sub-region.

shows a zoom of the initial rectangular mesh. In theory, Delaunay triangulations could potentially have the issue of not respecting complex boundaries. However, we have yet to see this issue in a mesh generated with our algorithm, perhaps the main application are global-scale models with relatively simple and smooth external boundaries.

In 3-D, the boundary nodes in the refined and coarse regions are created by recursively splitting an initial dodecahedron according to l_{0r} and l_{0c} respectively. This gives a uniform distribution of equilateral triangles on the spherical surface. In contrast to equilateral triangles in 2-D, which are able to fill up the plane, regular tetrahedra do not fill up the entire space. However, there do exist some compact lattices, e.g. the hexagonal close packing (hcp) lattice, that create a nodal distribution that leads to well-shaped tetrahedra. The interior nodes within the refined and coarse regions are created by a close-packing of equal spheres with radii equal to $l_{0r}/2$ and $l_{0c}/2$ respectively. Initial placement for boundary and interior nodes in the transition region is created using l_{0r} as explained above, with the rejection method of *Persson and Strang* [2004] used to discard points and create the initial nodal guess.

2.3.3 Quality factor

The ’quality’ of a mesh is determined by assessing the quality of its individual elements. This usually involves measures of angles, edge lengths, areas (in 2-D), volumes (in 3-D), or the radius of its inscribed and circumscribed circles/spheres [e.g. *Dompierre et al.*, 1998; *Shewchuk*, 2002]. Here we use a normalized quality factor, which in 2-D is given by

$$q_{2D} = \frac{2r_c}{R_c}, \tag{17}$$

where r_c is the radius of the element's inscribed circle and R_c is the radius of its circumscribed circle. R_c and r_c can be expressed as

$$r_c = \frac{1}{2}\sqrt{\frac{(b+c-a)(c+a-b)(a+b-c)}{a+b+c}}, \tag{18}$$

$$R_c = \frac{abc}{\sqrt{(a+b+c)(b+c-a)(c+a-b)(a+b-c)}}, \tag{19}$$

where a, b and c are the side lengths of the triangle. A fair criterion to evaluate the quality of a mesh is to provide the minimum and mean values of the quality factor [cf. *Alliez et al.*, 2005]. Here both are used as control parameters to determine when the iterative algorithm has reached the desired mesh quality tolerances (Figure 9).

The corresponding 3-D quality factor for a tetrahedron is defined by

$$q_{3D} = \frac{3r_s}{R_s}, \tag{20}$$

where r_s is the radius of the tetrahedron's inscribed sphere and R_s is the radius of its circumscribed sphere. R_s and r_s are given by

$$r_s = \frac{|\boldsymbol{a}\cdot(\boldsymbol{b}\times\boldsymbol{c})|}{(|\boldsymbol{a}\times\boldsymbol{b}| + |\boldsymbol{b}\times\boldsymbol{c}| + |\boldsymbol{c}\times\boldsymbol{a}| + |(\boldsymbol{a}\times\boldsymbol{b}) + (\boldsymbol{b}\times\boldsymbol{c}) + (\boldsymbol{c}\times\boldsymbol{a})|)}, \tag{21}$$

$$R_s = \frac{|\boldsymbol{a}^2\cdot(\boldsymbol{b}\times\boldsymbol{c}) + \boldsymbol{b}^2\cdot(\boldsymbol{c}\times\boldsymbol{a}) + \boldsymbol{c}^2\cdot(\boldsymbol{a}\times\boldsymbol{b})|}{2|\boldsymbol{a}\cdot(\boldsymbol{b}\times\boldsymbol{c})|}, \tag{22}$$

where $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are vectors pointing from one node, O, to the three other nodes of the tetrahedron A, B and C respectively (Figure 8a). This quality factor is normalized to be 0 for degenerate tetrahedra and 1 for regular tetrahedra. Different definitions for normalized aspect ratios lead to different estimators for the global quality of a mesh. For example, *Anderson et al.* [2005] define a shape measure s that depends on tetrahedral volume and the lengths of its edges. Computing q_{3D} and s for the same mesh gives differences of up to 0.1 for the worst element (Figure 8b). The quality factor q_{3D} that we choose to use is a more restrictive aspect ratio than the measure s.

2.4 Local mesh improvements

So far, the above algorithm only moves nodes within the domain to meet desired spring lengths/internodal distances. However, in general we do not know a priori how many nodes will be needed for a mesh. Therefore,

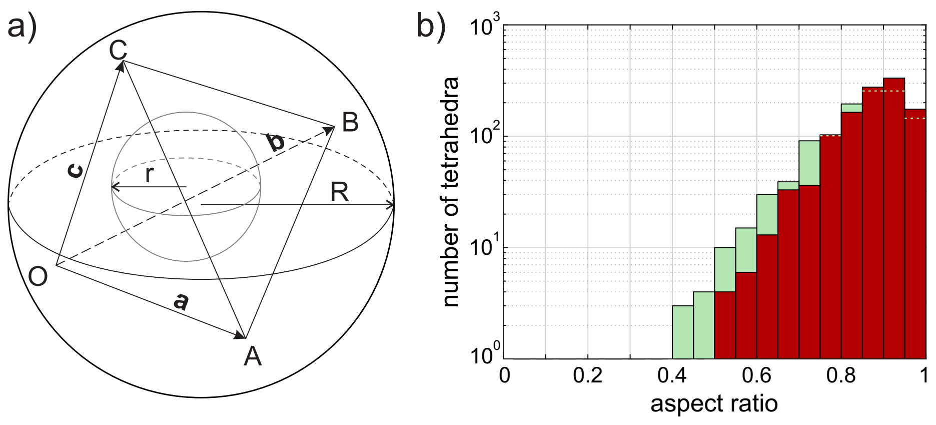

Figure 8: (a) Tetrahedron with vertices OABC. R and r are the radius of the circumscribed and inscribed spheres respectively. (b) Number of tetrahedra as a function of the quality factor q_{3D} (green) and the shape measure s (red) for the same mesh.

we use algorithms to locally add and remove nodes when the equilibrium spacing is too loose or tight. After solving for nodal positions, we check if the mesh has reached the expected nodal density by determining the mean of the misfit in spring lengths (Figure 9), given by

$$\mu = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{l_i - l_{0i}}{l_{0i}}\right|, \tag{23}$$

where l is the actual spring length, l_0 is the desired spring length and N is the total number of springs in the mesh. Nodes are added or rejected (Section 2.4.1) if $\mu \geq \mu_t$. When $\mu < \mu_t$ the expected nodal density is achieved and element shape improvements (Section 2.4.2) are applied to obtain higher quality elements. After some experimentation we found it appropriate to use $0.02 < \mu_t < 0.05$ for 2-D meshes. In 3-D we use $\mu_t = 0.14$, although this can vary from 0.1 to 0.2 depending on the degree of mesh refinement. Note attainable values of μ_t for 2-D meshes are much smaller than for 3-D meshes.

2.4.1 Add/reject nodes

The possibility to either add or reject nodes plays an important local role. This feature is especially relevant when the goal is to create a global coarse mesh with an embedded high-resolution sub-region. The logic for adding or rejecting nodes is based on the relative length change of the virtual springs that connect nodes

$$\epsilon = \frac{l - l_0}{l_0}, \tag{24}$$

indicating whether springs are stretched ($\epsilon > 0$) or compressed ($\epsilon < 0$) with respect to their desired lengths. A new node is created at the midpoint of those springs with $\epsilon > 0.5$, i.e. springs stretched more than 50%

greater than their desired length. One node at the end of a spring is rejected when $\epsilon < -0.5$, i.e. springs compressed more than 50% below their desired length. In order to save computational time, the add/reject nodes routine is called as a sub-iteration within the main iteration in which nodal positions are found. Sub-iterations are performed until the percentage of springs with $|\epsilon| > 0.5$ in the sub-iteration $j + 1$ is higher than in the sub-iteration j. This implementation is especially useful when a large fraction of nodes needs to be added or rejected within a particular region of the mesh, e.g. when a relatively poor initial guess is used. For a cylindrical annulus mesh and a spherical shell mesh the only difference appears when a new node is added on a boundary spring. In this case, the new boundary node needs to be projected onto the surface along the radial direction.

2.4.2 Element shape improvement

2.4.2.1 ***Smooth positions of the interior nodes*** Good quality meshes are directly related to the generation of isotropic elements [*Alliez et al.*, 2005]. The 'Laplacian smoothing' criterion [*Choi et al.*, 2003] is used to improve the shape of poorly shaped elements, i.e. to make elements as close to equilateral triangles or regular tetrahedra as possible. This fast matrix-free algebraic operation is only applied to interior nodes. We find that this additional algebraic smoothing operation leads to a significant mesh improvement at a computational cost much lower than that needed to form and solve the matrix equation for internal spring equilibrium. The interior nodes are repositioned to the mean of the barycentres of their surrounding elements, i.e.

$$\boldsymbol{x_s} = \frac{\sum_{i=1}^{N} \boldsymbol{x_b}_i}{N}, \tag{25}$$

where $\boldsymbol{x_s}$ are the new coordinates of the interior node, N is the number of elements surrounding the interior node and $\boldsymbol{x_b}_i$ are the barycentre coordinates of the i-th surrounding element. Figure S1 shows an example of Laplacian smoothing of interior nodes for a 2-D mesh.

In 3-D, even when the expected nodal density is achieved ($\mu < \mu_t$) by adding or rejecting nodes, a considerable number of poorly shaped tetrahedra can still persist. Methods based on swapping edges or faces to improve element quality can possibly generate non-Delaunay triangulations, which will cause problems in algorithms that rely on a mesh created by a Delaunay triangulation (e.g. point search algorithms). Hence, in addition to smoothing the position of interior nodes, we recommend two additional operations to further improve the quality of tetrahedral elements.

2.4.2.2 ***Improvement of badly shaped tetrahedra*** Unstructured 3-D meshes from Delaunay triangulations typically have a few elements with poor quality factors (see *Cheng et al.* [2000] for a complete categorization of badly shaped tetrahedra). We improve these by modifying one node of each badly shaped tetrahedron. For each badly shaped tetrahedron, identified by $q_{3D} < q_{bad}$, where $0.2 \leq q_{bad} \leq 0.3$, we select the spring with the maximum distortion, i.e. $max(|\epsilon|)$. If $\epsilon > 0$, a new node is created in the midpoint of the selected spring, while a node at one end of the selected spring is removed if $\epsilon < 0$. A new connectivity is then created by another Delaunay triangulation. The new connectivity is only modified in the surroundings of nodes that have been added or removed, keeping the rest of the connectivity to be the same as the old triangulation. Figure S2 illustrates a simple example that improves the badly shaped tetrahedra which form when meshing the unit cube.

2.4.2.3 ***Removing slivers*** Slivers are degenerate tetrahedra whose vertices are well-spaced and near the equator of their circumsphere, hence their quality factor and enclosed volume are close to zero. We define a sliver to be a tetrahedron with $q_{3D} < 0.1$. Our routine for removing slivers is purely geometrical, i.e. it does not consider the actual or desired length of the springs. The four vertices of each sliver are replaced by the three mesh points of the best potential triangle that can be generated from all permutations of its vertices and potential new nodes created at the midpoints of its springs (Figure S3). Delaunay triangulation is then called to create the connectivity matrix around the changed nodes. In all our tests, this simple algorithm removed all slivers within 2-3 iterations.

2.5 Flow charts for iterative mesh generation

This mesh-generation algorithm has its simplest form when creating a 2-D rectangular mesh with an embedded high-resolution sub-region (white and yellow boxes in Figure 9). The algorithm can also generate a 2-D cylindrical annulus mesh with an embedded high-resolution sub-region (white and orange boxes in Figure 9). The white and green boxes in Figure 9 show the flowchart that describes the algorithm for the generation of 3-D spherical shell meshes that include an embedded high-resolution sub-region.

3 Results

Several tests were performed with the above implementations to assess the robustness and performance of this mesh-generation algorithm (Figure 10). The input parameters that controlled the algorithm are listed in Table 1. All tests in this study were performed using MATLAB R2015a (8.5.0.197613) on a 3.2 GHz Intel Core i5 (MacOSX 10.12.5) with 24 GB of 1600 MHz memory. The code to generate these meshes is available in the Supporting information.

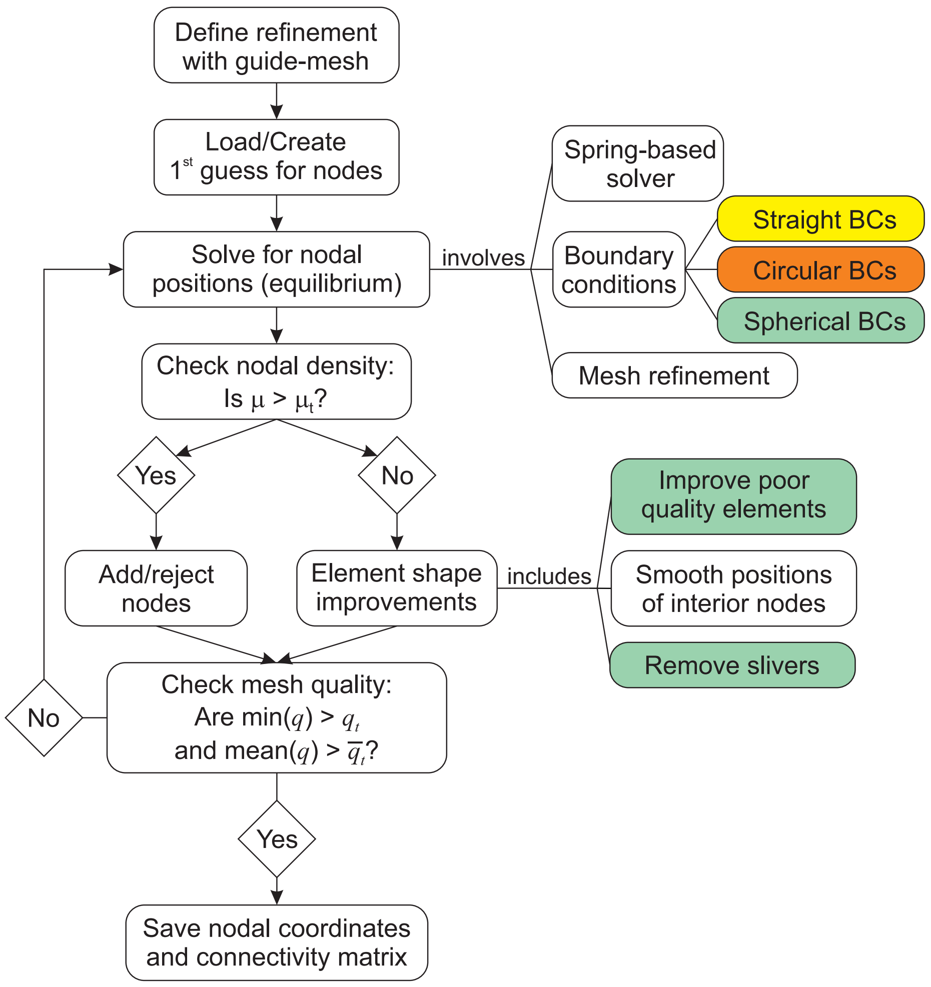
Define refinement with guide-mesh
Load/Create 1st guess for nodes
Solve for nodal positions (equilibrium)
involves
Spring-based solver
Boundary conditions
Straight BCs
Circular BCs
Spherical BCs
Mesh refinement
Check nodal density: Is μ > μt?
Yes
No
Add/reject nodes
Element shape improvements
includes
Improve poor quality elements
Smooth positions of interior nodes
Remove slivers
Check mesh quality: Are min(q) > qt and mean(q) > q̄t?
No
Yes
Save nodal coordinates and connectivity matrix

Figure 9: Flow chart of the mesh-generation algorithm. Yellow, orange and green boxes represent routines exclusively used for creating 2-D rectangular meshes, 2-D cylindrical annulus meshes and 3-D spherical shell meshes, respectively. White boxes represent shared routines. μ is the mean misfit spring length (equation (23)) and q is the element quality factor (equations (17) and (20) for triangular and tetrahedral elements respectively). Tolerance parameters μ_t, q_t and $\bar{q}_t$ are listed in Table 1.

Table 1: Mesh Parameters.

Symbol	**Meaning**	**Rectangular box**	**Cylindrical annulus**	**Spherical shell**
d	Depth	2900 km	-	-
l	Length	40000 km	-	-
r_i	Inner radius	-	3471 km	3471 km
r_o	Outer radius	-	6371 km	6371 km
x_0	x-coordinate centre of refined region	0 km	-	-
z_0	z-coordinate centre of refined region	0 km	-	-
θ_0	Colatitude centre of refined region	-	90°	90°
ϕ_0	Longitude centre of refined region	-	-	90°
r_0	Radial distance centre of refined region	-	6371 km	6371 km
l_{0c}	Desired spring length for elements inside the coarse region	1500 km	2000 km	2000 km
l_{0r}	Desired spring length for elements inside the refined region	7.5 km	10 km	60 km
d_t	Transition region depth	2900 km	2900 km	2900 km
l_t	Transition region length	8000 km	8000 km	6800 km
w_t	Transition region width	-	-	9600 km
d_r	Refined region depth	300 km	300 km	300 km
l_r	Refined region length	3333 km	3333 km	2200 km
w_r	Refined region width	-	-	5000 km
q_t	Tolerance for minimum quality factor	0.45	0.40	0.23
$\bar{q}_t$	Tolerance for mean quality factor	0.89	0.93	0.80
μ_t	Tolerance for mean misfit spring length	0.025	0.04	0.14

3.1 Rectangular mesh with an embedded high-resolution region

The algorithm created this mesh in 9 s (purple dot in Figure 10) after 8 outermost loop iterations (cf. Figure 9). Figure 11a shows the final mesh (top) and a zoom around the left boundary of the refined region (bottom) for the iteration 8 (see Figure S4 for iterations 0 (initial mesh) and 1). The final mesh has 22000 nodes forming 43000 triangles with an edge-length factor $l_{0r}/l_{0c} = 1/200$. The percentage of triangles within the coarse, transition and refined regions is 0.3%, 6.3% and 93.4% respectively.

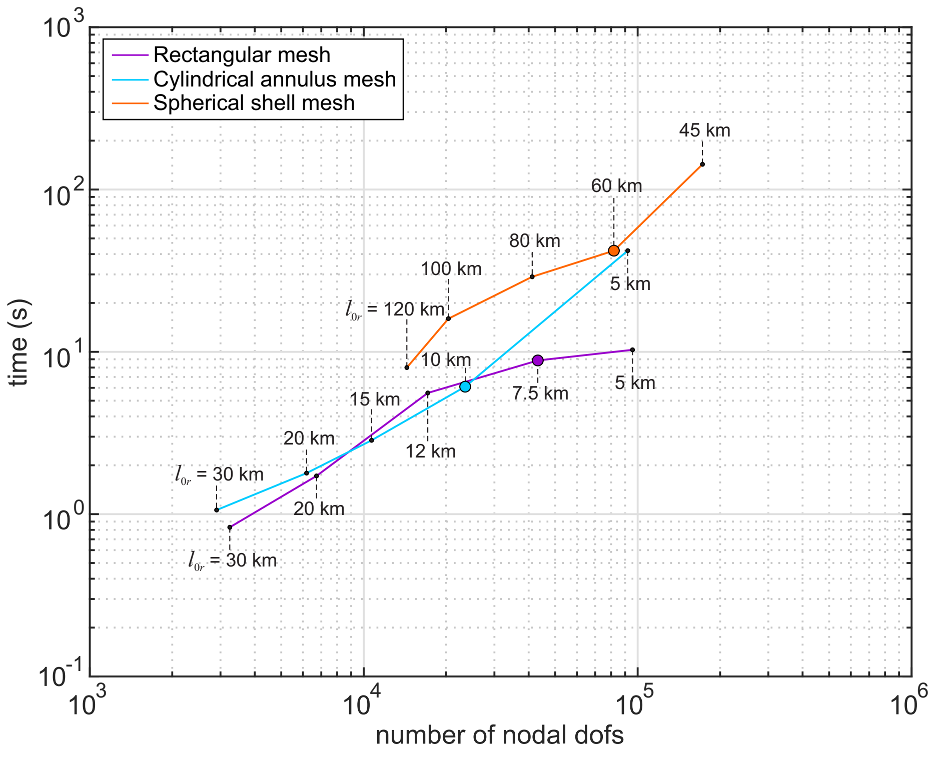

Figure 10: Computational time as a function of the number nodal degrees of freedom (dofs). All meshes were generated using the parameters listed in Table 1; only l_{0r} was varied. The purple dot (7.5 km-rect. mesh) is for the example shown in Figure 11, the blue dot (10 km-cyl. mesh) for Figure 12, and the orange dot (60 km-3D sph. mesh) for Figure 13. For the last data point in blue line (5 km-cyl. mesh), a reduction in q_t from the chosen 0.4 to a less stringent 0.35 would lead to a 2.6 × speed-up. This example highlights the trade-off between compute speed and mesh-quality.

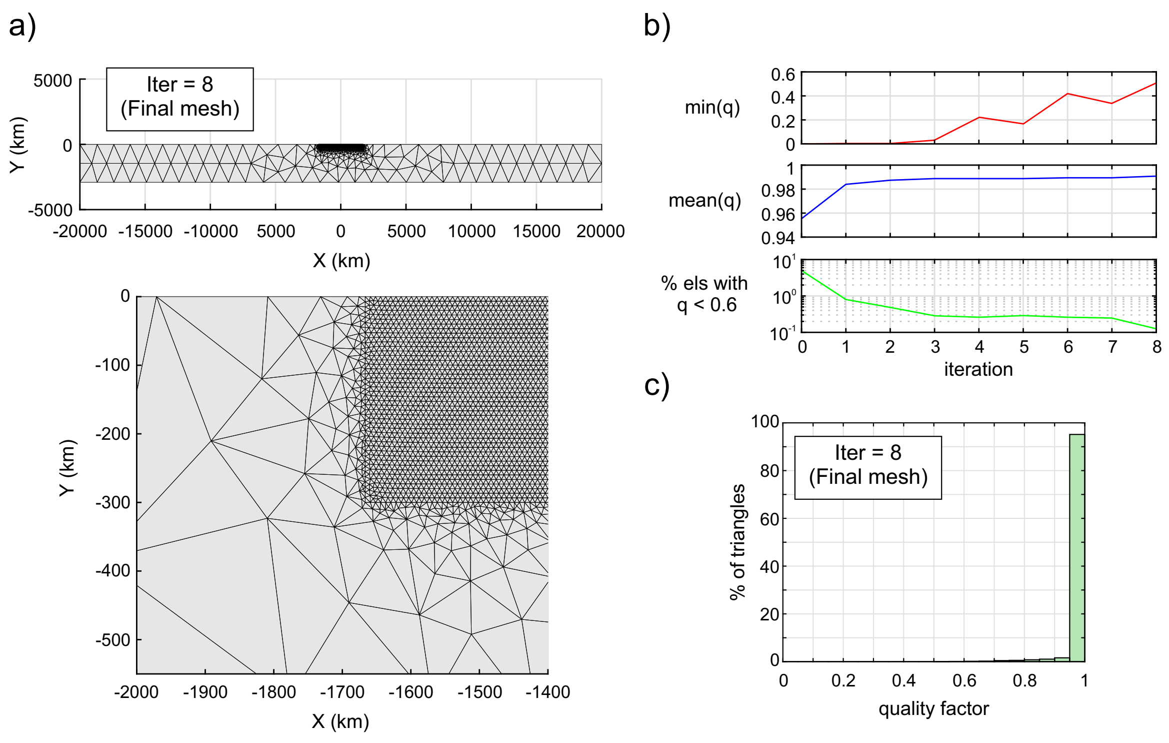

Figure 11: (a) Final mesh (top) for a rectangular box with an embedded high-resolution sub-region and a zoom around the left boundary of the refined region (bottom). (b) Minimum quality factor (red line), mean quality factor for all elements (blue line) and percentage of elements having a quality factor lower than 0.6% (green line) as a function of iteration number. (c) Histogram of the fraction of elements as a function of quality factor for the final mesh. The lowest quality factor for an element is 0.51.

3.2 Cylindrical annulus mesh with an embedded high-resolution region

The algorithm created this mesh in 6 s (blue dot in Figure 10) after 7 iterations. Figure 12a shows the final mesh for iteration 7 (see Figure S5 for iterations 0 (initial mesh) and 1). Figure 12b shows a zoom around an edge of the refined region. The final mesh has 12000 nodes forming 23000 triangular elements with an edge-length factor $l_{0r}/l_{0c} = 1/200$. The percentage of triangles within the coarse, transition and refined regions is 0.2%, 6.1% and 93.7% respectively.

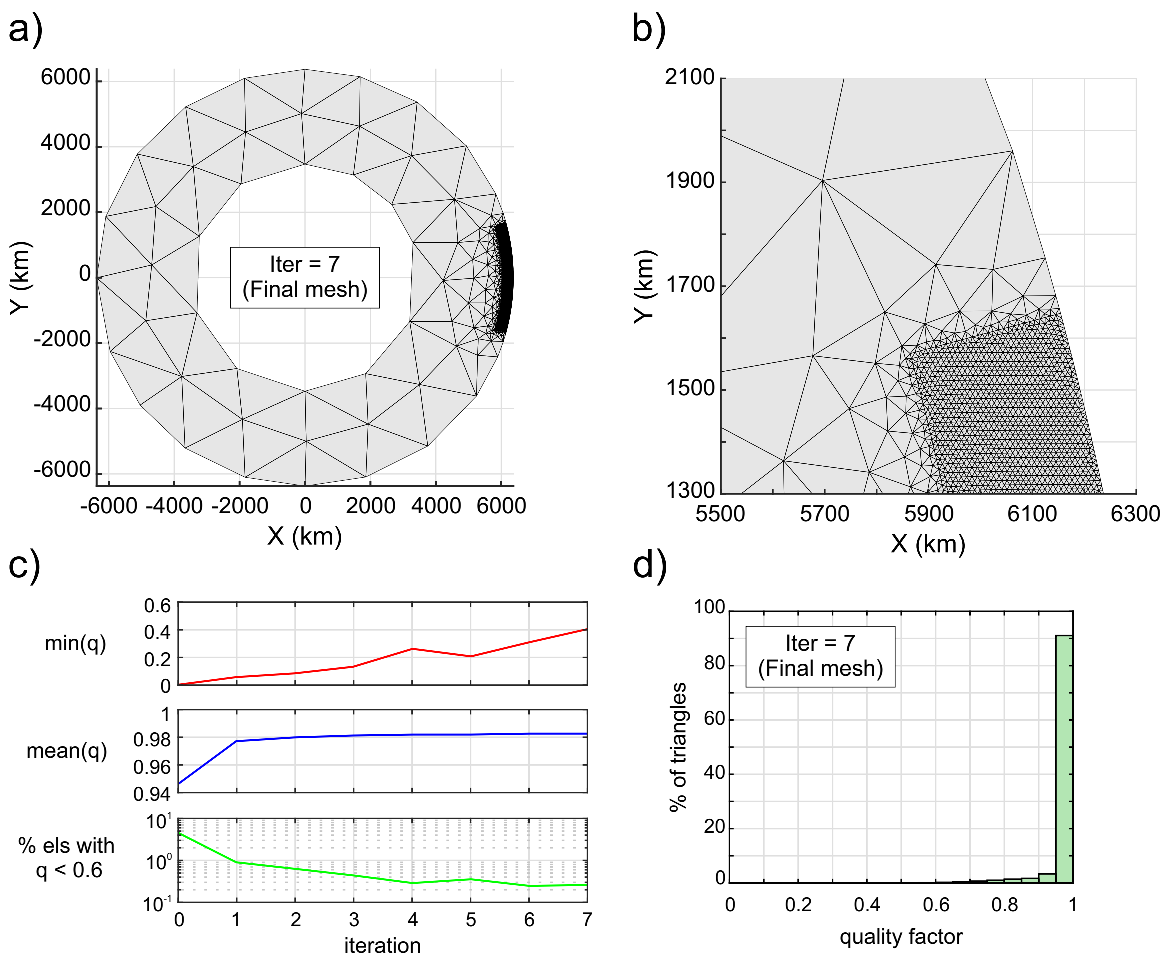

Figure 12: (a) Final mesh for a cylindrical annulus with an embedded high-resolution sub-region. (b) Zoom around an edge of the refined region. (c) Minimum quality factor (red line), mean quality factor for all elements (blue line) and percentage of elements having a quality factor lower than 0.6% (green line) as a function of iteration number. (d) Histogram of the fraction of elements as a function of quality factor for the final mesh. The lowest quality factor for an element is 0.40.

3.3 Spherical shell mesh with an embedded high-resolution region

The domain of this mesh is a spherical shell whose boundaries represent the core-mantle boundary and the Earth's surface (Figure 7b). The smallest tetrahedra with quasi-uniform size lie inside the high-resolution region (red spherical prism in Figure 7b). This region is embedded within a coarser global mesh. A transition region (green spherical prism in Figure 7b) guarantees a gradual change in tetrahedral size from the high-resolution region to the coarse region. We recommend setting the point around which the refined region is created far from the polar axis since the guide-mesh is worse at smoothly interpolating desired spring lengths near the polar axis. The algorithm created the mesh in 42 s (orange dot in Figure 10) after 2 iterations (see Figure 13a for a cross section of the final mesh). Figure 13b shows a detail of the mesh around the northern boundary of the refined region. The mesh has 27000 nodes forming 150000 tetrahedra with an edge-length factor $l_{0r}/l_{0c} = 1/33$. The fraction of tetrahedra within the coarse, transition and refined regions is 0.7%, 21.6% and 77.7% respectively (Figure S6).

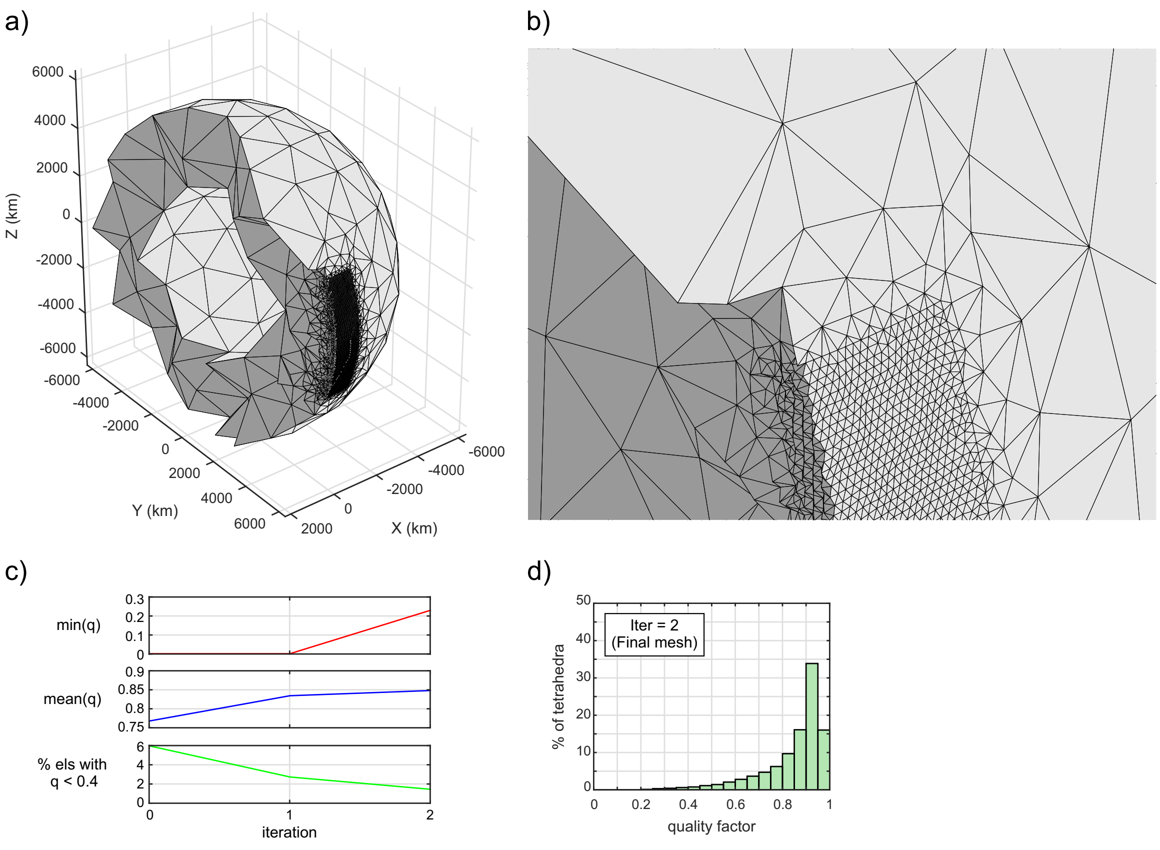

Figure 13: (a) Cross section of the final mesh with an embedded high-resolution sub-region after refinement using the guide-mesh. (b) Zoom around the boundary of the refined region. (c) Minimum quality factor (red line), mean quality factor for all elements (blue line) and fraction of elements having a quality factor lower than 0.4% (green line) as a function of iteration number. (d) Histogram of the fraction of elements as a function of quality factor for the final mesh. The lowest quality factor for an element is 0.23.

4 Geodynamic modelling example

As a practical use-case of the mesh generator, we show its application in recent work. This is a study of the potential mantle flow associated with the first 30 Myr of rifting evolution in the South Atlantic, exploring the influence of Tristan da Cunha plume and initial lithospheric thickness variations. We use M3TET_SPH [*Taramon*, 2018] to solve for the thermo-mechanical viscous flow evolution of the mantle in a whole-mantle spherical shell geometry. The mantle is modelled as a 3-D incompressible fluid that satisfies the Boussinesq approximation.

Figure 14 shows the 3-D model domain. With this mesh generator, we construct an embedded mesh that avoids the need for the fictitious bounding surface boundary conditions used in a nested modelling approach; instead boundary conditions are only applied on the top and bottom surfaces of the spherical shell. Velocity boundary conditions are prescribed to be free slip along the core-mantle boundary and prescribed plate motions along the top surface. Surface plate velocities are extracted every 1 Myr from the plate kinematic reconstructions given by *Gurnis et al.* [2012] using GPlates (`http://www.gplates.org`). Intermediate-age velocities are linearly interpolated from these values. The simulation time for early rifting and break-up of the South Atlantic Ocean spans from 130 to 100 Ma.

The initial thermal structure assumes the lithosphere to be the thermal boundary layer arising from a half-space cooling model. Plate thicknesses for non-cratonic and cratonic continental lithosphere are simulated using ages of 100 and 350 Myr, respectively. This results in a depth for the $1170\,°C$ isotherm of 130 km for non-cratonic continental lithosphere and 245 km for cratonic lithosphere. Craton contours are digitised from *de Wit et al.* [2008]. The model also contains a single 'hot Tristan Plume'. The initial geometry of the plume tail consists of a cylinder of radius 100 km extending from 670 km depth to the bottom of the high-resolution region. The initial thermal structure for the plume is assumed to follow a Gaussian-shaped radial temperature profile with a maximum temperature anomaly of $150\,°C$ with respect to background mantle. Velocity boundary conditions for the plume are implemented by a parabolic-shaped radial velocity profile with the maximum velocity in the centre of the plume tail. The maximum ascent velocity is

$$V_{max} = \frac{2Q_p}{\pi R^2}, \tag{26}$$

where Q_p is the plume flux (km^{3}yr^{-1}) and R is the plume tail radius (km). Here we show a run with a plume flux of 15 km^3 yr^{-1}, consistent with ~20 – 40 mantle plumes supplying an upward return flow to the shallow mantle that balances the ~300 km^3 yr^{-1} downward flux associated with plate subduction [cf. *Yamamoto et al.*, 2007].

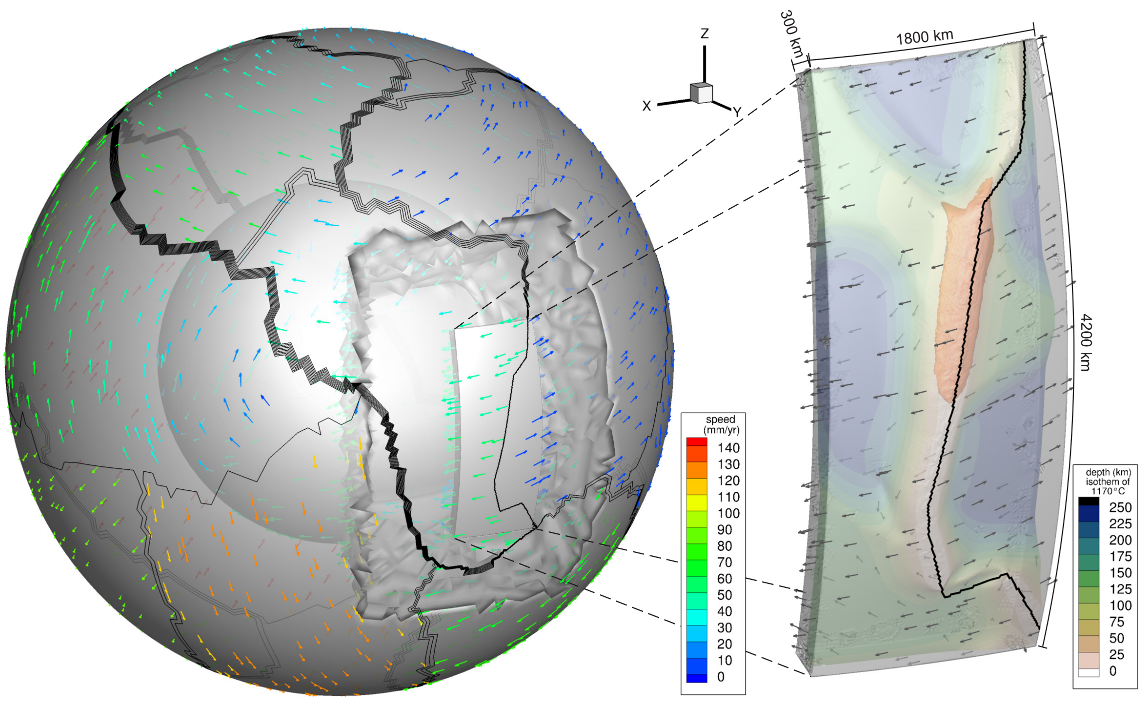

Figure 14: Model domain representing a 3-D spherical shell for the Earth's mantle (left) and zoom for the embedded high-resolution region (right). Black lines represent the plate boundaries. Colour arrows are the speed for the plate motion (left). In the embedded high-resolution region (right) the colour represents the depth of the isosurface of temperature $T = 1170\,^{\circ}C$ and black arrows are the velocity field obtained after imposing the plate motion boundary conditions. Red colour shows plume material with the isosurface of $\log(\eta) = 18.2$, where η is the viscosity.

The calculation also assumes a temperature and pressure dependent upper mantle rheology given by

$$\eta(T,p) = \eta_0 exp\left[\frac{1}{RT_0}\left(E_a\left(\frac{T_0}{T} - 1\right) + pV_a\right)\right], \qquad (27)$$

where η_0 = 2 x 10^{18} Pa·s is the reference viscosity, R is the universal gas constant, $T_0 = 1300\,^{\circ}C$ is the reference mantle temperature, E_a = 400 kJ/mol is the activation energy, T is the temperature, p is the pressure and V_a = 4 x 10^{-6} m^3/mol is the activation volume [*Hirth and Kohlstedt*, 2003]. The minimum and maximum cut-off viscosities are 10^{18} Pa·s and 10^{23} Pa·s, respectively. For simplicity, the lower mantle viscosity is considered to be uniform with a value of 5x10^{21} Pa·s.

Figure 15 shows the 3-D evolution of a model where the plume flux is 15 km^3 yr^{-1}, located at colatitude = 118$\,^{\circ}$ and longitude = 354$\,^{\circ}$. In this experiment hotter, weaker, plume material preferentially migrates southwards. This preferential southward flow appears to be mainly due to the presence of thicker São Francisco and conjugate Congo cratonic roots in the North combined with 'suction' associated with early plate stretching in the non-cratonic regions [*Taramon*, 2018]. For this calculation, the use of local higher-resolution embedded mesh allowed us to take a more computationally efficient approach involving a problem with only $7 \cdot 10^6$ flow unknowns, instead of the $2 \cdot 10^9$ unknowns for a similar resolution global mesh. We were also able to apply 'easy-to-characterize' global plate motion boundary conditions at the surface of the global mesh instead of being forced to apply a more restrictive nested approach in which the effects of the Tristan plume are potentially not considered in the determination of the time dependent flow boundary conditions along the interior surfaces of the nested high-resolution region.

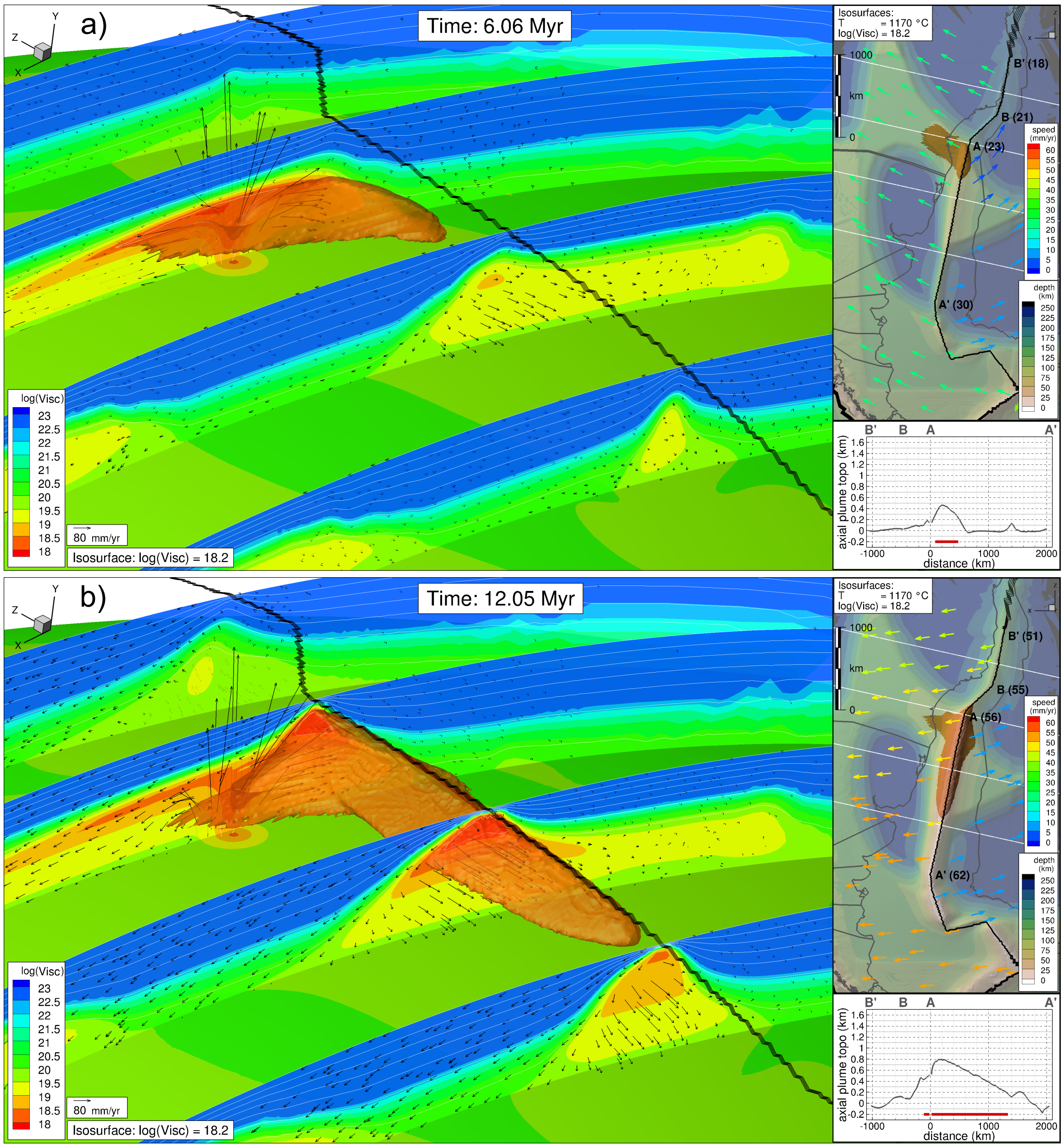

Figure 15: 3-D evolution of plume material during rifting and break-up of the South Atlantic.

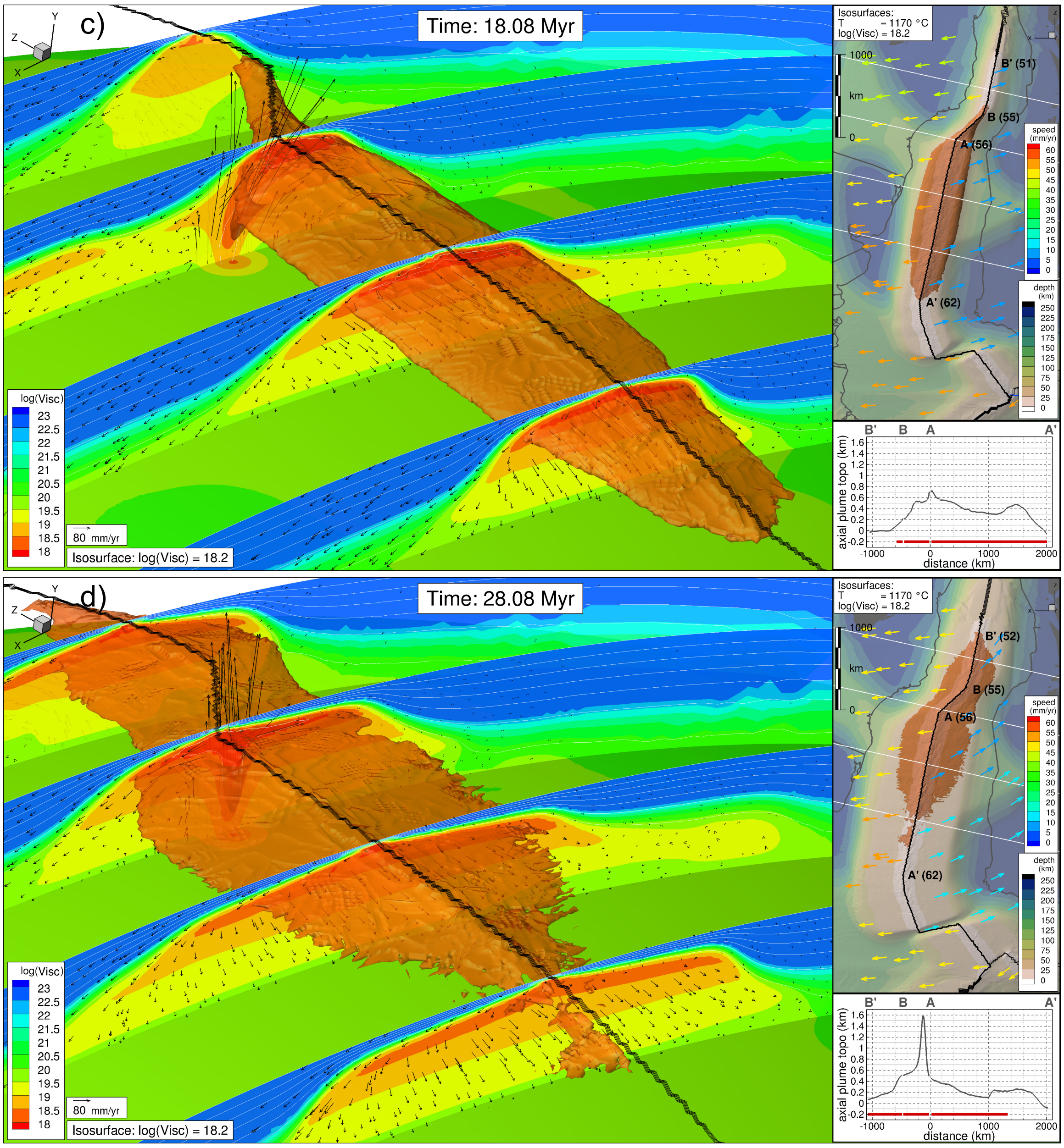

Figure 15 (Cont.): 3-D evolution of plume material during rifting and break-up of the South Atlantic. Main: 3-D images showing the geometry of the plume material at different times: (a) 6.06 Myr, (b) 12.05 Myr, (c) 18.08 Myr and (d) 28.08 Myr. Colour represents logarithm of viscosity. The velocity field and isotherms every 200 $°C$ in the vertical cross sections are represented by arrows and white lines, respectively. Black line represents the plate boundary. The plume material is represented by the red isosurface with $\log(\eta) = 18.2$. Top inset: Top view of the 3-D evolution shown in the main inset. The isosurface with a temperature of 1170 $°C$ is coloured with depth to show the lithospheric thickness variations. The plume material is represented by the red isosurface with $\log(\eta) = 18.2$. Colour arrows represent the top surface plate motion. Grey lines and black thick lines represent the reconstructed coastlines and the plate boundaries, respectively. Capital letters show the ends of the along ridge profiles shown in bottom inset. Numbers between parentheses show the full opening speed in mm/yr. Bottom inset: Plume contribution to axial topography in km. The red horizontal line represents the plume material beneath the ridge profile.

5 Discussion and comparison with other algorithms

In 2-D, there are many open source and commercial mesh generators that are flexible and work well, e.g. Triangle [*Shewchuk*, 1996], so we will not further discuss the 2-D version of our code here. In general, comparing meshes created by different algorithms is a complex task because typically each algorithm creates a mesh with desirable characteristics for a specific problem. For this reason, we test our 3-D algorithm by creating a simple geometry that can be easily reproduced by each compared algorithm. We compare the performance of three additional algorithms (ADP3D [*Dompierre et al.*, 1998], DistMesh [*Persson and Strang*, 2004] and Netgen [*Schöberl*, 1997]) to create a unit-radius sphere with a preferred nodal distance $l_0 = 0.2$. DistMesh, Netgen and this study's algorithm were run on the same machine. For ADP3D we only have the benchmark published by *Dompierre et al.* [1998]. Table 2 shows the number of nodal degrees of freedom (dofs), the number of mesh elements, the computational time (in sec on our 3.2 GHz Intel Core i5 (MacOSX 10.12.5) machine with 24 GB of 1600 MHz memory) and several tetrahedral shape-quality measures [cf. *Dompierre et al.*, 1998] for each algorithm:

- Quality factor q_{3D} given by equation (20), also known as the radius ratio ρ.

Table 2: Statistical data.

		ADP3D [1]	**DistMesh** [2]	**Netgen** [3]	**This study**
Number of nodal dofs		3132	3207	3453	3498
Number of elements		4905	5177	4942	5230
Computational time (s)		- -	16.51	3.46	2.77
Quality factor q_{3D}	min	0.324	0.045	0.501	0.486
	mean	0.873	0.898	0.793	0.881
Aspect ratio γ	min	0.317	0.038	0.457	0.405
	mean	0.772	0.796	0.684	0.773
Mean ratio η	min	0.501	0.112	0.624	0.560
	mean	0.898	0.915	0.832	0.899
Solid angle θ_{min}	min	0.178	0.044	0.185	0.232
	mean	0.665	0.737	0.525	0.686

[1][*Dompierre et al.*, 1998] [2][*Persson and Strang*, 2004] [3][*Schöberl*, 1997]

- Aspect ratio γ given by

$$\gamma = 2\sqrt{6}\frac{r_s}{l_{max}}, \tag{28}$$

where l_{max} is the length of the longest edge of the tetrahedron.

- Mean ratio η given by

$$\gamma = \frac{12\sqrt[3]{9v^2}}{\sum_{i=1}^{6} l_i^2}, \tag{29}$$

where l_i is the length of each edge of the tetrahedron.

- Solid angle θ_{min} given by

$$\theta_{min} = \beta \min\left[\sin\left(\frac{\theta_i}{2}\right)\right], \tag{30}$$

$$\sin(\theta_i/2) = 12v\left(\prod_{\substack{j,k\neq i\\ 0\leq j<k\leq 3}} \left(\left(l_{ij}+l_{ik}\right)^2 - l_{jk}^2\right)\right)^{-1/2}, \tag{31}$$

where $\beta^{-1} = \sqrt{6}/9$.

The histograms corresponding to these data are shown in Figure 16. The new approach, in MATLAB code, creates a comparable mesh in 20% less time than the compiled Netgen C++ code with Python interface and six times faster than the Distmesh MATLAB code (Table 2). Distmesh produces slightly higher *mean* values for all shape measures. However, since it does not include routines to deal with slivers, it also produces the *lowest* minimum values for all shape measures, e.g. it always produces a few sliver-like 'bad' elements. Netgen gives the highest minimum value for all shape measures except for the solid angle. However, its mean values are the lowest. ADP3D produces comparable, but slightly lower minimum and mean values for all the shape measures to our MATLAB codetool. A missing feature from our code is that there is no graphical user interface (GUI). For the meshes we wished to make, it is relatively easy to define a guide-mesh with a few lines of MATLAB code. Since our plan is to combine this algorithm with an adaptive finite element code, it was not a high priority to create an associated GUI. Distmesh is also written in MATLAB, so a user could straightforwardly modify the original Distmesh code to include our guide-mesh approach and tetrahedral shape-improvement routines. The other algorithms would require larger modifications to similarly control the mesh refinement associated with mesh generation and include embedded high-resolution sub-regions.

6 Summary and conclusions

We have developed the tools for generating unstructured meshes in 2-D, 3-D, and spherical geometries that contain embedded high-resolution sub-regions. For the generation of a Cartesian 3-D mesh, only small modifications to the 3-D spherical code would be needed to place boundary points along linear boundary

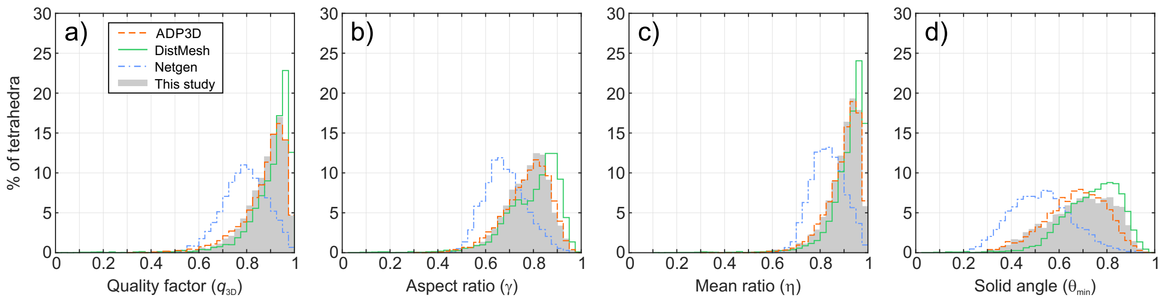

Figure 16: Histogram of the fraction of elements as a function of: (a) quality factor, (b) aspect ratio, (c) mean ratio and (d) solid angle.

edges and planar boundary surfaces in place of spherical shell boundaries. The algorithm uses the FEM to solve for nodal locations that would arise if they were connected by virtual springs with prescribed preferred lengths. Straight line, circular and spherical boundary conditions are imposed to constrain the shape of mesh boundaries. A guide-mesh is used to smoothly refine the mesh around higher resolution sub-regions. Methods to achieve the expected nodal density and further improve element shape and quality are also discussed in detail. Comparison to other open source mesh generators shows that our algorithm generates the highest quality mesh, i.e. the highest minimum and mean value for all the shape measures, with the fastest computational time. The new mesh generator can be easily modified for adaptive mesh refinement by varying the desired spring length depending on solution variables of interest. Since an adaptive refinement (or coarsening) would often only change node positions in regions where the spatial resolution is changed, most nodes of the spring system would remain in equilibrium so that few iterations would be required to update the mesh. We have demonstrated the utility of this approach for geodynamic modelling by showing its application solving for 3-D spherical mantle flow associated with mantle plume-rift interactions, with use of a global spherical shell mesh that contains an embedded high-resolution sub-region.

Code availability

The mesh generator code is open source and available to download at `https://data.mendeley.com/datasets/nkr6p8ndtd/1`. This site also contains the input files to create the meshes discussed in this paper.

Acknowledgments

We thank Cornell University for supporting the initial work on this problem by Morgan and Shi, and the COMPASS Consortium for the Ph.D. support for Jorge M. Taramón.

Supporting information

The following supporting information is available as part of the online article:

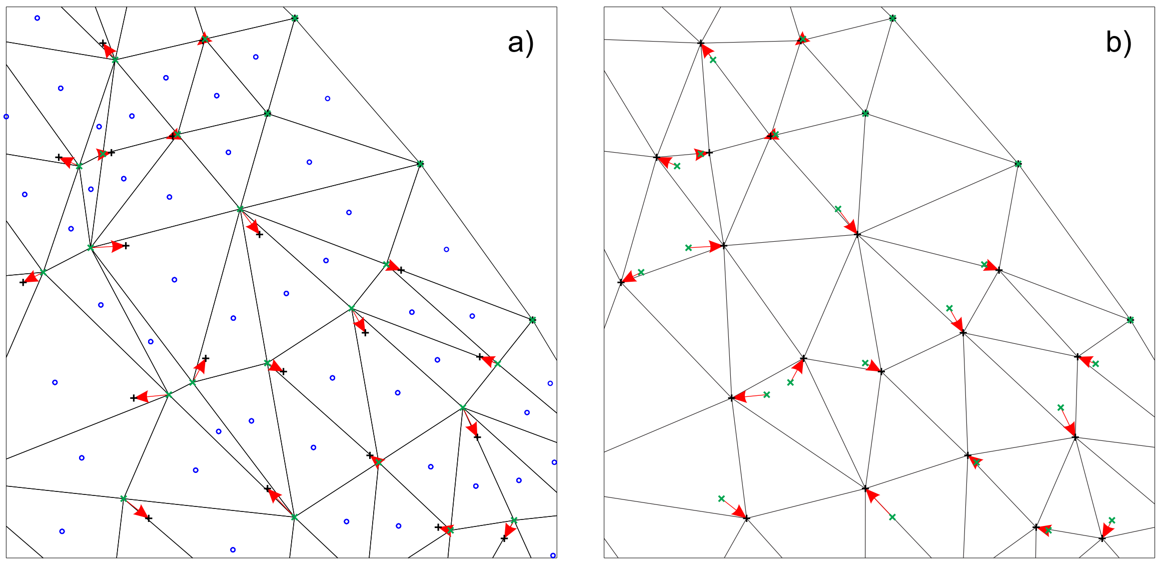

Figure S1: (a) Initial 2-D mesh. (b) Mesh after applying the Laplacian correction to smooth positions of its interior nodes. Blue points are the barycentres of the triangles. Green and black crosses are the nodal positions before and after smoothing, respectively. Red arrows indicate the motions of interior nodes.

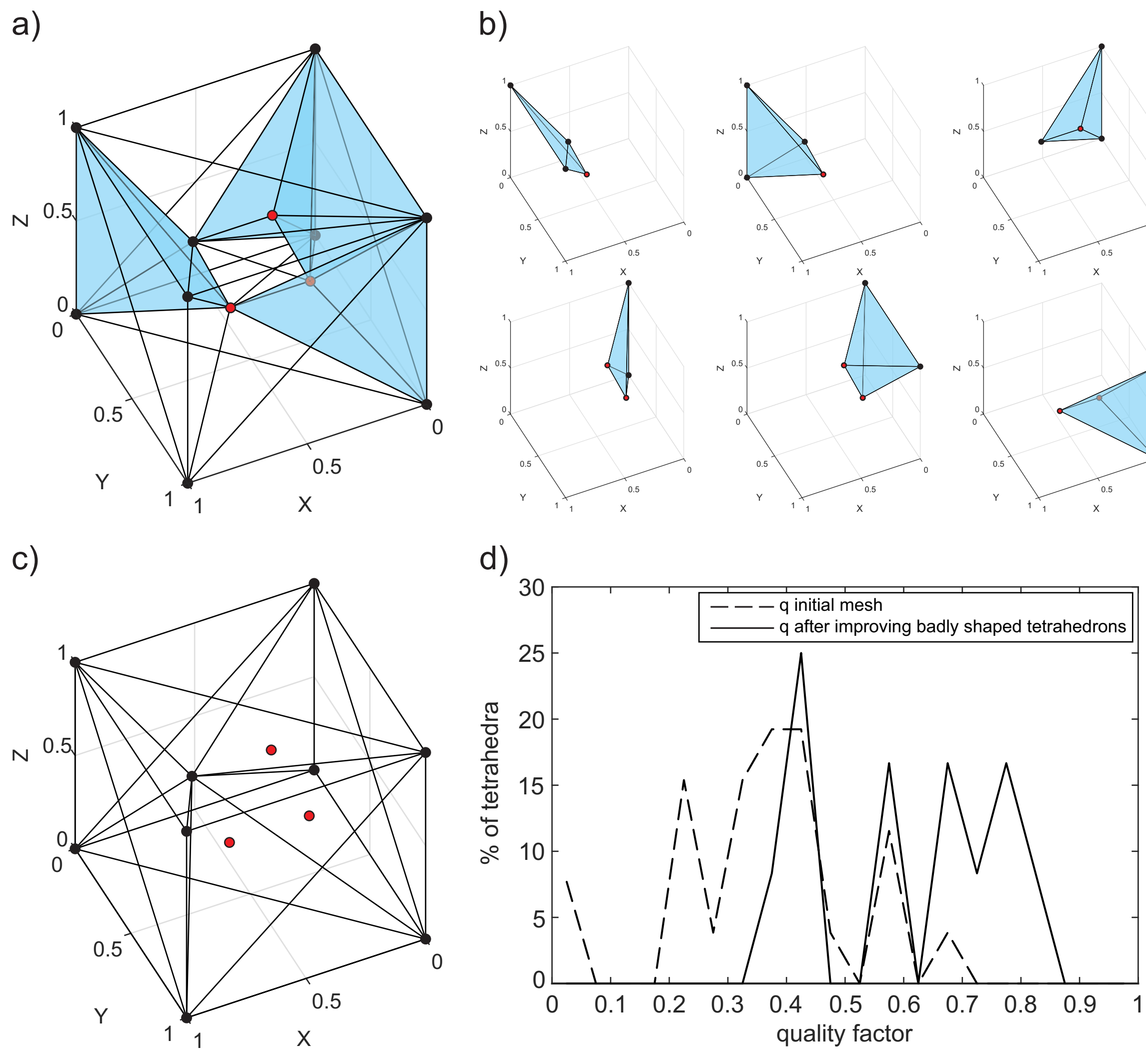

Figure S2: (a) Initial mesh with badly shaped tetrahedra (in blue). Rejected nodes in red. (b) Badly shaped tetrahedra. (c) Mesh after improving badly shaped tetrahedra contains no badly shaped tetrahedra. (d) Fraction of tetrahedra for a given quality factor for both before (dashed line) and after (solid line) local improvements to the shape of badly shaped tetrahedra. The minimum quality factor for the initial mesh is 0.04 and for the final mesh is 0.39.

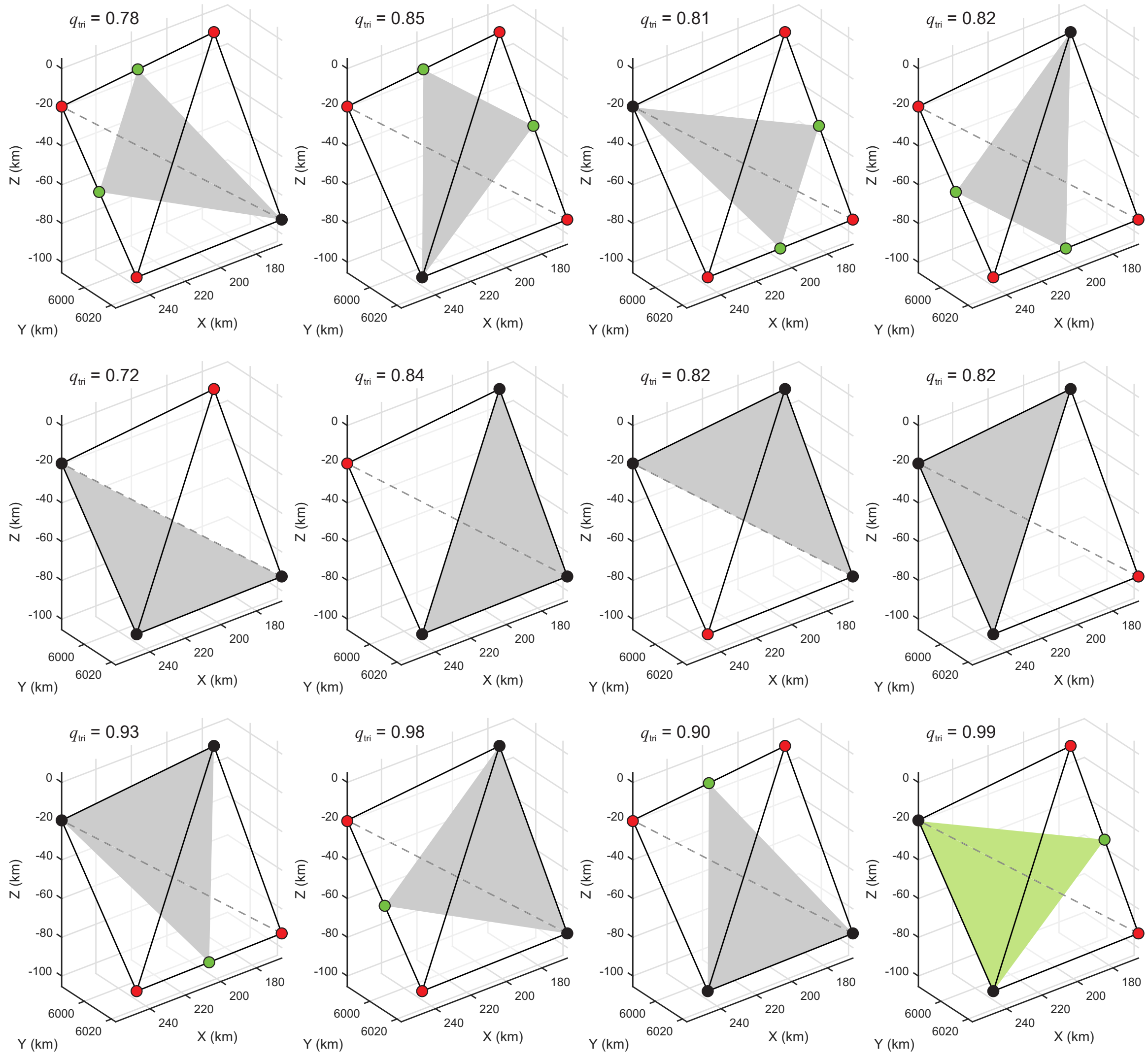

Figure S3: Removing a sliver (represented by black lines and dashed grey line for hidden edge). Possible triangles (grey and green colours) created from permutations of the vertices and midpoints of the edges of a sliver. Black, red and green points represent unaltered, removed and added nodes, respectively. q_{tri} is the quality factor for each triangle. The four vertices of the sliver are replaced by the three mesh points of the potential triangle with the best quality factor (green colour).

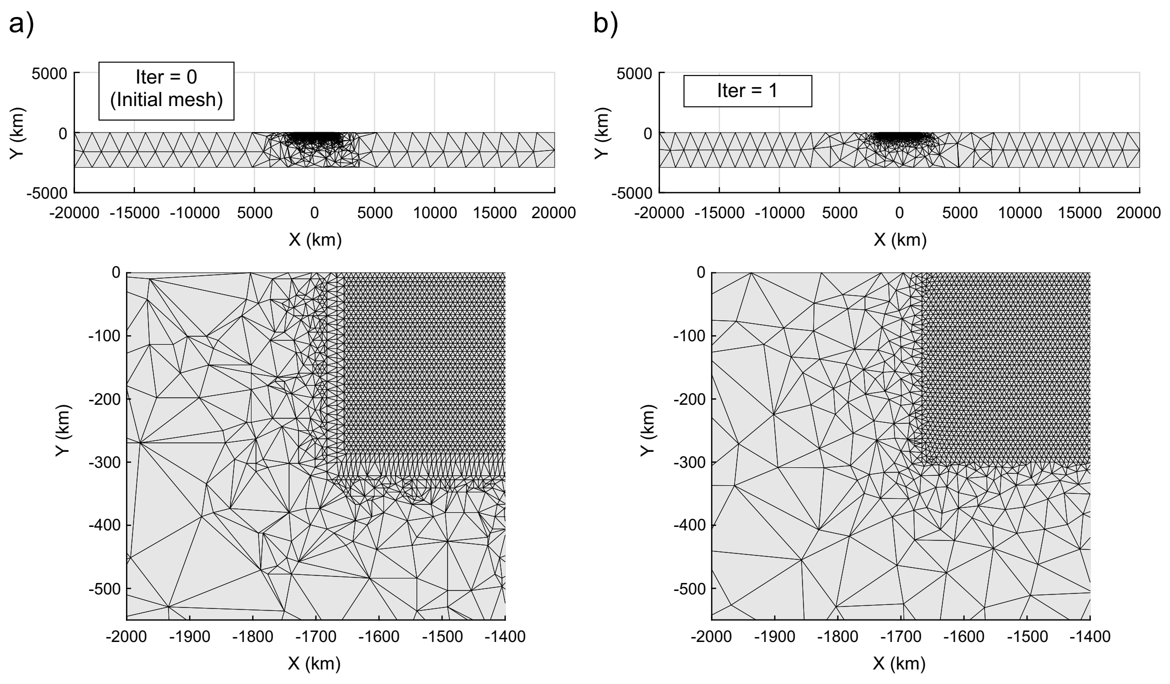

Figure S4: (a) Initial mesh (top) for a rectangular box with an embedded high-resolution sub-region and a zoom around the left boundary of the refined region (bottom). (b) Mesh (top) and zoom (bottom) after the first iteration.

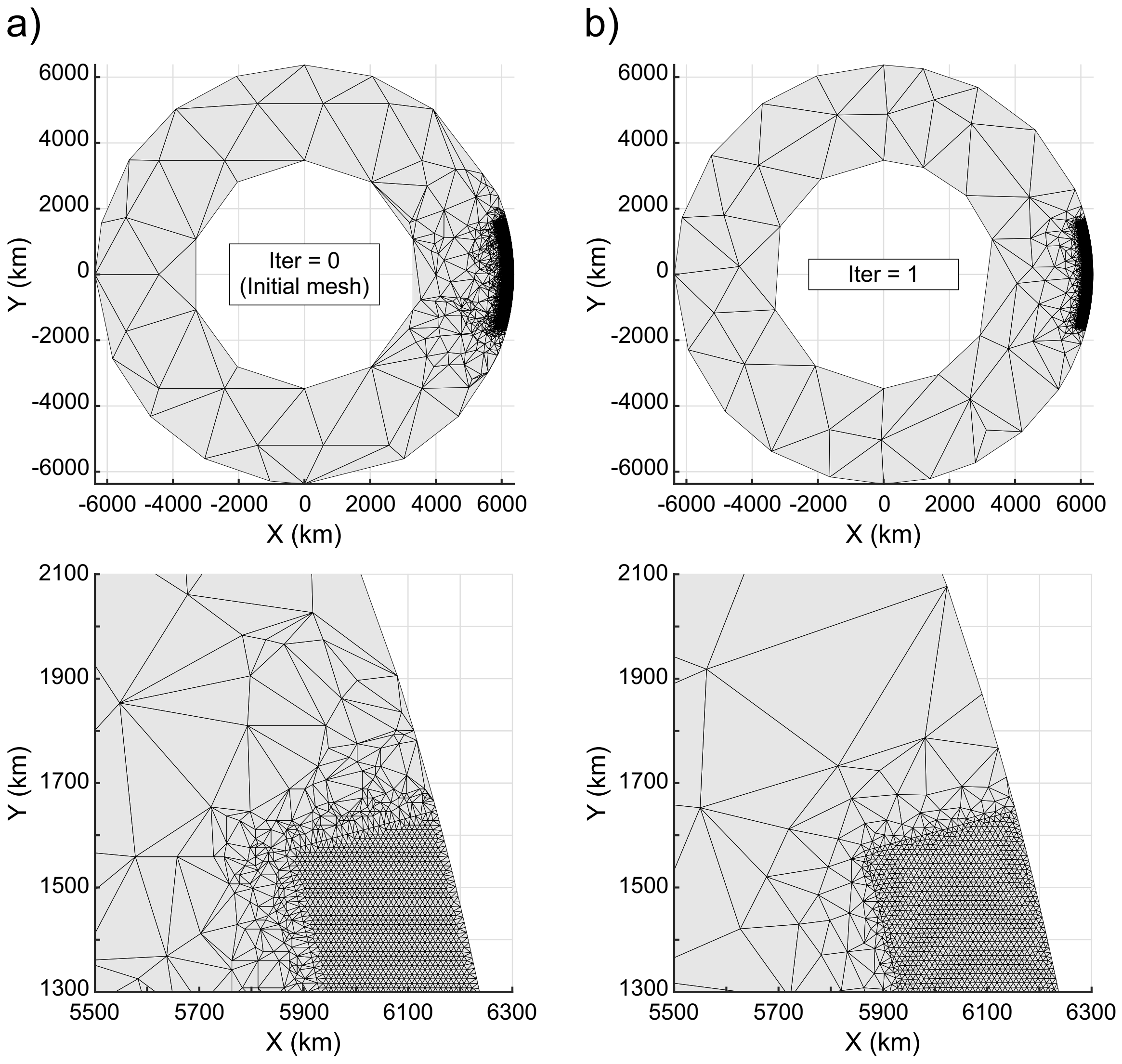

Figure S5: (a) Initial mesh (top) for a cylindrical annulus with an embedded high-resolution sub-region and a zoom around an edge of the refined region (bottom). (b) Mesh (top) and zoom (bottom) after the first iteration.

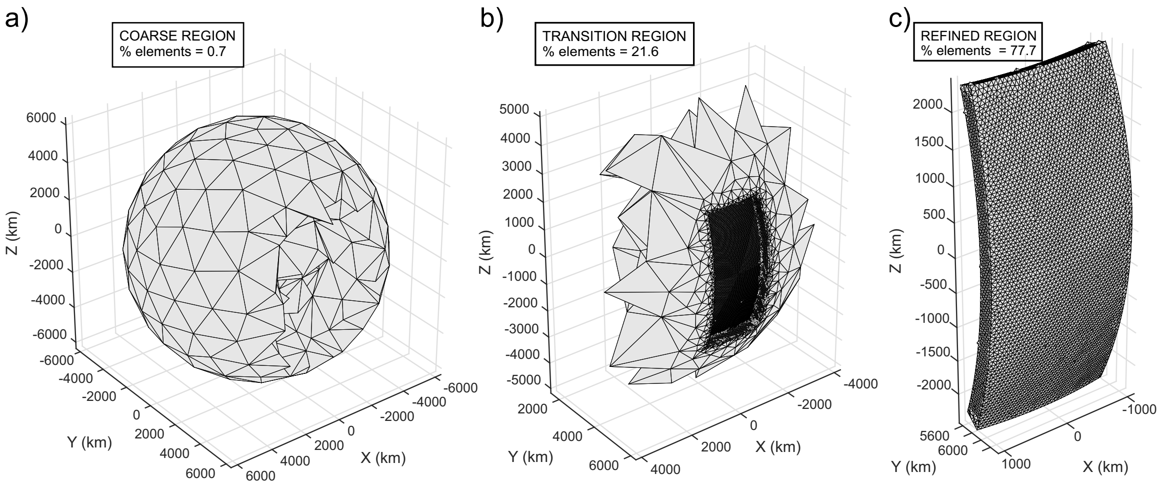

Figure S6: (a) Tetrahedra within the coarse region. (b) Tetrahedra within the transition region. (c) Tetrahedra within the refined region.

A: Derivation of equation (5)

The 2-D development of equation (3), rewritten here for convenience

$$\begin{pmatrix} f_1' \\ f_2' \end{pmatrix} + k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 0 \\ l_0 \end{pmatrix} = k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix}, \tag{A.1}$$

is given by two steps. First, develop the right-hand side of equation (A.1) by writing local coordinates as a function of global coordinates (see Figure 1a)

$$\begin{aligned} k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} & \\ &= k \begin{bmatrix} x_2' - x_1' \\ -(x_2' - x_1') \end{bmatrix} \\ &= k \begin{bmatrix} \left[(x_2 - x_1)c_\alpha + (y_2 - y_1)s_\alpha\right] \\ -\left[(x_2 - x_1)c_\alpha + (y_2 - y_1)s_\alpha\right] \end{bmatrix} \\ &= k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 c_\alpha + y_1 c_\alpha \\ x_2 c_\alpha + y_2 c_\alpha \end{bmatrix} \\ &= k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ 0 & 0 & \cos\alpha & \sin\alpha \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix}, \end{aligned} \tag{A.2}$$

where $s_\alpha \equiv \sin\alpha$ and $c_\alpha \equiv \cos\alpha$. Second, express the global coordinates of the force vector as a function of local coordinates (see Figure 1a)

$$\begin{pmatrix} f_{1,x} \\ f_{1,y} \\ f_{2,x} \\ f_{2,y} \end{pmatrix} = \begin{bmatrix} c_\alpha & 0 \\ s_\alpha & 0 \\ 0 & c_\alpha \\ 0 & s_\alpha \end{bmatrix} \begin{pmatrix} f_1' \\ f_2' \end{pmatrix}. \tag{A.3}$$

Combining equations (A.1) and (A.2) gives

$$\begin{pmatrix} f_1' \\ f_2' \end{pmatrix} = k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ 0 & 0 & \cos\alpha & \sin\alpha \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} - k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 0 \\ l_0 \end{pmatrix}. \tag{A.4}$$

Substituting equation (A.4) into equation (A.3) and reordering gives

$$k\begin{bmatrix} c_\alpha & 0 \\ s_\alpha & 0 \\ 0 & c_\alpha \\ 0 & s_\alpha \end{bmatrix}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ 0 & 0 & \cos\alpha & \sin\alpha \end{bmatrix}\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} f_{1,x} \\ f_{1,y} \\ f_{2,x} \\ f_{2,y} \end{pmatrix} + k\begin{bmatrix} c_\alpha & 0 \\ s_\alpha & 0 \\ 0 & c_\alpha \\ 0 & s_\alpha \end{bmatrix}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{pmatrix} 0 \\ l_0 \end{pmatrix}, \tag{A.5}$$

which is equivalent to equation (5).

B: Derivation of equation (25)

The 3-D development of equation (3), rewritten here for convenience

$$\begin{pmatrix} f_1{}' \\ f_2{}' \end{pmatrix} + k\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{pmatrix} 0 \\ l_0 \end{pmatrix} = k\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{pmatrix} x_1{}' \\ x_2{}' \end{pmatrix}, \tag{B.1}$$

also involves two steps. First, develop the right-hand side of equation (B.1) by writing local coordinates as a function of global coordinates (see Figure 1b)

$$\begin{aligned} k\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{pmatrix} x_1{}' \\ x_2{}' \end{pmatrix} & \\ &= k\begin{bmatrix} x_2{}' - x_1{}' \\ -(x_2{}' - x_1{}') \end{bmatrix} \\ &= k\begin{bmatrix} \Big(\big[(x_2 - x_1)c_\beta + (y_2 - y_1)s_\beta\big]c_\alpha + (z_2 - z_1)s_\alpha\Big) \\ -\Big(\big[(x_2 - x_1)c_\beta + (y_2 - y_1)s_\beta\big]c_\alpha + (z_2 - z_1)s_\alpha\Big) \end{bmatrix} \\ &= k\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} x_1 c_\alpha c_\beta + y_1 c_\alpha s_\beta + z_1 s_\alpha \\ x_2 c_\alpha c_\beta + y_2 c_\alpha s_\beta + z_2 s_\alpha \end{bmatrix} \\ &= k\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta & s_\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\alpha c_\beta & c_\alpha s_\beta & s_\alpha \end{bmatrix}\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix}, \end{aligned} \tag{B.2}$$

where $s_\alpha \equiv \sin\alpha$, $c_\alpha \equiv \cos\alpha$, $s_\beta \equiv \sin\beta$ and $c_\beta \equiv \cos\beta$. Second, express the global coordinates of the force vector as a function of local coordinates (see Figure 1b)

$$\begin{pmatrix} f_{1,x} \\ f_{1,y} \\ f_{1,z} \\ f_{2,x} \\ f_{2,y} \\ f_{2,z} \end{pmatrix} = \begin{bmatrix} c_\alpha c_\beta & 0 \\ c_\alpha s_\beta & 0 \\ s_\alpha & 0 \\ 0 & c_\alpha c_\beta \\ 0 & c_\alpha s_\beta \\ 0 & s_\alpha \end{bmatrix} \begin{pmatrix} f_1{}' \\ f_2{}' \end{pmatrix}. \tag{B.3}$$

Combining equations (B.1) and (B.2) gives

$$\begin{pmatrix} f_1{}' \\ f_2{}' \end{pmatrix} = k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta & s_\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\alpha c_\beta & c_\alpha s_\beta & s_\alpha \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} - k \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 0 \\ l_0 \end{pmatrix}. \tag{B.4}$$

Substituting equation (B.4) into equation (B.3) and reordering gives

$$k \begin{bmatrix} c_\alpha c_\beta & 0 \\ c_\alpha s_\beta & 0 \\ s_\alpha & 0 \\ 0 & c_\alpha c_\beta \\ 0 & c_\alpha s_\beta \\ 0 & s_\alpha \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta & s_\alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\alpha c_\beta & c_\alpha s_\beta & s_\alpha \end{bmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} f_{1,x} \\ f_{1,y} \\ f_{1,z} \\ f_{2,x} \\ f_{2,y} \\ f_{2,z} \end{pmatrix} + k \begin{bmatrix} c_\alpha c_\beta & 0 \\ c_\alpha s_\beta & 0 \\ s_\alpha & 0 \\ 0 & c_\alpha c_\beta \\ 0 & c_\alpha s_\beta \\ 0 & s_\alpha \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} 0 \\ l_0 \end{pmatrix}, \tag{B.5}$$

which is equivalent to equation (9).

References

Alliez, P., D. Cohen-Steiner, M. Yvinec, and M. Desbrun (2005), Variational tetrahedral meshing, *ACM Trans. Graph.*, *24*(3), 617–625, doi:10.1145/1073204.1073238.

Anderson, A., X. Zheng, and V. Cristini (2005), Adaptive unstructured volume remeshing - I: The method, *J. Comput. Phys.*, *208*(2), 616–625, doi:10.1016/j.jcp.2005.02.023.

Bern, M., D. Eppstein, and J. Gilbert (1994), Provably good mesh generation, *J. Comput. Syst. Sci.*, *48*(3), 384–409, doi:10.1016/S0022-0000(05)80059-5.

Burstedde, C., O. Ghattas, M. Gurnis, G. Stadler, E. Tan, T. Tu, L. C. Wilcox, and S. Zhong (2008), Scalable adaptive mantle convection simulation on petascale supercomputers, 2008 SC - Int. Conf. High Perform. Comput. Networking, Storage Anal., pp. 1–15, doi:10.1109/SC.2008.5214248.

Cheng, S., and T. Dey (2002), Quality meshing with weighted Delaunay refinement, *Proc. Thirteen. Annu. ACM-SIAM Symp. Discret. algorithms*, *33*(1), 137–146, doi:10.1137/S0097539703418808.

Cheng, S., T. Dey, H. Edelsbrunner, M. A. Facello, and S. Teng (2000), Sliver exudation, *J. ACM*, *47*(5), 883–904, doi:10.1145/355483.355487.

Chew, L. P. (1989), Guaranteed-Quality Triangular Meshes, *Tech. rep.*, Department of Computer Science, Cornell University, Ithaca, New York.

Chew, L. P. (1997), Guaranteed-Quality Delaunay Meshing in 3D (short version), *Proc. Thirteen. Annu. Symp. Comput. Geom.*, pp. 391–393, doi:10.1145/262839.263018.

Choi, W., D. Kwak, I. Son, and Y. Im (2003), Tetrahedral mesh generation based on advancing front technique and optimization scheme, *Int. J. Numer. Methods Eng.*, *58*(12), 1857–1872, doi:10.1002/nme.840.

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

de Wit, M. J., J. Stankiewicz, and C. Reeves (2008), Restoring pan-african-brasiliano connections: more gondwana control, less trans-atlantic corruption, *Geol. Soc. London, Spec. Publ.*, *294*(1), 399–412, doi:10.1144/SP294.20.

Dompierre, J., P. Labbé, F. Guibault, and R. Camarero (1998), Proposal of benchmarks for 3D unstructured tetrahedral mesh optimization, 7th Int. Meshing Roundtable, pp. 525–537.

Edelsbrunner, H., and D. Guoy (2002), An Experimental Study of Sliver Exudation, *Eng. Comput.*, *18*, 229–240, doi:10.1007/s003660200020.

Gurnis, M., M. Turner, S. Zahirovic, L. DiCaprio, S. Spasojevic, R. Müller, J. Boyden, M. Seton, V. C. Manea, and D. J. Bower (2012), Plate tectonic reconstructions with continuously closing plates, *Comput. Geosci.*, *38*(1), 35–42, doi:10.1016/j.cageo.2011.04.014.

Hirth, G., and D. Kohlstedt (2003), Rheology of the upper mantle and the mantle wedge: A view from the experimentalists, in *Insid. subduction Fact.*, edited by J. Eiler, pp. 83–105, American Geophysical Union, Washington, D. C., doi:10.1029/138GM06.

Ito, Y., A. M. Shih, and B. K. Soni (2004), Reliable Isotropic Tetrahedral Mesh Generation Based on an Advancing Front Method, 13th Int. Meshing Roundtable, pp. 95–105.

Ito, Y., A. M. Shih, and B. K. Soni (2009), Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates, *Int. J. Numer. Methods Eng.*, *77*, 1809–1833, doi:10.1002/nme.2470.

Kronbichler, M., T. Heister, and W. Bangerth (2012), High accuracy mantle convection simulation through modern numerical methods, *Geophys. J. Int.*, *191*(1), 12–29, doi:10.1111/j.1365-246X.2012.05609.x.

Labelle, F., and J. R. Shewchuk (2007), Isosurface stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles, *ACM Trans. Graph.*, *26*(3), doi:10.1145/1276377.1276448.

Li, X., and S. Teng (2001), Generating well-shaped Delaunay meshed in 3D, 12th Annu. ACM-SIAM Symp. Discret. algorithms, pp. 28–37, Washington, D. C.

Löhner, R., and P. Parikh (1988), Generation of three-dimensional unstructured grids by the advancing-front method, *Int. J. Numer. Methods Fluids*, *8*(10), 1135–1149, doi:10.1002/fld.1650081003.

Mitchell, S. A., and S. A. Vavasis (1992), Quality mesh generation in three dimensions, Proc. Eighth Annu. Symp. Comput. Geom. ACM, pp. 212–221, doi:10.1145/142675.142720.

Persson, P., and G. Strang (2004), A Simple Mesh Generator in MATLAB, *SIAM Rev.*, *46*(2), 329–345, doi:10.1137/S0036144503429121.

Ruppert, J. (1995), A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation, *J. Algorithms*, *18*, 548–585, doi:10.1006/jagm.1995.1021.

Schöberl, J. (1997), An advancing front 2D/3D-mesh generator based on abstract rules, *Comput. Vis. Sci.*, *1*(1), 41–52, doi:10.1007/s007910050004.

Shewchuk, J. R. (1996), Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, in *Lin, Ming C and Manocha, Dinesh*, *Lecture Notes in Computer Science*, vol. 1148, edited by A. C. G. T. G. Eng., pp. 203–222, Springer, Berlin, doi:10.1007/BFb0014497.

Shewchuk, J. R. (1998), Tetrahedral mesh generation by Delaunay refinement, 14th Annu. Symp. Comput. Geom. SCG '98, pp. 86–95, doi:10.1145/276884.276894.

Shewchuk, J. R. (2002), What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures, Elev. Int. Meshing Roundtable, pp. 115–126, doi:10.1.1.68.8538.

Si, H. (2015), TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator, *AMC Trans. Math. Softw.*, *41*(2), doi:10.1145/2629697.

Taramon, J. M. (2018), 3-d spherical high-resolution modelling of south atlantic rifting-related mantle flow, Ph.D. thesis, Royal Holloway University of London.

Yamamoto, M., J. P. Morgan, and W. J. Morgan (2007), Global plume-fed asthenosphere flow–I: Motivation and model development, in *Plates, Plumes Planet. Process.*, vol. 430, edited by D. M. Foulger, G. R. Jurdy, pp. 165–188, Spec. Pap. Geol. Soc. Am.