Introduction

Partial differential equation (PDE) solvers based on the finite-difference (FD) method have for many years been a keystone of seismic processing and modelling applications, commonly found in practical migration and full-waveform inversion methods. Their suitability in such contexts stems from their comparative simplicity, along with relatively low memory and computational requirements, versus other common solvers (Liu and Sen, 2009).

Accuracy is a principal concern when finding a discretized solution to any PDE. In the case of the FD method, there are two broad routes through which the accuracy of a solution is typically refined: increased discretization order (derivation for any formal accuracy is trivial (Liu and Sen, 2009)), and optimized schemes devised to minimize numerical dispersion (e.g. Tam & Webb, 1993; Zhang & Yao, 2013). However, the presence of a sharp density contrast within the computational domain has potential to nullify any such efforts to minimize numerical error in the solution: a particularly acute shortcoming considering the inclusion of topography in a seismic model (Mulder, 2017).

Including a simple step change in density to approximate an air layer (typically referred to as the “air-layer” or “vacuum” approach) is one straightforward method for incorporating boundaries into a seismic model. Although this may seem a tempting solution, it compromises both stability and numerical accuracy, often requiring a smoothing of the contrast to prevent the solution blowing up (Zeng et al., 2012; Mulder, 2017), and inducing both first and second order errors in space (Zhebel et al., 2014; Mulder, 2017). The first-order component of the latter arises from the “staircasing” effect induced by mapping of the boundary surface to the nearest grid nodes (assuming sampling of the subsurface model as opposed to averaging) (Zhebel et al., 2014). The boundary position relative to the grid increment is consequently ambiguous (Mulder, 2017). The non-differentiable nature of sharp interfaces produces the second-order error component (Mulder, 2017).

The necessity of more sophisticated topography handling, ideally incorporated as a free-surface boundary, is thus clear if such errors are to be avoided (Gao et al., 2015). Free surface boundaries of arbitrary shape can be implemented via the immersed boundary method. By constructing such a boundary conforming to topography, it is possible to accurately encapsulate the behaviour of seismic waves, whilst minimizing the numerical error introduced by topography (Gao et al., 2015; Mulder, 2017). This is achieved through extrapolation of the wavefield across the boundary to find solution values at necessary stencil nodes located externally. As this process is confined to the pre-processing step, it has negligible effect on the computational cost of the simulation (Lombard, Piraux and Virieux, 2008). A marked advantage over comparably expensive and complex methods based upon curvilinear grids and other such geometrical transformations is thus clear (Hu, 2016).

Despite the relative simplicity of immersed boundary methods, the additional algorithmic complexity inherent in the inclusion of curvilinear topography remains an unwelcome obfuscation of a simulation’s workings, and a distraction from the overarching task: full waveform inversion or migration. Implementation of an immersed boundary is not an end in itself, and it is therefore sensible to abstract the user from the process of boundary construction, lest their workflow become derailed by the tedious, time-consuming process of implementing and testing a proprietary algorithm. This approach to development is exemplified by Devito: an embedded domain-specific language for finite-difference applications (Luporini et al., 2018; Louboutin et al., 2019). Devitoboundary is a tool in its early stages of development, intended to compliment Devito as a means of including immersed boundaries in practical applications without becoming ensnared in their inner workings. Accessibility of the immersed boundary method is thereby improved, allowing users to include topography in their models in the form of a high-level object. Additionally, workflow is greatly accelerated since users have immediate access to a robust, verified implementation. The current workings of Devitoboundary are presented alongside simple examples to showcase the potential of the tool.
Method

Topography is inputted as an arbitrarily sampled point cloud of elevation. Delaunay triangulation is carried out in the x-y plane, allowing the boundary to be expressed as a polygonal surface. Each triangular plane is then characterized in terms of its equation and bounding edges, allowing for determination of the position of the boundary relative to adjacent nodes in each coordinate direction.

Devitoboundary is based on the method presented in Mulder, 2017, due to its stability and relative simplicity, making it well suited for generalisation and automation. The method utilizes independent 1D extrapolation schemes to modify the finite difference operator in each relevant coordinate direction, thereby sidestepping instability-causing ambiguities which may present themselves in multiple dimensions (Mulder, 2017). To further safeguard against instability, nodes within half a grid increment of the boundary are not used for the extrapolation (Mulder, 2017).

For a scheme of formal order \( M \), an extrapolation matrix of size \( M/2 \times M/2 \) can be calculated. The extrapolation matrix is multiplied by a column vector containing the \( M/2 \) nodes closest to the inner side of boundary, yielding the nearest \( M/2 \) stencil nodes on the exterior in terms of the interior nodes. This can then be combined with the standard stencil coefficients for these exterior points to obtain the necessary modifications to the interior coefficients. The exterior coefficients can then be zeroed. Note that the full matrix is only required if the point at which a derivative is to be calculated lies within one grid increment of the boundary. Only necessary extrapolation matrix rows are calculated for efficiency. Modified stencils are then passed to the FD operator in the form a Devito Function object via Coefficient and Substitutions objects.

Examples

Figure 1 The geometry of the boundary, as interpreted by Devitoboundary from the point cloud supplied. Note that whilst the topography in this case is regularly sampled for clarity and simplicity, irregular sampling incurs no additional computational cost, and is equally valid as an input.
Subfigures A through D show a slice through the wavefield in the plane $y=500$ at 60ms, 68ms, 77ms, and 85ms respectively. The profile of the boundary is clearly outlined in subfigure B by the wave incident upon it. There are no visible numerical artefacts trailing the reflected wave in subfigures C and D, implying that the immersed boundary introduces minimal error, despite only using a 4th order discretization.

To demonstrate the capabilities of Devitoboundary for a simple case, a Ricker wavelet was propagated within a homogenous media. The computational domain consisted of a $126 \times 126 \times 126$ node cube 1000m across, with an additional 50 absorbing layers on each side. A regularly sampled blunted conical immersed boundary was included, with a geometry shown in Figure 1. Wavefield slices through the centre of the cone in the x-z plane are shown in Figure 2.

The wave is shown to completely reflect with no transmission through the immersed boundary, confirming the desired functionality of the implementation. Additionally, the wave behaves naturally upon encountering the sharp kink bounding the edge of the cone’s upper surface, implying that topography which does not vary smoothly can be handled without issue when extended to more complex boundary shapes.
It is apparent that the immersed boundary method provides a remarkably error-free reflection at the internal boundary, and the polygonal representation is capable of accurately representing the topography.

Summary

Accommodation of non-trivial topography in an FD model without inducing unacceptable error has historically presented a significant challenge. Devitoboundary intends to address this in much the same way Devito has simplified the process of implementing the model itself, freeing the user from its potentially labyrinthine inner workings, and allowing for efforts to instead be focused on the task at hand.

The implementation of complex 3D topography from a point cloud via use of immersed boundaries has been highlighted. Integration of the implementation with Devito means features including automatic parallelism can be exploited, ensuring efficient, high-performance code without the associated complexity for the user.

Forthcoming developments include modification to leverage Devito’s subdomain functionality, further augmenting model efficiency and minimizing memory footprint, improved handling of very large topography datasets and enabling boundary construction from multiple sources of survey-scale data in parallel.

References