FAST CUBIT-PYTHON TOOL FOR HIGHLY ACCURATE TOPOGRAPHY GENERATION AND LAYERED DOMAIN RECONSTRUCTION

J. May[1], D. Pera[1], F. Di Michele[2],
R. Aloisio [2,3], B. Rubino[1], P. Marcati [2]

[1] *Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila – via Vetoio, loc. Coppito, I–67100 L'Aquila, Italy*
[2] *Gran Sasso Science Institute (GSSI), via M. Iacobucci 2, 67100 L'Aquila, Italy*
[3] *INFN-Laboratori Nazionali del Gran Sasso, Via G. Acitelli 22, Assergi (AQ), Italy*

February 2021

ABSTRACT

In this paper we present a new tool which can be used to simplify and speed up the reconstruction of real Earth surfaces, cake-layered domains and planar fault sources for numerical simulations. The tool makes use of the CUBIT-Python interface in order to directly 'communicate' and to allow for maximum portability across different operating systems. We will focus the use of the tool to earthquake simulations, although many other types of simulation are able to make use of the same output. Indeed the features created using this software can be applied to numerous other model scenarios, including but not limited to: water flow models, avalanche models, subsurface cavity effects on travelling waves, and earthquake simulations.

Keywords: Domain generation, Surface topography reconstruction, Computational seismology

1. INTRODUCTION

Modern earthquake simulations require a high level of accuracy in domain construction in order to build representative synthetic seismograms. The introduction of a real surface representation into a 3D computational model is important to fully appreciate any effects this has on the travelling waves at surface regions. It is also mandatory to have efficient and portable programming tools in order to both simplify and speed-up the creation of a realistic domain.
The advantages of the procedure automation are: less working user input, the parallelisation of domain and feature creation, and the confidence that the final structures are consistent with the input data. The accuracy of any computational model depends on several features including: the numerical method, the properties of the domain, the underlying equations, the mesh size, and time step selection. Which of these could be considered the most important depends on the project goal and any application of the solution. Depending on the project target, different elements of any 2D or 3D domain will take on more significance. In some applications, such as the modelling of subsurface cavities, [1, 2, 3, 4], the effect of the upper surface of the domain may be considered negligible, however when modelling water flow or avalanches top surface accuracy is incredibly important in predicting which direction of flow. In these cases it is required to be able to construct a surface in a way that guarantees a minimum level of detail in a suitable time frame. It may also be important to use software which allows the po-

tential for other features to be added in the future.
Equally, in seismic models the surface can influence the propagation of the waves through the media [5, 6, 7, 8], for example causing local reflections of energy which can increase peak ground motion at a particular site. Also important in seismic models are the rupture sources, these are fault parts where local slip causes an earthquake and which are calculated post event. These sources may be approximated using a plane of given size, strike, dip and location.

The following work concerns the reconstruction of an accurate computational domain, more specifically any real world surface and subsurface layers, along with fault plane sources which may be used within a seismic model. It will outline a Python tool, made available along with user instructions through contacting the authors, which may be used to create real world surfaces of required accuracy and size. The tool also includes the option of adding any layer(s) to create a 3D domain and the creation of geo-located planar fault sources. A simple way of building surfaces will start with a series of points. One can generate a simple surface using few points using inbuilt commands and these points. These surfaces are useful in numerous cases and models, however there is no in-built method for building a highly irregular real Earth surface. It is possible to import many different file types into CUBIT, however it can then be complicated to clean any leftover features which, coupled with the difficulty in editing mesh based geometry, can cause problems.

In this article we will use earthquake simulations as an example setting for our tool. We designed our application for use with SPEED (`http://speed.mox.polimi.it`) however integration with other software such as SPECFEM3D (`https://geodynamics.org/cig/software/specfem3d`) is simply done. We will describe the methods used to reconstruct surfaces, build computational domains and generate planar surfaces for fault source. Example user input files are shown in an appendix and we will present some timing results for the tool in order to demonstrate both the simplicity of user input and its fast working.
The paper is organized in the following way, in section 2, for the sake of completeness, we introduce some basic mathematical concepts needed to understand the seismological modelling aspects. In section 3 we present and discuss the working of our software. Finally, in section 4 we provide conclusions and other possible applications are suggested.

2. ELASTODYNAMICS EQUATIONS AND SPECTRAL ELEMENT METHOD

In this section we review the mathematical concepts related to seismic wave propagation. For more detail refer to [9, 10, 11].
The standard approach is to use the *Cauchy Equation* (1), first introduced by Augustin-Louis Cauchy, which describes how the i$-th$ component of the momentum u_i (the displacement in our case) changes over time t

$$\rho\frac{\partial^2 u_i}{\partial t^2} = \frac{\partial \sigma_{ik}}{\partial x_k} + \rho f_i, \quad (x_k) \in \Omega \subset \mathbb{R}^3 \text{ and } t \in [0,T], \tag{1}$$

where ρ is the density and f is the body force modelling the seismic source.
The stress tensor σ_{ij} can be written as a function of the displacement as

$$\sigma_{ij} = \lambda\delta_{ij}\frac{\partial u_k}{\partial k} + \mu\left[\frac{\partial u_i}{\partial j} + \frac{\partial u_j}{\partial i}\right], \tag{2}$$

where λ and μ are the Lamé parameters. They are usually expressed in terms of shear wave velocities (v_s) and pressure wave velocities (v_p):

$$\lambda = \rho(v_p^2 - 2v_s^2) \quad \text{and} \quad \mu = \rho v_s.$$

Using (2), the equation (1) becomes

$$\begin{cases} \rho\frac{\partial^2 u}{\partial t^2} = (\lambda+2\mu)\frac{\partial^2 u}{\partial x^2} + (\lambda+\mu)\left[\frac{\partial^2 v}{\partial x\partial y} + \frac{\partial^2 w}{\partial x\partial z}\right] \\ \qquad +\mu\left[\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right] \\ \rho\frac{\partial^2 v}{\partial t^2} = (\lambda+2\mu)\frac{\partial^2 v}{\partial y^2} + (\lambda+\mu)\left[\frac{\partial^2 u}{\partial x\partial y} + \frac{\partial^2 w}{\partial y\partial z}\right] \\ \qquad +\mu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2}\right] \\ \rho\frac{\partial^2 w}{\partial t^2} = (\lambda+2\mu)\frac{\partial^2 w}{\partial z^2} + (\lambda+\mu)\left[\frac{\partial^2 u}{\partial x\partial z} + \frac{\partial^2 v}{\partial y\partial z}\right] \\ \qquad +\mu\left[\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2}\right] \end{cases} \tag{3}$$

In equation (3), for simplicity, we use (x, y, z) instead of (x_i, x_j, x_k) and (u, v, w) instead of (u_x, u_y, u_z).
To solve (3) we need to couple the system with a suitable set of initial and boundary conditions.
As the initial condition we set

$$(u, v, w) = \left(\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t}, \frac{\partial w}{\partial t}\right) = (0, 0, 0), \tag{4}$$

namely we assume that the domain is *at rest* for $\forall x \in \Omega$ at $t = 0$.
For the boundary conditions multiple choices are allowed. Usually, in the context of seismic waves propagation, the topographic surface is treated as a free surface for which we set $\sigma\mathbf{n} = \mathbf{0}$. while the absorbing boundary condition is applied at the other external

faces of the domain, namely $(\sigma\mathbf{n})^* = \mathbf{0}$. The interested reader can refer to [12] to get the expression of $(\sigma\mathbf{n})^*$.

With a system of equations capable of modelling the elastic wave propagation within an elastic media, the next step is the selection of a method capable of solving the continuous system of equations. A common approach is to reduce the continuous system to a finite problem, this can be achieved through the use of numerical methods such as finite differences, finite volumes or finite elements.

The complete treatment of numerical methods for elastic waves is out the scope of this paper. We just remark that many software packages used to simulate elastic wave propagation in elastic media (such as SPEED and SPECFEM3D), are based on the spectral element method [13, 14, 15, 16].

3. GEOLOGICAL DOMAIN AND MESH CREATION

The following section outlines each individual step that the tool can perform. Domains created using this pre-processing tool where used for the earthquake simulator available at `https://www.opendatalaquila.it/APPS/CUIM-sisma/`. Further models using this tool can be seen in [17], where the authors also introduced a new sedimentary basin and compared two known potential sources and a blind test prediction to real seimograms for simulations related to the 6th April 2009 L'Aquila earthquake.

3.1 Auto Creation tool

The following section outlines the Python tool for feature generation and gives a brief explanation of its requirements. The tool takes care of the four main issues when creating a 3D domain for use in seismic modelling, namely: real surface generation (section 3.1.1), adding layers to create a 3D domain (section 3.1.2), creation of fault planes (section 3.1.3), and the calculation of Hypo-center, Hypo-strike and Hypo-dip (section 3.1.4) within the model coordinates. Figure 1 shows a graphical representation of the 3 main parts of the tool with the control script (in red) and their respective input files (in blue). The Hypo-centre information is handled by a separate script which is run independently of the others.

Each part of the tool shown in separate red blocks in figure 1 may be run independently from the others if so desired. However the linking *Main* script offers ease of use for almost complete model creation.

The following sections will describe in more detail the role of each part, its input, output and methods. All

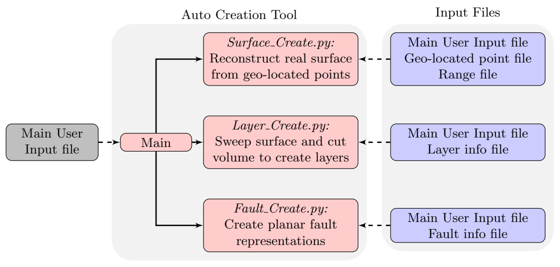
Auto Creation Tool
Input Files
Main User Input file
Main
Surface_Create.py: Reconstruct real surface from geo-located points
Layer_Create.py: Sweep surface and cut volume to create layers
Fault_Create.py: Create planar fault representations
Main User Input file Geo-located point file Range file
Main User Input file Layer info file
Main User Input file Fault info file

Figure 1: Graphical representation of the tool workflow. The red blocks represent tool functions. The blue blocks contain the input files for their connected parts. The grey block at the left of the figure is the only input to *Main.py* and is the same file as read by the other functions of the tool

user input files can be seen in appendix A. Filenames are highlighted in italic-bold and match those given in blue blocks of figure 1.

3.1.1 Surface Creation

The most important of the scripts is the surface creation script, *Surface_Create.py*. This will read input and output file names from ***Main User Input file***, shown in listing 1. From these names it is able to read a ***Geo-located point file*** in *CSV* format, see listing 2 for a partial example, which contains an ordered set of geo-located XYZ points defining a surface. From here we will call the surface created the *primary* surface. The routine will also read from a *CSV* formatted ***Range file***, example in listing 3, which contains the minimum and maximum X and Y points for the primary surface. The output file name read from ***Main User Input file*** is used to control the naming of the saved *.trelis* output file containing the primary surface.

After reading from the input files the routine runs a loop over the dataset and skips any points where the X or Y coordinate lies outside the user defined range. Any point lying within the bounds generates a CUBIT command resulting in the creation of a *vertex* at the defined location. Within the loop is also a check on the first column coordinate, this controls a command to create a *spline* from the vertices created before this point, while any subsequent vertices will now form part of a new *spline* curve. Both the *vertices* and *splines* are generated using the software's inbuilt commands.

The *spline* generation requires that the data be ordered. The first column must be ordered, then the second column ordered within equal values of the first, whether the first column denotes the X or Y coordinate is not important. From a set of splines it is possible to generate a surface using the *skin curve* command. After the removal of the splines what is left is a reconstructed surface from the original data-set and accurate to the inbuilt *spline* and *skin curve* commands.

Figures 2 and 3 show examples of real surfaces reconstructed using this technique, each surface is roughly centered around the city of L'Aquila and has been generated from a 400m resolution data set which is available from [18].

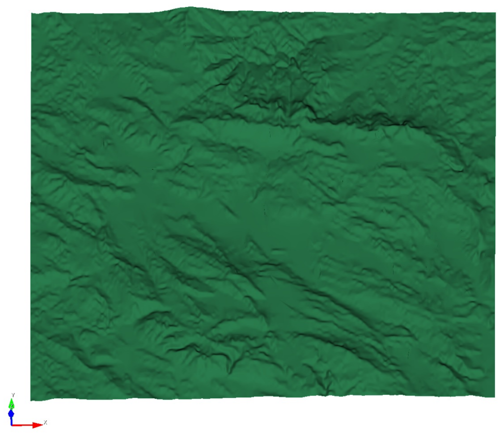

Figure 2: Real surface reconstructed from 21750 geo-located points surrounding L'Aquila city, taken from [18]

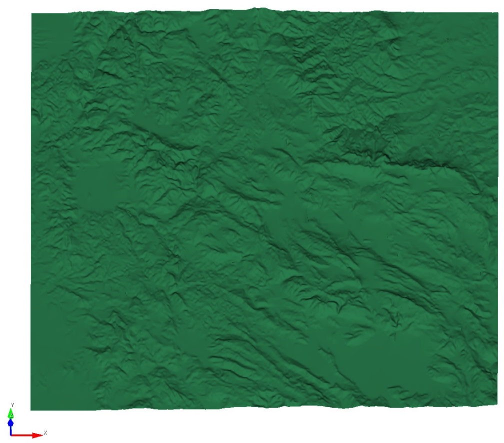

Figure 3: Real surface reconstructed from 62500 geo-located points surrounding L'Aquila city, taken from [18]

3.1.2 Layer Creation

This routine, contained in *Layer_Create.py*, is 'linked' to the first in that it needs a primary surface in order to function, however the primary surface need not be an Earth surface or special in any way. It is possible to skip the first routine and create or import a different primary surface before this step if desired, this can be controlled by the input inside the ***Main User Input file***. This routine will also allow the importing of an existing *.trelis* file containing a set of subsurface layers that the user wishes to use, again controlled by ***Main User Input file***.

For the typical use, that is creating a cake-layered model, the routine will work as follows. It will read the ***Main User Input file*** for the input file name related to the layer definitions and an output file name for the cake-layered model to be saved to. The routine will read the layer defining ***Layer info file*** in *CSV* format, see listing 4, for the requested layer depths. It then gives a series of commands to CUBIT in order to create planar surfaces at the necessary Z values. It is then possible to use these planar surfaces to cut the volume generated by the sweeping of the primary surface. After removing the small volume at the base of the set, which will contain a copy of the primary surface having been swept to create the volume, the remaining volumes will define a cake-layered domain. This domain contains the primary surface at its *top*, and a number of layers at depths depending on the user input at this step. An example is shown in figure 4.

3.1.3 Fault creation for earthquake dynamic models

Models able to simulate earthquake dynamics require the introduction of a source model. Seismic sources are modeled by defining a slip distribution along a plane known as a fault plane. Given the importance of the fault source to the model we have created a routine which automates the creation of the fault plane.

The technique that we use to introduce a planar fault source into a model is to perform a cut within the geometric entities, this will force the fault plane to occur at the interface between two or more volumes and therefore ensure that the fault plane is modelled exactly within the 3 dimensional mesh of the final domain. This method will lead to the introduction of the planar source without strong impact to the meshing process used to generate the hexahedral mesh required for the simulations. Complexities can arise depending on the planar source location, strike (rotation from North) etc, these complexities may lead to a hexahedral mesh with negative Jacobian. In section 3.3 we outline two techniques for planar source addition that lead to a usable mesh being created.

This routine within the tool, contained in *Fault_Create.py*, will read ***Main User Input file*** in order to access the name related to the ***Fault info file***. This file contains geo-paramaters related to the fault (such as the length, width etc.). This routine is not 'linked' to the previous two since it is

designed to start from a new instance. The routine requires the following information within the *CSV* file:
X-location, Y-location, minimum depth, length, width, dip, strike and a name which will be used as the filename upon saving, an example can be seen in listing 5. Upon completion of the routine each fault is saved in an individual file, which is then easily imported into a domain file for editing as needed. Since the fault plane is placed using the location information provided in the ***Fault Info file*** the fault plane is immediately generated in the correct place within the domain. Figure 4 shows an image of two real fault approximations imported into a layered domain. Information shown in listing 5 was taken from `http://diss.rm.ingv.it/dissGM/`, and was used to generate the faults shown in figure 4.

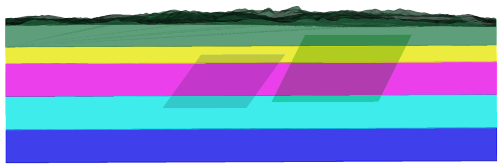

Figure 4: 5 layer domain with two fault planes located within it. Fault planes were generated from information available at `http://diss.rm.ingv.it/dissGM/`.

3.1.4 Additional source information - Hypo-center, Hypo-strike, Hypo-dip

In earthquake dynamic numerical simulations it may be useful to know the Hypo-center, Hypo-strike and Hypo-dip in the *domain system*. The Hypo-center is the point directly beneath the epicenter which defines the initial point of energy release for a given Earthquake, the Hypo-strike and Hypo-dip refer to the movement across the source plane in the horizontal and vertical directions respectively. Each is measured from the upper left corner of the plane to the Hypo-center. To calculate this *Hypo_Info.py* projects a given epicenter onto a previously generated fault plane, from which it is possible to find the Hypo-center, Hypo-strike and Hypo-dip in the *domain* system of reference in relation to the upper left corner of the fault.

Hypo_Info.py works by reading input from a *.txt* file, an example is shown in listing 6, which contains the fault name (name of existing *.trelis* file(s) containing a fault) and the epicenter used for each fault. The routine will then open this fault into a new instance and generate a vertex representing the epicenter. This vertex is then projected onto the fault surface and from here it is possible to create curves on the surface which represent the hypo-strike and hypo-dip. This is currently not linked through the *main* routine however, work is underway to connect it with the rest of the tool through *Fault_Create.py*.

3.2 Performance evaluation and examples

In numerical geo-dynamic applications and simulations one of the most labour intensive parts is the creation of the model domain and any required features, such as fault input planes. We now report some timings for two different surfaces and their respective 5 layered 3D domains. All were generated in batch mode on the Linux HPC cluster "Caliban" of the High Performance Parallel Computing Laboratory of the Department of Information Engineering, Computer Science and Mathematics (DISIM) at the University of L'Aquila. Tests were run on a compute node with the following specs: DELL R730 2CPUs Intel Xeon E5 2698 v4 2.2 GHz (40cores/80threads) 256 Gbyte RAM with Linux CentOS 6.5 , python 3.7 and CUBIT 16.5/17.0. The timing results are shown in table 1. Each domain and surface contained within a delineated 2 row section are linked, meaning that the 5-layer domain created starts from that given surface. Therefore the domain time noted includes the surface creation time.

To time the fault creation routine 100 faults were created to extend the timing to record-able lengths. Of the 100 faults created there are 4 distinct faults, each repeated 25 times, with the output files overwritten each time.

CUBIT Version	Geometric Object	Time
Pro 16.5	Surface (21750 points)	0:20
Pro 16.5	Domain (5 layers)	0:38
Pro 17.0	Surface (21750 points)	0:20
Pro 17.0	Domain (5 layers)	0:37
Pro 16.5	Surface (62500 points)	1:27
Pro 16.5	Domain (5 layers)	2:27
Pro 17.0	Surface (62500 points)	1:26
Pro 17.0	Domain (5 layers)	2:26
Pro 16.5	Fault (100 faults)	0:03
Pro 17.0	Fault (100 faults)	0:03

Table 1: Linux timings

The timings reported in table 1 are an average over 5 tests. The surface containing 21750 points is created with 150 curves of 145 points, the surface of 62500 points consists of 250 curves of 250 points. Both surfaces use an underlying dataset with approximately 400m resolution in X and Y, taken from [18]. Examples of these surfaces are shown in figures 2 and 3 respectively, with an example layered domain shown in figure 4.

Looking at table 1 it is obvious that the two versions of

CUBIT have a negligible difference in timing. The two surfaces and domains also overlap completely, again meaning negligible (if any) difference between the versions. The main result of this work, demonstrated in table 1, is that it is possible to create a 5 layered domain, from a real surface with 62500 points in around 2:30, including all IO operations. It must be noted that while this reduces simple but potentially labour intensive work to minutes it also allows for parallel running of the scripts on any available multi-core machine. This could allow the creation of any number of distinct domains with different features located in different files, limited only by the number of cores present on any machine, inside 3 minutes.

3.3 Mesh Generation

In geological applications it is important that the mesh is able to represent the underlying geometry. This issue is particularly apparent in any simulations that are effected by the resolution of a real surface. CUBIT is mainly used for engineering applications with regular geometries, however the irregular surfaces and domains obtained using the tool described in this paper are, in general, meshable with in-built meshing tools. These meshed domains are usable for geo-computing applications without mesh deformation and irregularities that could lead to errors during simulations. CUBIT is able to generate a representative mesh for the real surface without difficulty, an example is shown in figure 5, the mesh shown contains a large mesh size (approximately 1500m) for ease of visibility within the figure.

However, any highly irregular cuts made are likely to generate a negative Jacobian when using a hexahedral mesh. Therefore the addition of a planar fault source requires the cutting of the domain in a way that will allow the final hexahedral mesh to be generated without any features that could prevent usability, (i.e. negative Jacobian).

We have worked on domain reconstruction techniques to overcome the mesh problems listed. The required cuts for a planar fault source depend on the position of the source relative to the domain boundaries, the orientation of the source, its size, its locality relative to any highly irregular surface features and its distance from the real surface. In the simplest case it is possible to perform a simple series of cuts originating from the source plane in order to divide the domain into two pieces, an example is shown in figure 6. In this example we see three distinct interfaces which combine to divide the domain. The central interface exists on the plane of the source and extends it to the surface, the interfaces propagating from this central one are defined by the two outer edges of the source plane. Figure 7 shows the mesh generated at the edge of the domain after a cut is made to include a fault source. In more complicated situations it is necessary to perform numerous cuts and subdivide the domain into smaller volumes. This can be required to generate a working mesh by simplifying the shapes within the domain that need to be meshed. An example of this is shown in figure 8. Here the fault is bound within a box defined by its 4 edges and this box is used to cut towards the domain boundaries similar to the first example. The issue of meshing after performing cuts to add planar sources exists due to our need for a hexahedral mesh, the use of a tetrahedral mesh would greatly simplify this as it would allow far more irregular shapes within the domain to be meshed, however the simulation software which we use in our work requires a hexahedral mesh.

In built tools also allow mesh refinement, this is important within geo-dynamic models due to the size of the domains and the requirement for small element sizes in order to correctly treat more interesting frequencies, therefore the meshes are typically created with larger sized elements and then refined around an area of interest. In figure 9 it is possible to see a refinement to the surface as shown in figure 5. It is clear when comparing figures 5 and 9 that with each refinement the mesh is able to more accurately represent the real surface. The new offering of parallel meshing from CUBIT offers the user the option of creating a usable mesh in a fraction of the time. The meshing is split over separate entities by the software and each meshed in parallel over a number of nodes selected by the user. This can offer a significant speed up with the 'base mesh'. However it does not seem to accommodate refinement, it is a new option and still has several issues for users with highly irregular surfaces - as we have in our geo-dynamic models.

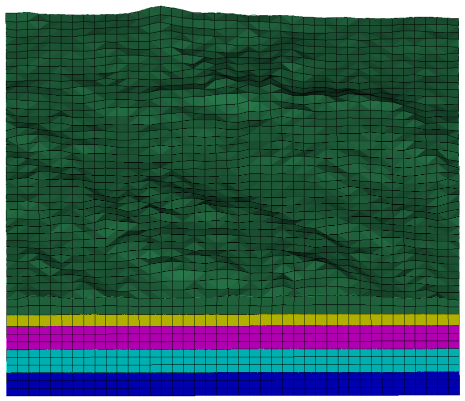

Figure 5: This figure shows an example 1500m element size hexahedral mesh.

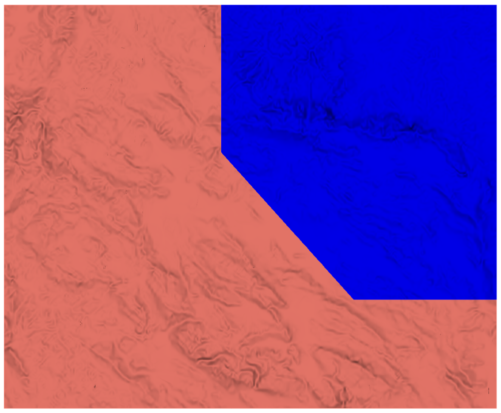

Figure 6: Example of the simplest cut required to introduce a planar fault source. The central of the three curves defining the boundary between the two surfaces extends directly from the planar fault which itself does not touch the surface.

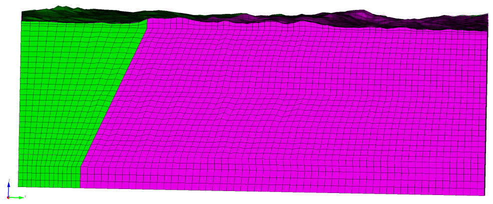

Figure 7: Mesh section surrounding a cut required for the introduction of a planar fault. Green volumes are on the left and pink on the right of a cut made through the domain to accommodate a planar fault source.

4. CONCLUSION

In general creating a domain manually with CUBIT is not a complicated task but it can be a time consuming and labour intensive one, particularly for geological models and applications. Although the tool introduced here does not necessarily create the absolute final version of a domain - depending on the needs of the user - it will generate one which is almost complete, as extra steps may be required depending on the simulations. For this reason the tool does not currently include a meshing step. What's more, this process has been streamlined into a single collection of scripts controlled by the user which allows the creation of a usable domain in minutes. The scripts also allow the user to pick and choose which parts of the model to build - just surfaces, faults or entire 3D domains.

The tool may be used in parallel through the use of multiple instances on Linux, Windows, or MAC machines, in order to accelerate the creation of multiple

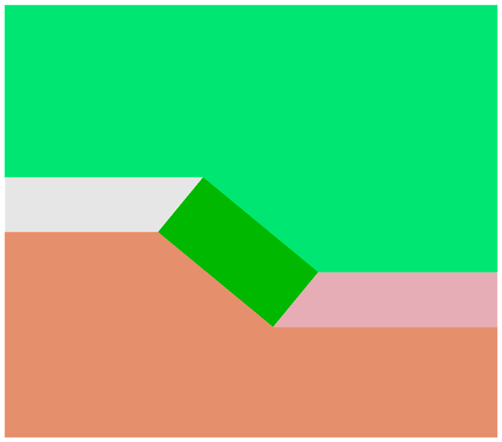

Figure 8: Example of the *box* cut to introduce a planar fault source. The central dark green surface is the top of a cuboid which contains the planar fault. The surfaces connected to its smaller edges are cuboids created by cutting from the fault box to the domain boundaries.

domains with differing features. This option could be used to generate multiple models with minor changes by altering the IO filenames inside the ***Main User Input file*** in order to point to files with, for example, a different layer structure or a different surface resolution.

The tool offers a further advantage, it is accurate to the data-set given and to CUBIT interpolations. Should the user wish to change the geo-located points used in the surface creation then the script will create a more or less accurate surface depending on the new input. The only limit for the tool is the input data set and the ability of CUBIT to interpolate a spline from vertices and a surface from a set of splines. In the future, should CUBIT have an updated interpolation method, the surface creation script will benefit immediately with no cost to the user.

Ongoing and future work includes the extension of the existing tool with the target of creating output that is usable in other simulation software, such as SPECFEM3D and PyLith `https://geodynamics.org/cig/software/pylith`, by incorporating, for example, the need for the input parameters to be attached to each layer block.

5. ACKNOWLEDGEMENTS

This work was partially supported by the GSSI "Centre for Urban Informatics and Modelling" (CUIM) (`https://www.opendatalaquila.it`) and the GNCS-INdAM. All numerical simulations have been realized on the Linux HPC cluster Caliban of the High Performance Parallel Computing Laboratory of the De-

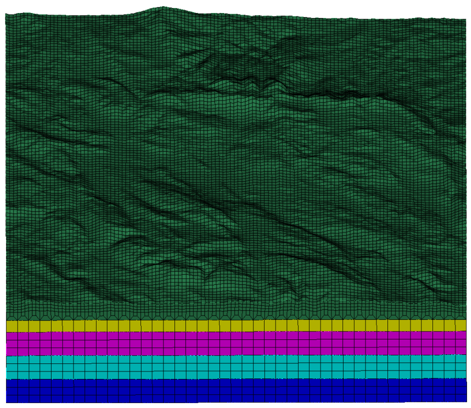

Figure 9: A surface refinement of the mesh shown in figure 5

partment of Information Engineering, Computer Science and Mathematics (DISIM) at the University of L'Aquila (`https://caliband.disim.univaq.it`). We thank the DISIM for the technical support.

References

[1] Datta S., El-Akily N. "Diffraction of elastic waves by cylindrical cavity in a half-space." *The Journal of the Acoustical Society of America*, vol. 64, no. 6, 1692–1699, 1978

[2] Lee V.W., Trifunac M.D. "Response of tunnels to incident SH-waves." *Journal of the Engineering Mechanics Division*, vol. 105, no. 4, 643–659, 1979

[3] Dravinski M. "Scattering of SH waves by subsurface topography." *Journal of the Engineering Mechanics Division*, vol. 108, no. 1, 1–17, 1982

[4] Smerzini C., Aviles J., Paolucci R., Sánchez-Sesma F. "Effect of underground cavities on surface earthquake ground motion under SH wave propagation." *Earthquake Engineering & Structural Dynamics*, vol. 38, no. 12, 1441–1460, 2009

[5] Bouchon M., Schultz C.A., Toksöz M.N. "Effect of three-dimensional topography on seismic motion." *Journal of Geophysical Research: Solid Earth*, vol. 101, no. B3, 5835–5846, 1996

[6] Bouchon M., Barker J.S. "Seismic response of a hill: the example of Tarzana, California." *Bulletin of the Seismological Society of America*, vol. 86, no. 1A, 66–72, 1996

[7] Durand S., Gaffet S., Virieux J. "Seismic diffracted waves from topography using 3-D discrete wavenumber-boundary integral equation simulation." *Geophysics*, vol. 64, no. 2, 572–578, 1999

[8] Lee S.J., Komatitsch D., Huang B.S., Tromp J. "Effects of topography on seismic-wave propagation: An example from northern Taiwan." *Bulletin of the Seismological Society of America*, vol. 99, no. 1, 314–325, 2009

[9] Shearer P. *Introduction to seismology.* Cambridge university press, 2019

[10] Aki K., Richards P.G. *Quantitative seismology.* Univ Science Books, 2002

[11] Igel H. *Computational Seismology A Practical Introduction.* 2016, Oxford University Press

[12] Antonietti P.F., Ferroni A., Mazzieri I., Paolucci R., Quarteroni A., Smerzini C., Stupazzini M. "Numerical modeling of seismic waves by discontinuous spectral element methods." *ESAIM: Proceedings and Surveys*, vol. 61, 1–37, 2018

[13] Patera A.T. "A spectral element method for fluid dynamics: laminar flow in a channel expansion." *Journal of computational Physics*, vol. 54, no. 3, 468–488, 1984

[14] Maday Y., Patera A.T. "Spectral element methods for the incompressible Navier-Stokes equations." *IN: State-of-the-art surveys on computational mechanics (A90-47176 21-64). New York, American Society of Mechanical Engineers, 1989, p. 71-143. Research supported by DARPA.*, pp. 71–143. 1989

[15] Priolo E., Carcione J.M., Seriani G. "Numerical simulation of interface waves by high-order spectral modeling techniques." *The Journal of the Acoustical Society of America*, vol. 95, no. 2, 681–693, 1994

[16] Seriani G., Priolo E. "Spectral element method for acoustic wave simulation in heterogeneous media." *Finite elements in analysis and design*, vol. 16, no. 3-4, 337–348, 1994

[17] Michele F.D., May J., Pera D., Kastelic V., Carafa M., Smerzini C., Mazzieri I., Rubino B., Antonietti P.F., Quarteroni A., Aloisio R., Marcati P. "Spectral elements numerical simulation of the 2009 L'Aquila earthquake on a detailed reconstructed domain." *Submitted*

[18] Tarquini S., Isola I., Favalli M., Mazzarini F., Bisson M., Pareschi M.T., Boschi E. "TINITALY/01: a new triangular irregular network of Italy." *Annals of Geophysics*, 2007

A. APPENDIX USER INPUT EXAMPLE

Listing 1: Example user input file script control

```
# —— User selection
Create_Surface=True
Create_Layers=True
Create_Faults=False
# —— Basic IO File names
Range_in=Range.csv
Surf_in=Surface.csv
Layer_in=Layer.csv
Fault_in=Fault_real.csv
Surf_out=IMPPIO_Surf.trelis
Domain_out=IMPPIO_Dom.trelis
# —— Import options & file names
Import_Surface?=False
Import_Layer?=False
Surf_import=
Layer_import=
Number_Layer=
Depth=
```

Listing 2: Partial example of user input file for surface

```
800150.4,4650150.4,37.174
800150.4,4650551.2,26.234
800150.4,4650952.0,26.095
800150.4,4651352.8,33.032
```

Listing 3: Example user input file for surface range

```
840000,899850
4670000,4728000
```

Listing 4: Example user input file for layers

```
5
2000
4000
8000
12000
16000
```

Listing 5: Example user input file for fault planes

```
863969.9,4715921.3,−12000,10000,6000,75,95,Isola
866721.1,4694904.7,−3000,14000,9500,43,133,Paganica
850503.0,4705208.2,−3000,23400,13600,50,132,Montereale
881035.4,4690099.6,−700,16200,10500,50,127,Pio
```

Listing 6: Example user input file for Hypo-centre information

```
Fault=Montereale,Paganica
Epicentre=853138.07,4707202.97,866721.1,4694904.7
```