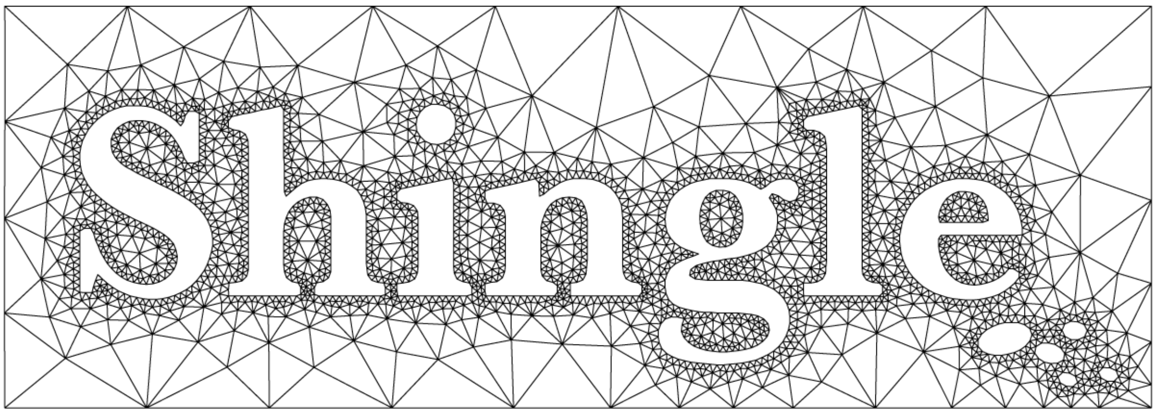
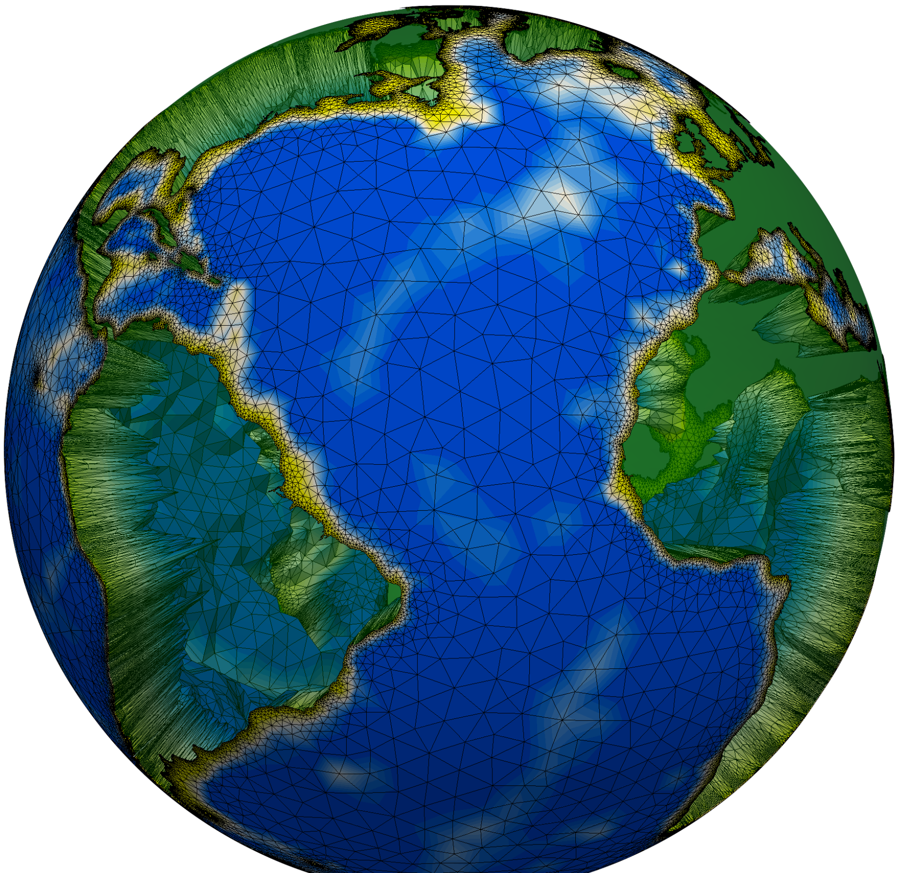


Shingle: Self-consistent and automated domain discretisation for multi-scale geophysical models

Manual, version 2.0.0



Adam S. Candy, Julie D. Pietrzak and Shingle project contributors

Monday, 14th November 2016

Details of the project presented at: https://www.shingleproject.org.
Library code, verification tests and examples available in the project repository at:
https://github.com/shingleproject/Shingle.

Project contact: Dr Adam S. Candy
a.s.candy@tudelft.nl

Environmental Fluid Mechanics Section,
Faculty of Civil Engineering and Geosciences,
Delft University of Technology,
Delft, The Netherlands.

Contents

Chapter 1

Introduction

1.1 The challenge

The challenge (see figure 1.1) of constraining and fully describing an arbitrarily unstructured spatial discretisation[1] bounded by complex, fractal-like bounds that typically characterise geophysical domains, with inhomogeneous and potentially anisotropic spatial resolution, is a significant one. Defining the domain geoid bounds is no longer a simple case of applying a land mask to similarly regular gridded data. The generalised constraints are now a heterogeneous set of functions [Candy, 2017], and as a consequence are more difficult to describe.

The approaches taken to describe and develop spatial discretisations of the domains required for geophysical simulation models are commonly ad hoc, model or application specific and under-documented. This is particularly acute for simulation models that are flexible in their use of multi-scale, anisotropic, fully unstructured meshes where a relatively large number of heterogeneous parameters are required to constrain their full description. As a consequence, it can be difficult to reproduce simulations, ensure a provenance in model data handling and initialisation, and a challenge to conduct model intercomparisons rigorously.

Moreover, these challenges present a significant hurdle for new users to unstructured mesh models, and tend to limit their use to the specialist community.

1.2 The solution and project purpose

This project takes a novel approach to spatial discretisation, considering it much like a numerical simulation model problem of its own. It has been approached to both make this part of unstructured mesh numerical simulation accessible to a wide range of users (including those new to flexible mesh models), whilst providing a formalised platform for existing users. It introduces a generalised, extensible, self-documenting approach to carefully describe, and necessarily fully, the constraints over the heterogeneous parameter space that determine how a domain is spatially discretised. This additionally provides a method to accurately record these constraints, using high-level natural language based abstractions, that enables full accounts of provenance, sharing and distribution. Together with this description, a generalised consistent approach to unstructured mesh generation for geophysical models is developed, that is automated, robust and repeatable, quick-to-draft, rigorously verified and consistent to the source data throughout. This interprets the description above to execute a self-consistent spatial discretisation process, which is automatically validated to expected discrete characteristics and metrics.

1.3 Objectives

1. Introduce a consistent approach to the generation of boundary representation to arbitrary geoid bounds.

[1]For the purposes of the discussion here, *spatial discretisation* specifically refers to the division of a continuous spatial domain into discrete parts – a discrete tessellation or honeycomb – a generalised notion of triangulation.

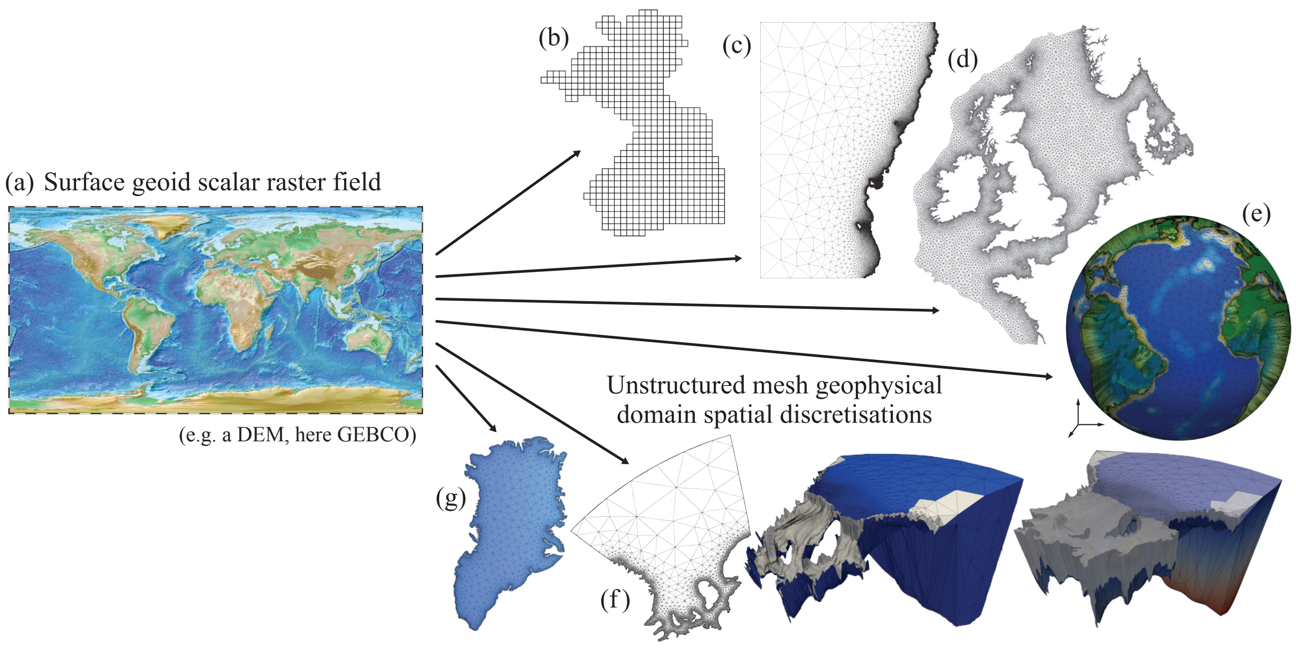

(a) Surface geoid scalar raster field
(e.g. a DEM, here GEBCO)
(b)
(c)
(d)
(e)
Unstructured mesh geophysical domain spatial discretisations
(g)
(f)

Figure 1.1: The challenge: to generate a self-consistent domain discretisation approach for geophysical domains that is generalised such that it can be applied to a wide range of applications, with new domains efficiently prototyped and iterated on, and is fully described such that the process can be automated, is reproducible and easily shared. (a) shows a typical source Digital Elevation Map (DEM) dataset (that naturally lend themselves to structured grid generation) used to produce a regular grid of the Atlantic Ocean (e.g. under a format-native land mask) in (b), and a selection of unstructured mesh spatial discretisations: (c) Bounded by part of the Chilean coastline and a meridian. (d) North Sea. (e) Global oceans. (f) Grounding line of the Filchner-Ronne ice shelf ocean cavity up to the 65°S parallel, with surface geoid mesh $\mathcal{T}_h$, full mesh $\mathcal{T}$ with ice-ocean melt interface highlighted, and accompanied by ice sheet full discretisation. (g) Greenland ice sheet.

2. A user-friendly, accessible and extensible framework for model-independent geophysical domain mesh generation.
3. An intuitive, hierarchical formal grammar to fully describe and share the full heterogeneous set of constraints for the spatial discretisation of geophysical model domains.
4. Natural language basis for describing geophysical domain features.
5. Self-consistent, scalable, automated and efficient mesh prototyping.
6. Platform for iterative development that is repeatable, reproducible with a provenance history of generation.
7. Enabling rigorous unstructured mesh generation in general, for a wide range of geophysical applications, in a process that is automated, quick-to-draft and repeat, rigorous and robust, and consistent to the source data throughout.

1.4 Citation

Please cite this work with reference to the following publication:

> Candy, A.S and Pietrzak, J.D., 2017. Shingle 2.0: generalising self-consistent and automated domain discretisation for multi-scale geophysical models, *Submitted to Geoscientific Model Development* [Candy and Pietrzak, 2017].

This builds on the consistent approach to mesh generation for the boundary-conforming, variable resolution domains for geophysical models developed in:

> Candy, A.S., 2016. A consistent approach to unstructured mesh generation for geophysical models. *In review*, [Candy, 2017].

The following, where Shingle has found use, may also be useful resources:

> Candy, A.S., Avdis, A., Hill, J., Gorman, G.J., Piggott, M.D., 2014. Integration of Geographic Information System frameworks into domain discretisation and meshing processes for geophysical models. *Geoscientific Model Development Discussions* 7, 5993–6060, DOI:10.5194/gmdd-7-5993-2014 [Candy et al., 2014].

Chapter 2

Installation and building

2.1 Quick start

Install Shingle, Diamond and all Python dependencies with the following:

```
pip install --user numpy ScientificPython matplotlib shapely
  GDAL Pydap pyproj Pillow setuptools spud-diamond shingle
```

Note the `--user` option can be dropped to install system-wide, if you have sufficient system privileges.
Download the source code, which includes the verification test suite:

```
git clone https://github.com/shingleproject/Shingle
```

Move to the source folder:

```
cd Shingle
```

and perform the verification tests:

```
make test
```

Examine the output of a test case:

```
gmsh test/Antarctica_all/Antarctica_all.geo
  test/Antarctica_all/Antarctica_all.msh
```

Examine test case description BRML with a text editor, e.g. the file:

```
test/Antarctica_all/Antarctica_all.brml
```

Examine the test case description BRML using Diamond:

```
diamond -s schema/shingle_options.rng test/Antarctica_all/Antarctica_all.brml
```

Individual BRML descriptions can be interpreted and processed with:

```
shingle Caribbean_basin.brml
```

2.2 Dependencies

Shingle is written in Python (`http://python.org/`) and has been routinely used with Python 2.7. Its dependencies are summarised in figure 5.1 and more precisely specified in the standard `requirements.txt` file in the root source folder. These include:

- numpy (`http://www.numpy.org`)
- ScientificPython (`http://dirac.cnrs-orleans.fr/ScientificPython`)
- matplotlib (`http://matplotlib.org`)

– shapely (http://toblerity.org/shapely)
– GDAL (http://www.gdal.org)
– Pydap (http://pydap.org)
– pyproj (http://proj4.org)
– Python Imaging Library (PIL) provided by Pillow (https://python-pillow.org)
– setuptools (http://peak.telecommunity.com/DevCenter/setuptools)

These are all available on Python Package Index, PyPI (https://pypi.python.org) and this is the recommended way to source these dependencies. The following can be used to install these using `pip`:

```
pip install numpy ScientificPython matplotlib shapely GDAL
  Pydap pyproj Pillow setuptools
```

To avoid issues with file permissions, the easiest way is to install these on a user basis with the following:

```
pip install --user numpy ScientificPython matplotlib shapely GDAL
  Pydap pyproj Pillow setuptools
```

2.3 Obtaining the source

The source for Shingle is available from https://github.com/shingleproject/Shingle with further details available at https://www.shingleproject.org.
The recommended way to install Shingle is with `pip`, again from PyPI, using:

```
pip install shingle
```

Again, use the `--user` option to install locally, on a user basis:

```
pip install --user shingle
```

2.4 Building from source

Shingle can be built from source. It should be possible to build using the standard:

```
./configure
make
```

2.5 Diamond

2.5.1 Installing Diamond

Diamond is provided by the package Spud. The easiest method to install Diamond is using `pip`, again from PyPI, using:

```
pip install spud-diamond
```

Further documentation on Diamond is available in the Spud manual, which is provided alongside the Shingle source in the spud package folder: `spud/doc/spud_manual.pdf`.

2.5.2 Set up of Diamond

To examine with Diamond, first let Diamond know when to find the schema containing the formal grammar:

```
mkdir -p ~/.diamond/schemata
cp doc/schemata/brml ~/.diamond/schemata/
```

Where the `brml` file above is contained in `doc/schemata/` from the root source folder. This file contains paths to the schema

```
diamond test/Antarctica_all/Antarctica_all.brml
```

2.5.3 Building Diamond from source

Diamond is provided by the package Spud. Further details of building Diamond from source can be found with the Spud source code, which is provided alongside the main Shingle source in the `spud` folder.

2.6 Building of the manual

The manual is built using LaTeX (http://www.latex-project.org/), and currently requires the following packages:

- gensymb
- etoolbox
- enumitem
- url
- natbib
- cleveref
- ntheorem
- minted
- ifplatform
- xstring
- lineno

Where possible, the manual automatically includes figures of the output spatial discretisations from the verification tests. This is achieved in Python which links directly to the Shingle library to read all of the test BRML files in the test folders to locate the output mesh images.
The mesh images for all active tests can be generated by:

```
make testimage
```

in the route source folder, or using the Shingle executable:

```
shingle -t path_to_test_folder -image
```

The manual is built using the following Makefile directive from the root source folder:

```
make manual
```

Chapter 3

Command line operation

3.1 Straight-forward direct use on the command line

Shingle can be used straight-forwardly on the command line to interpret and process BRML spatial discretisation problem descriptions. This is simply a matter of passing Shingle the filename of the BRML description. For example:

```
shingle test/Amundsen_Sea/Amundsen_Sea.brml
```

This reports back each stage, following through the set of constraints (e.g. figure 9.1). For example:

```
DATASETS: Found 1 datasets:
  1. RTopo
      Path:       /dataset::RTopo
      Form:       Raster
      Source:     Local_file
      Location:   ../../dataset/RTopo105b_50S.nc
      Projection: Automatic
SURFACE GEOID REPRESENTATIONS: Found 1 surface geoid representations:
  1. Amundsen_Sea
      path:      /geoid_surface_representation::filchner_ronne_ice_ocean::Amundsen_Sea
Logging to file: /home/acandy/src/Shingle/test/Amundsen_Sea/shingle.log
  Initialising surface geoid representation Amundsen_Sea
  Output to Amundsen_Sea.geo
  Projection type cartesian
Extending region to meet parallel on latitude -64.0
BOUNDARY IDENTIFICATION: Found 2 boundary definitions:
  1. Coast
      Path:        /geoid_surface_representation::Amundsen_Sea/boundary::Coast
      Physical ID: 3
  2. OpenOcean
      Path:        /geoid_surface_representation::Amundsen_Sea/boundary::OpenOcean
      Physical ID: 4
COMPONENT BOUNDARY REPRESENTATIONS: Found 2 component boundary representations:
  1. Amundsen_Sea_brep
      Path:           /geoid_surface_representation::Amundsen_Sea/brep_component::Amundsen_Sea_brep
      Form:           Raster
      Identification: Coast
  2. ExtendTo64S
      Path:           /geoid_surface_representation::Amundsen_Sea/brep_component::ExtendTo64S
      Form:           ExtendToParallel
      Identification: OpenOcean
  Reading boundary representation Amundsen_Sea_brep
  Region defined by ((longitude >= -130.0) and (longitude <= -85.0)
      and (latitude >= -85.0) and (latitude <= -60.0))
  Open contours closed with a line formed by points spaced 0.1 degrees apart
  Source NetCDF located at ../../dataset/RTopo105b_50S.nc
  Generating contours, from raster: /home/acandy/tmp/dataset/rtopo/RTopo105b_50S.nc
    Including iceshelf ocean cavities
    Found raster, sizes: lat 2401, lon 21601, shape (2401, 21601)
  Paths found: 348
  Merged paths that cross the date line:
    Path 1: points 5165 (of 59599) area 3.84218e+06 (required closing in 2 parts of the path)
    Path 23: points 158 (of 159) area 2191.14 (required closing in 2 parts of the path)
    Path 27: points 512 (of 513) area 8103.48 (required closing in 2 parts of the path)
    Path 28: points 18 (of 19) area 5.40939 (required closing in 2 parts of the path)
    Path 29: points 680 (of 681) area 58481.4 (required closing in 2 parts of the path)
    Path 30: points 180 (of 181) area 1374.98 (required closing in 2 parts of the path)
    Path 31: points 172 (of 173) area 1082.44 (required closing in 2 parts of the path)
    Path 36: points 108 (of 109) area 538.654 (required closing in 2 parts of the path)
    Path 37: points 212 (of 213) area 1205.91 (required closing in 2 parts of the path)
    Path 39: points 130 (of 131) area 913.168 (required closing in 2 parts of the path)
    Path 40: points 60 (of 61) area 225.112 (required closing in 2 parts of the path)
    Path 41: points 240 (of 241) area 2238.38 (required closing in 2 parts of the path)
    Path 44: points 104 (of 105) area 800.669 (required closing in 2 parts of the path)
    Path 45: points 1274 (of 1275) area 25276.9 (required closing in 2 parts of the path)
    Path 82: points 60 (of 61) area 65.104 (required closing in 2 parts of the path)
  Paths found valid (renumbered order): 15, including 1 5 10 13 31 37 40 44 47 51 56 62 91 93 159
  Processing paths:
    Interior path 5
    Interior path 10
    Interior path 13
    Interior path 31
    Interior path 37
    Interior path 40
    Interior path 44
    Interior path 47
    Interior path 51
    Interior path 56
    Interior path 62
    Interior path 91
    Interior path 93
    Interior path 159
    Exterior path 1
  Extending exterior boundary developed in Amundsen_Sea_brep to parallel -64.0
  Boundary OpenOcean (ID 4): 16
  Boundary Coast (ID 3): 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Writing surface geoid representation to file: Amundsen_Sea.geo
    /home/acandy/src/Shingle/test/Amundsen_Sea/Amundsen_Sea.geo
Mesh properties: /home/acandy/src/Shingle/test/Amundsen_Sea/Amundsen_Sea.msh
  Number of nodes:    2630
  Number of elements: 5282
  Reading validation tests (1 in total)
    Test 1: BrepDescription  PASS
    Test 2: NodeNumber  PASS
    Test 3: ElementNumber  PASS
    Test 4: SurfaceGeoidArea  PASS
  Result: Amundsen_Sea  PASS  (4)
```

3.2 Diagnostics

There are various ways to check what the BRML describes. A plot of contours identified can be shown quickly with the `plot` option:

```
shingle test/Amundsen_Sea/Amundsen_Sea.brml -plot
```

3.3 Diamond GUI editor

The BRML descriptions are structured, human-readable files and can be read simply in a text editor.

To ensure the descriptions are structured following the BRML grammar, they can be edited in the Diamond GUI. It is simply a matter of passing Diamond the schema file describing the grammar, together with the BRML file. For example:

```
diamond -s schema/shingle_options.rng test/Antarctica_all/Antarctica_all.brml
```

3.4 Command line options

Command line options for Shingle can be list by passing the help argument `-h`:

```
Usage for shingle
 shingle [options] (filename)
-- Options ----------\
   filename          | Option tree file fully specifying the
                     |   surface geoid representation and spatial discretisation
   -t (testfolder)   | Run all tests located in testfolder (defaults to the source test folder)
   -l (logfile)      | Log messages to file (optional filename, default shingle.log in project folder)
   -c                | Use a cache file, create and read
                     |_____________________________________________________________________________
   -stage stagename  | Partial processing, up to given stage from: path brep mesh metric verify post
   -pickup           | Use existing generated output if exists and where possible
   -plot             | Plot contour before representation generation
   -image            | Generate image of mesh (forced)
                     |   combine with Xvfb :5 -screen 0 2560x1440x8 to run in the background
   -mesh             | Mesh geometry (forced)
   -update           | Update verification test files
                     |_____________________________________________________________________________
   -h                | Help
   -v                | Verbose
   -vv               | Very verbose (debugging)
   -q                | Quiet
                     \_____________________________________________________________________________
```

3.5 Command line constraint specification

It is possible to develop simple spatial discretisation constraint descriptions on the command line. This is for compatibility with earlier versions of the library. More information is available by providing the arguments `-legacy -h` on the command line.

Chapter 4

Background motivation

Numerical simulation models have become a vital tool for scientists studying geophysical processes. Mature operational models inform continuously updated short-term public weather forecasts, whilst studies of mantle dynamics and ice sheet evolution improve understanding of physical systems in relatively inaccessible locations, where data is sparse.

Use of unstructured mesh spatial discretisations is growing in the fields of modelling geophysical systems, where it is possible to conform accurately to complex, fractal-like surfaces and vary spatial resolution to optimally capture the physical process, or multi-scale range of processes under study. The past few years have seen a global unstructured ocean model [FESOM, Sidorenko et al., 2014] join structured studies in internationally coordinated climate studies, such the Coupled Model Intercomparison Project [CMIP, Meehl et al., 2007; Taylor et al., 2012] and the Coordinated Ocean-ice Reference Experiments [CORE Griffies et al., 2014, and accompanying studies in the *Ocean Modelling* special issue]. More are in active development [e.g. Ringler et al., 2013] and the number of unstructured models joining these efforts – that directly contribute to reports compiled by the Intergovernmental Panel on Climate Change (IPCC) – likely to grow. Similarly, on smaller scales, the geometric flexibility of unstructured discretisations are being applied to reduce the need for nesting models, and in accurately applying forcings or coupling physics [e.g. Kimura et al., 2013] on complex and possibly dynamic, deformable physical interfaces. At the cusp where these efforts meet, prospects for introducing successively greater complexity in the representation of coastal seas in global ocean models are reviewed in Holt et al. [2017].

The challenge (see figure 1.1) of constraining and fully describing an arbitrarily unstructured spatial discretisation bounded by complex, fractal-like bounds that typically characterise geophysical domains, with inhomogeneous and potentially anisotropic spatial resolution, is a significant one. Defining the domain geoid bounds is no longer a simple case of applying a land mask to similarly regular gridded data. The generalised constraints are now a heterogeneous set of functions [Candy, 2017], and as a consequence are more difficult to describe. In general, domain discretisations are often under-described leaving it difficult to repeat simulations exactly, which particularly for the unstructured case, can have a strong influence on model output. Not only is the description and generation process a significant challenge, but achieving this in a way that maintains a record of provenance such that simulations as a whole are reproducible, that scales and is efficient, and consistent to source data – attributes required and expected in scientific modelling studies – make this a much more difficult problem (summarised in table 7.1). Existing, standard structured-mesh tools cannot be used.

Grid generation for geophysical models in real domains is not only becoming a significantly more complex and challenging problem to constrain and describe, but additionally in the computational processing required. As models include a greater range of spatial scales, more computational effort is required to optimise the discretisation before a simulation proceeds (e.g. the actively developed MPAS models, Ringler et al. [2013], strongly optimise their hexagonal prism based mesh discretisation). An increasing number of geometric degrees of freedom demand the meshing process is broken up over multiple parallel threads [as demonstrated in Candy, 2017], just as simulation models have evolved to run in parallel.

These challenges are identified in Candy [2017] by the *nine tenets of geophysical mesh generation*, summarised in table 7.1. This work takes the view that significant progress can be made towards these by approaching the mesh generation problem in the same way as a numerical simulation model.

Chapter 5

Overview of Shingle library and approach

Simulation domains in geophysical models are typically defined with reference to geographical features. A tsunami simulation geoid surface domain is, for example, usually described by a length of coastline between two points (commonly marked by longitude or latitude references) extended out to an orthodrome. In the case of 2010 Chile earthquake centred about 35.9°S 72.7°W (see figure 11.2), the domain is concisely described:

> *"... bounded by the 0m depth coastline from 32 °S to 40 °S,*
> *extended along parallels to the 77 °W meridian,* (∗)
> *in a latitude-longitude WGS84 projection ..."*

As part of the generalisation of domain description, this new approach interacts directly with these natural language based geographic references, structured by a formal grammar, to provide a general, model-independent and accurate description of spatial discretisation for geophysical model domains. This forms part of the Shingle [2011–2017] computational research software library, that accompanies this work, providing a novel approach to describing and generating highly multi-scale boundary-conforming domain discretisations, for seamless concurrent simulation.

The previous work Candy [2017] developed a consistent approach to domain discretisation, with a focus on uniform processing and data sources, which further enabled the discretisation of domains not possible with standard approaches. Additionally, it identified the complete set of heterogeneous constraints required to fully describe a mesh generation problem for the discretisation of geophysical domains. This work now extends and generalises this consistent approach introducing a natural language based formal grammar for a modeller to describe and share the constraints. Under the formal grammar the description is ensured necessarily complete, such that the problem is fully constrained and is therefore reproducible. This employs the novel hierarchical problem descriptor framework Spud [Ham et al., 2009] which has been specifically designed to manage large and diverse option trees for numerical models. The formal self-describing data file is a universal, shareable description of the full constraints, written in a standard data format, presented in context through a natural hierarchical structure, readable by established open source libraries.

The pathways of interaction with the library have grown (outlined in figure 5.1), such that it is accessible to a wide range of users. Its modular library framework, with for example, geospatial operations, homeomorphic projections, meshing algorithms and model format writers are the focus of distinct modular parts, and the use of standard external libraries where possible, allows development to remain in small sections of the code base, such that develops can stay within their specialisms. Additionally, the dictionary approach to managing option parameters taken by Spud means new features can be added and exposed through interfaces, such as the Diamond Graphical User Interface (GUI), without the need to pass new arguments through code functions, and similarly require small changes and only in low-level code.

Output writers in the library prepare the solution discretisation for use in simulation codes, in cases where the output Python objects cannot be used directly, encouraging the use of standard formats and also supporting existing proprietary model-specific formats. These additionally support supplementing the spatial discretisation (which itself includes a vector field describing mesh node coordinate locations) with additional interpolated fields for simulation model initialisation and forcing (figure 5.1).

Through both the objects in the problem description file (figures 9.1 – 9.3) and those in the Python library LibShingle (figure 11.1), Shingle provides a language to combine geographic components to build up boundary representation, mesh spatial variation and identification – a high-level abstraction to the complex constraint description problem – which is then processed by the library in deterministic (or as close to as possible) process to accurately construct the specified mesh in a repeatable way.

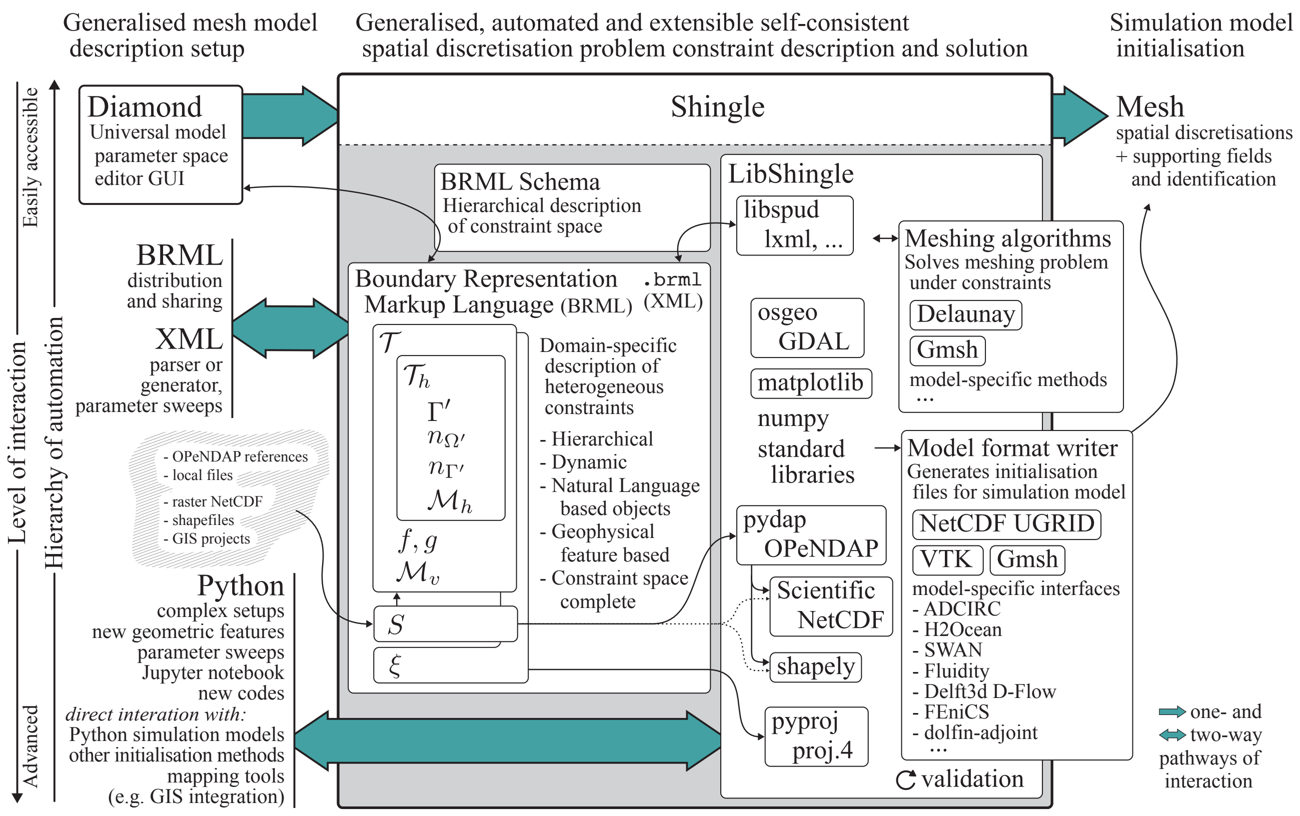


Figure 5.1: A schematic illustrating the generalised approach to flexible unstructured mesh specification and generation for geophysical models. The hierarchy of automation (tenet 7) is highlighted, from a relatively simple high-level interaction: Diamond GUI $\leftrightarrow$ Shingle $\rightarrow$ Mesh, to complex low-level development communicating with the LibShingle library. Nomenclature defined in chapter 6.

The validation tests of Candy [2017] have been significantly widened from the limited boundary representation tests to include expected discrete properties and metrics of the high fidelity description and resulting domain discretisation. These expected characteristics are prescribed as part of the self-describing problem file, such that other users can check the output is as intended. This self-contained description and validation is then straight-forwardly processed by the library verification engine, making it easy to add new tests.

Through this approach, geophysical domain discretisation can be the relatively simple steps (top of figure 5.1) of using the Diamond GUI to choose a dataset and specify bounds using natural language objects, which is then run through the Shingle executable to produce a mesh. This is accessible and straightforward to new users. More so with the suite of test cases that provide examples and easily ensure verification through a built-in test engine.

More advanced use can be built up in stages through the GUI, with validation checks on expected mesh properties easily added to ensure reliable reproduction throughout the iterative mesh prototyping process. Beyond this the XML based description is easily interrogated and modified with standard tools. Lower-lever still, the natural language based objects and discretisation constraints can be accessed directly through its Python library interface. This has grown since its first iteration reported in Candy [2017], where it was used to develop complex discretisations dependent on the mean position of Antarctic Circumpolar Current (ACC) and domains to complex grounding line positions under the floating ice shelves of Antarctica. Python plugins for QGIS [Quantum GIS Development Team, 2016] were developed using parts of the Shingle library code to demonstrate integration with Geographic Information Systems (GIS) in Candy et al. [2014].

With mesh generation becoming a complex problem to describe and a computationally challenging process, that we argue is best handled in an approach that mirrors the development of a numerical simulation model, support and interaction with other frameworks such as GIS is best maintained with a standalone library and a formal problem description specifically designed to constrain the general geophysical domain discretisation problem.

Chapter 6

Generalised unstructured spatial discretisation for geophysical models

6.1 Constraints for mesh generation in geophysical domains

The contrast in dominant dynamical processes that characterise geophysical systems, split in orthogonal directions parallel and perpendicular to the local gravitational acceleration $\boldsymbol{g}$, leads to a spatial decoupling that restricts the parameter space of general spatial domains $\Omega \in \mathbb{R}^3$. Meshes of geophysical domains can be built differently in these distinct directions in order to well-support the associated dynamics, with mesh characteristics on the geoid plane considered independently of those in the perpendicular direction of $\boldsymbol{g}$. A formal description of the heterogeneous set of constraint functions, homeomorphic mappings and topological spaces, required to fully describe geophysical model domain spatial discretisations, is developed and detailed in Candy [2017], of which a summary of the key outcome follows.

Constraints: *The spatial domain discretisation for a computational geophysics simulation in a domain $\Omega \subset \mathbb{R}^3$, requires the constraint of*

1. ***Geoid boundary representation*** *Γ_g, of the geoid surface $\Omega_g \subset \mathbb{R}^3$, inclusive of the maximal extent of Ω perpendicular to $\boldsymbol{g}$. Under a homeomorphic projection ξ, this is considered as the chart $\Omega' \subset \mathbb{R}^2$, such that the boundary Γ' is described by*

   $$\Gamma' : t \in \mathbb{R} \mapsto \zeta(t) \in \mathbb{R}^2, \tag{6.1}$$

   an orientated vector path of the encompassing surface geoid bound defined in two-dimensional parameter space.

2. ***Geoid element edge-length resolution metric*** *for dynamics aligned locally to a geoid, described by the functional*

   $$\mathcal{M}_h : \boldsymbol{x} \in \Omega' \mapsto \mathcal{M}_h(\boldsymbol{x}) \in \mathbb{R}^2 \times \mathbb{R}^2. \tag{6.2}$$

3. ***Boundary and region identification****, prescribed by*

   $$n_{\Gamma'} : t \in \mathbb{R} \mapsto n_{\Gamma'}(t) \in \mathbb{Z}, \text{ and} \tag{6.3}$$

   $$n_{\Omega'} : \boldsymbol{x} \in \Omega' \mapsto n_{\Omega'}(\boldsymbol{x}) \in \mathbb{Z}, \text{ respectively.} \tag{6.4}$$

4. ***Surface bounds****, height maps defined on the surface geoid domain, described by the functions*

   $$f, g : \boldsymbol{x} \mapsto \mathbb{R} \quad \forall \boldsymbol{x} \in \Omega'. \tag{6.5}$$

5. ***Vertical element edge-length resolution metric*** *for dynamics in the direction of gravitational acceleration (e.g. buoyancy driven), described by the functional*

   $$\mathcal{M}_v : \boldsymbol{x} \in \Omega \mapsto \mathcal{M}_v(\boldsymbol{x}) \in \mathbb{R}. \tag{6.6}$$

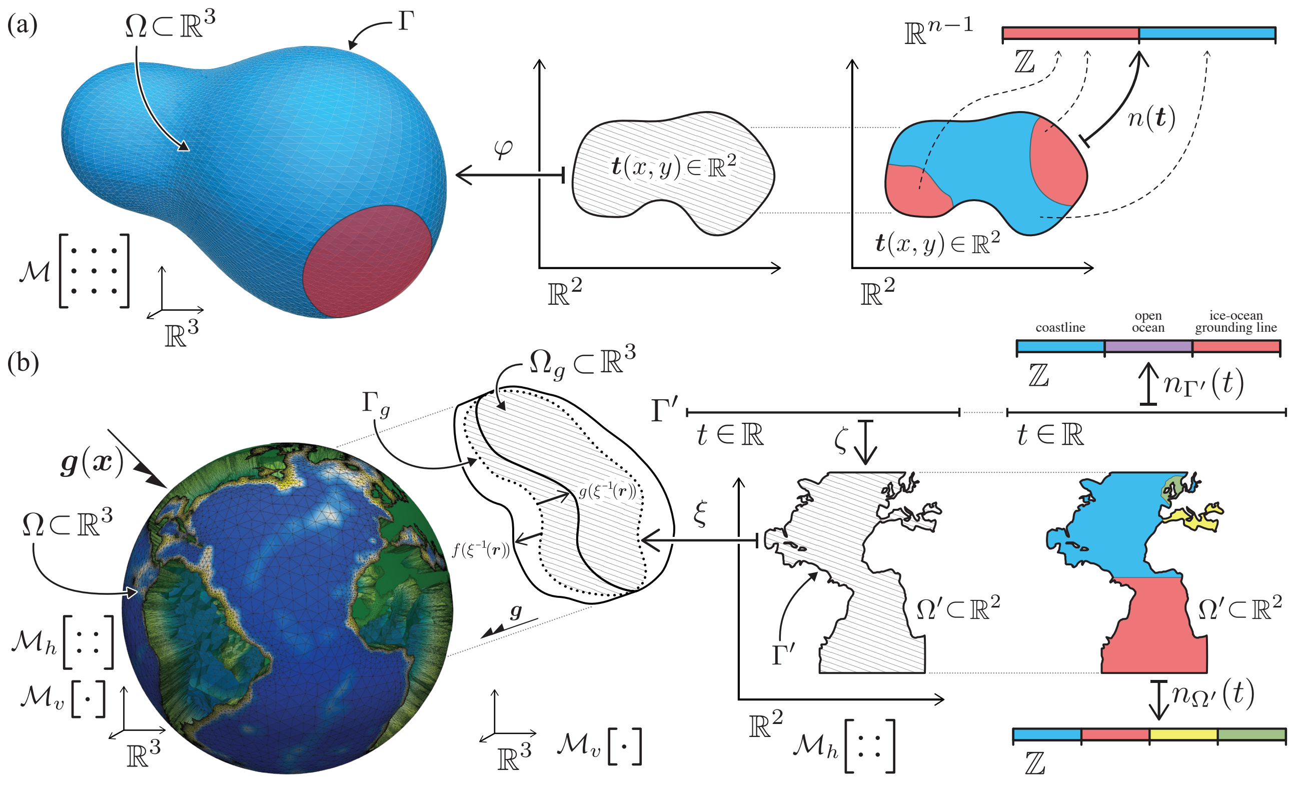


Figure 6.1: Breakout schematic of mesh generation for (a) general unstructured spatial discretisation and (b) typical geophysical domains, referencing constraint 1 of section 6.1.

6.2 Decoupled mesh development

The spatial decoupling permits discretisation in two stages corresponding to directions parallel and perpendicular to the local gravitational acceleration (refer to figure 9.1). Firstly, the 'horizontal' geoid surface domain discretisation problem is solved under constraints 1–3 using the surface geoid boundary representation Γ' (6.1), geoid element edge-length metric $\mathcal{M}_h$ (6.2), with boundary and region identifications, $n_{\Gamma'}$ (6.3) and $n_{\Omega'}$ (6.4) respectively, such that

$$h\colon \{\Gamma', \mathcal{M}_h, n_{\Gamma'}, n_{\Omega'}\} \mapsto \mathcal{T}_h, \tag{6.7}$$

a tessellation of $\Omega' \subset \mathbb{R}^2$, with identification elements.

Secondly, if needed, this is followed by discretisation in a direction aligned with gravitational acceleration. The constraints 4 and 5, describing the surface bounds f and g (6.5) and vertical edge-length metric $\mathcal{M}_v$ (6.6), together with the surface geoid discretisation $\mathcal{T}_h$ (6.7), forms a discretisation problem that is solved through the process

$$v\colon \{\mathcal{T}_h, f, g, \mathcal{M}_v\} \mapsto \mathcal{T}, \tag{6.8}$$

to give the full domain discretisation of $\Omega \subset \mathbb{R}^3$, with identification elements.

Chapter 7

The nine tenets of geophysical mesh generation

Accompanying the constraints, Candy [2017] identifies the nine attributes listed in table 7.1 as key to geophysical mesh generation processes.

---

1. Accurate description and **representation of arbitrary and complex boundaries** such that they are contour-following to a degree prescribed by the metric size field, with aligned faces so forcing data is consistently applied (Γ', f, g).
2. **Spatial mesh resolution** to minimise error; with efficient aggregation of contributing factors, ease of prototyping and experimentation of metric functions and contributing fields, over the entire extent of the bounded domain ($\mathcal{M}_h$, $\mathcal{M}_v$).
3. Accurate geometric **specification of regions** and **boundary features**; to provide for appropriate interfacing of regions of differing physics, model coupling and parameterisation application ($n_{\Omega'}$, $n_{\Gamma'}$).
4. **Self-consistent**, such that all contributing source data undergoes the same pre-processing, ensuring self-consistency is inherited.
5. **Efficient drafting and prototyping** tools, such that user time can be focused on high-level development of the physics and initialisation of the modelled system.
6. **Scalability**, with operation on both small and large datasets, facilitating the easy manipulation and process integration, independent of data size.
7. **Hierarchy of automation**, such that individual automated elements of the workflow can be brought down to a lower-level for finer-scale adjustments.
8. **Provenance** to ensure the full workflow from initialisation to simulation and verification diagnostics are reproducible.
9. **Standardisation of interaction** to enable interoperability between both tools and scientists.

Table 7.1: The nine tenets of geophysical mesh generation. Solutions to the spatial discretisation of geophysical model domains need to address these nine attributes [from Candy, 2017].

Chapter 8

Self-consistent appraoch

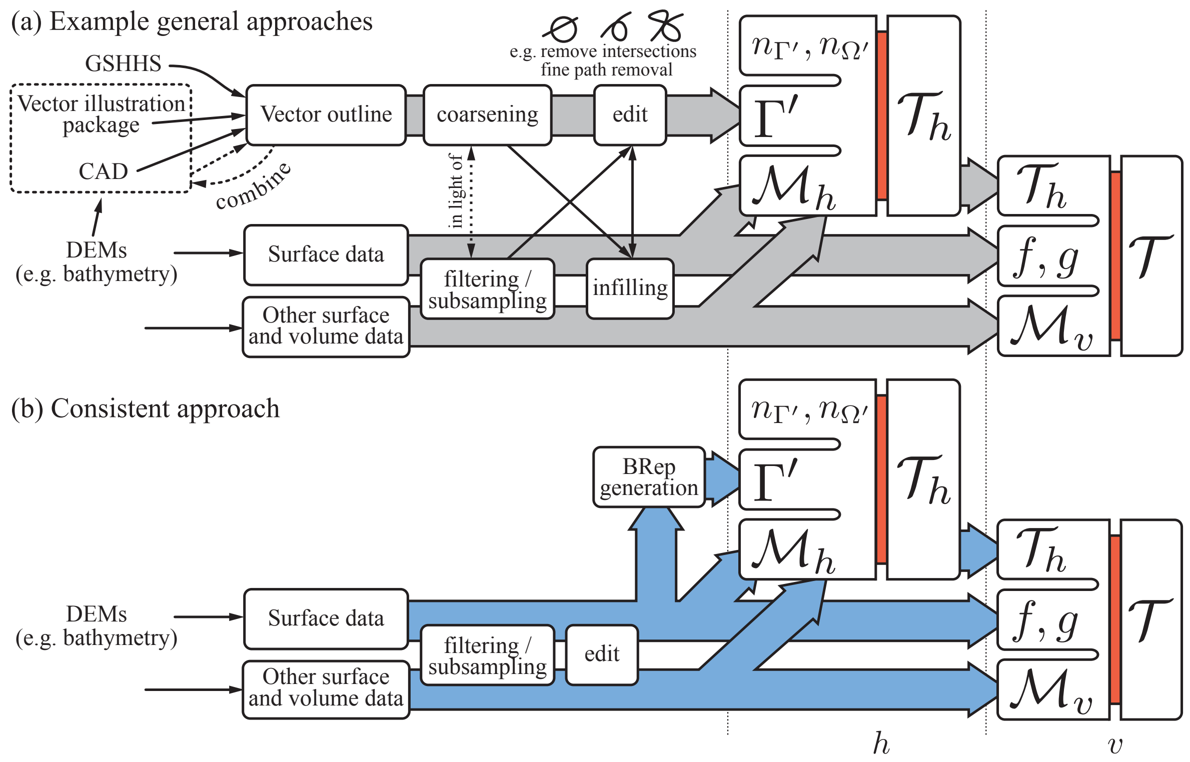


Figure 8.1: A schematic illustrating the approaches to unstructured mesh generation for geophysical models, highlighting (a) the case in general and (b) the consistent approach introduced here, with data pathways coloured in grey and blue, respectively [from Candy, 2017]. The connecting red sections represent spatial discretisation processes h (6.7) and v (6.8), that are fully constrained by the heterogeneous set of parameters provided to the left, and result in the discretisation to the right as an output. Here for example, h could be provided by an established Delaunay triangulation implementation to generate the two-dimensional tessellation $\mathcal{T}_h$, and v an advancing front algorithm to extrude this tessellation $\mathcal{T}_h$ out to a three-dimensional mesh $\mathcal{T}$.

Chapter 9

Boundary Representation Markup Language

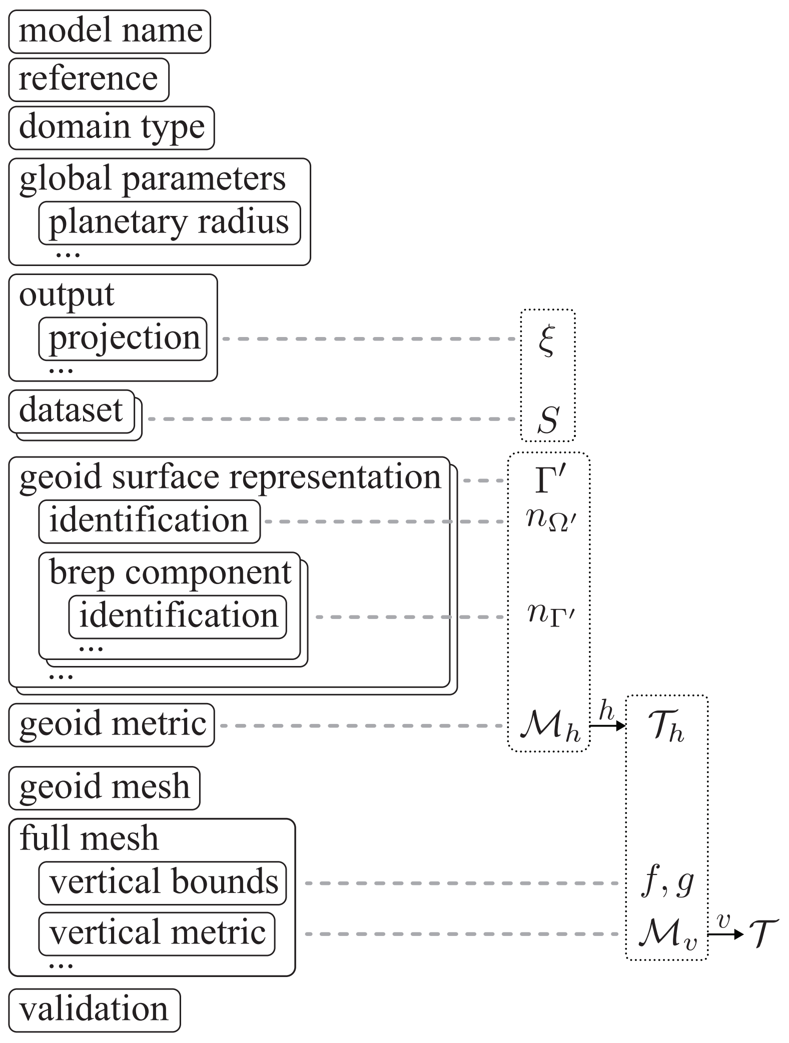
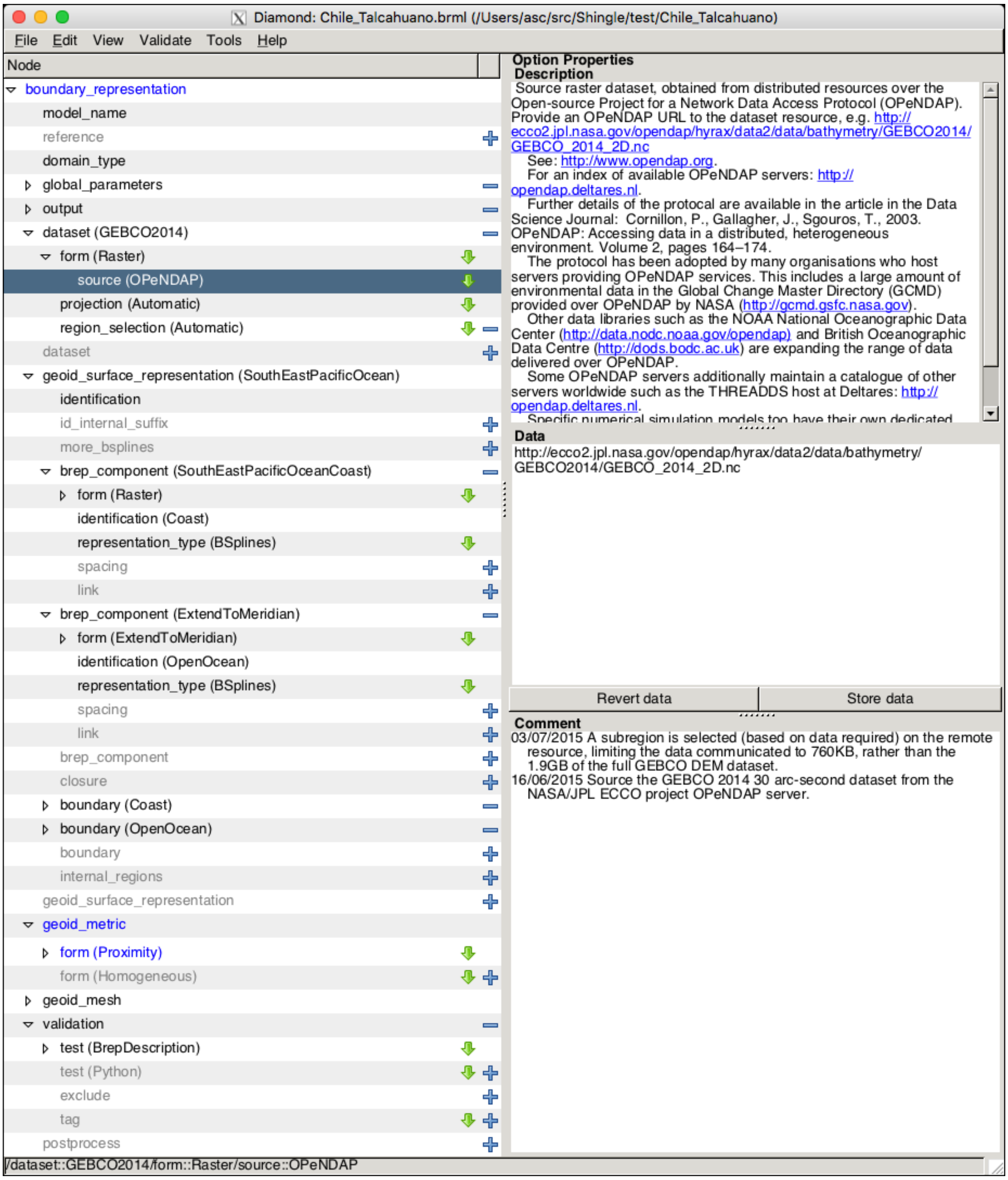


Figure 9.1: Overview layout of geophysical domain mesh constraint description highlighting extensible dynamic components and correspondence to source data S, projection ξ and constraints 1 – 5.


Diamond: Chile_Talcahuano.brml (/Users/asc/src/Shingle/test/Chile_Talcahuano)

```
▽ boundary_representation
    model_name
    reference
    domain_type
  ▷ global_parameters
  ▷ output
  ▽ dataset (GEBCO2014)
    ▽ form (Raster)
        source (OPeNDAP)
      projection (Automatic)
      region_selection (Automatic)
    dataset
  ▽ geoid_surface_representation (SouthEastPacificOcean)
      identification
      id_internal_suffix
      more_bsplines
    ▽ brep_component (SouthEastPacificOceanCoast)
      ▷ form (Raster)
        identification (Coast)
        representation_type (BSplines)
        spacing
        link
    ▽ brep_component (ExtendToMeridian)
      ▷ form (ExtendToMeridian)
        identification (OpenOcean)
        representation_type (BSplines)
        spacing
        link
      brep_component
      closure
    ▷ boundary (Coast)
    ▷ boundary (OpenOcean)
      boundary
      internal_regions
    geoid_surface_representation
  ▽ geoid_metric
    ▷ form (Proximity)
      form (Homogeneous)
  ▷ geoid_mesh
  ▽ validation
    ▷ test (BrepDescription)
      test (Python)
      exclude
      tag
    postprocess
/dataset::GEBCO2014/form::Raster/source::OPeNDAP
```

Option Properties

Description

Source raster dataset, obtained from distributed resources over the Open-source Project for a Network Data Access Protocol (OPeNDAP). Provide an OPeNDAP URL to the dataset resource, e.g. http://ecco2.jpl.nasa.gov/opendap/hyrax/data2/data/bathymetry/GEBCO2014/GEBCO_2014_2D.nc

See: http://www.opendap.org.

For an index of available OPeNDAP servers: http://opendap.deltares.nl.

Further details of the protocal are available in the article in the Data Science Journal: Cornillon, P., Gallagher, J., Sgouros, T., 2003. OPeNDAP: Accessing data in a distributed, heterogeneous environment. Volume 2, pages 164–174.

The protocol has been adopted by many organisations who host servers providing OPeNDAP services. This includes a large amount of environmental data in the Global Change Master Directory (GCMD) provided over OPeNDAP by NASA (http://gcmd.gsfc.nasa.gov).

Other data libraries such as the NOAA National Oceanographic Data Center (http://data.nodc.noaa.gov/opendap) and British Oceanographic Data Centre (http://dods.bodc.ac.uk) are expanding the range of data delivered over OPeNDAP.

Some OPeNDAP servers additionally maintain a catalogue of other servers worldwide such as the THREADDS host at Deltares: http://opendap.deltares.nl.

Specific numerical simulation models too have their own dedicated

Data

http://ecco2.jpl.nasa.gov/opendap/hyrax/data2/data/bathymetry/GEBCO2014/GEBCO_2014_2D.nc

Comment

03/07/2015 A subregion is selected (based on data required) on the remote resource, limiting the data communicated to 760KB, rather than the 1.9GB of the full GEBCO DEM dataset.
16/06/2015 Source the GEBCO 2014 30 arc-second dataset from the NASA/JPL ECCO project OPeNDAP server.

Figure 9.2: An example view of the Graphical User Interface Diamond inspecting the hierarchical tree of option parameters that fully constrain the geophysical domain mesh problem. Each node is shown in context on the left, with their option properties presented on the right, including raw data and the possibility to note comments. This is guided by the BRML schema developed and supplied with Shingle, which additionally provides the fuller self-describing option descriptions shown in the top right. Options down the tree highlighted in blue are mandatory and guide the user to defining a complete set of constraints.

```xml
<?xml version='1.0' encoding='utf-8'?>
<boundary_representation>
  <model_name>
    <string_value lines="1">Chile_Talcahuano</string_value>
  </model_name>
  <global_parameters/>
  <output>
    <projection>
      <string_value>LatLongWGS84</string_value>
    </projection>
  </output>
  <dataset name="GEBCO2014">
    <form name="Raster">
      <source name="OPeNDAP" file_name="http://ecco2.jpl.nasa.gov/ opendap/hyrax/data2/data/bathymetry/GEBCO2014/
  ↪   GEBCO_2014_2D.nc"/>
    </form>
    <projection name="Native"/>
    <region_selection name="Automatic"/>
  </dataset>
  <geoid_surface_representation name="SouthEastPacificOcean">
    <identification>
      <integer_value rank="0">9</integer_value>
    </identification>
    <brep_component name="SouthEastPacificOceanCoast">
      <form name="Raster">
        <source name="GEBCO2014"/>
        <region>
          <longitude>
            <minimum>-77.0</minimum>
            <maximum>-71.0</maximum>
          </longitude>
          <latitude>
            <minimum>-40.0</minimum>
            <maximum>-32.0</maximum>
          </latitude>
        </region>
        <contourtype name="coastline0m"/>
        <comment>Simple single bounding box centred about the epicentre 35.909S 72.733W.</comment>
      </form>
      <identification name="Coast"/>
      <representation_type name="BSplines"/>
    </brep_component>
    <brep_component name="OpenMeridian">
      <form name="ExtendToMeridian">
        <longitude>
          <real_value rank="0">-77.0</real_value>
        </longitude>
      </form>
      <identification name="OpenOcean"/>
      <representation_type name="BSplines"/>
    </brep_component>
    <boundary name="Coast">
      <identification_number>
        <integer_value rank="0">3</integer_value>
      </identification_number>
    </boundary>
    <boundary name="OpenOcean">
      <identification_number>
        <integer_value rank="0">4</integer_value>
      </identification_number>
    </boundary>
  </geoid_surface_representation>
  <geoid_metric>
    ...
  </geoid_metric>
  <validation>
    <test name="BrepDescription" file_name="data/Chile_Talcahuano.geo.bz2">
      <compressed/></test>
    <test name="NodeNumber"> ... </test>
  </validation>
</boundary_representation>
```

Figure 9.3: Example domain discretisation description, in a self-describing BRML description file (with a few parts marked ... skipped). This is a human-readable translation of the simple description (∗) under the formal grammar of the schema that defines the geophysical domain discretisation constraint space. This file is examined by the GUI in figure 9.2 and, on straight-forward and automated processing by Shingle, produces the simulation-ready spatial discretisation of figure 11.2.

9.1 Unstructured domain discretisation a model problem

The functional forms (6.1) – (6.6) of the unstructured meshing problem require a range of types of data, from more standard two-dimensional raster maps, to tensors and orientated vector paths. It is a challenge to manage this heterogeneous collection of parameters (tenets 5 and 8), such that they are handled consistently (tenet 4) and for the level of complexity that can be encountered (tenets 6 and 7). This is in contrast with the structured mesh case, which requires relatively simple data of the same format as its inputs: a two-dimensional Digital Elevation Map (DEM) raster dataset supplying a two-dimensional raster mask, for example.

Mesh specification in the unstructured case, with flexibility to include conforming boundaries, is much more like the initialisation of a numerical simulation model. This typically contains a heterogeneous set of functions: those defined over $\mathbb{R}^3$ initialising or forcing full fields, together with boundary conditions defined on surfaces in $\mathbb{R}^2$ and potentially line and point sources, or full field functions of reduced rank such as the gravitational acceleration parameter, or value of a bulk eddy viscosity, for example. Mesh descriptions and constraints are only going to become more complex as simulation models include a larger range of spatial scales and physical processes. Moreover, like a simulation model, unstructured mesh generation includes calculations that can be computationally demanding. The generation of conforming boundary representations is no longer a simple binary operation identifying which elements lie in the simulation domain through mask fields. Similarly, the construction of domain discretisations with variable element sizes contains many more unknowns in the unstructured case than the corresponding local cell-division approaches typically used to increase spatial resolution in the structured case.

In light of this, Shingle takes the approach that domain discretisation specification and generation is best considered as a model problem. Formalised, the output mesh is the solution of a discretisation problem under a heterogeneous parameter space of constraints.

9.2 Spud constraint space management

Much like numerical model input parameter specification, mesh generation is often overlooked, and a secondary consideration to the dynamical core of a numerical model. Typically inputs are ad hoc, model-specific, plain text files containing name lists that are expanded as a model develops. For only but simple cases, this leads to model interfaces (and their associated pre- and post-processing tools) that are difficult to maintain and simulation setups that are not easily shared and understood.

This problem of model input parameter specification is considered in Ham et al. [2009], together with the proposed solution Spud. This provides a generalised, model-independent method of describing all constraints to a model problem, that is dynamic, easily extensible with a hierarchical context for parameters. Formal grammars guide user input, minimise errors and formalise parameter specification.

9.3 Constraint space description

The options available to describe a mesh discretisation are typically defined by model interfaces. These tend to be ad hoc and unportable, tied directly to numerical simulation codes. Initialisation tools then require their own implementation to interpret and write model options, which is prone to error and potential inconsistencies.

Existing file formats have been used, and their syntax overloaded, to describe geophysical spatial discretisations. Ice sheet domains are built up using a Constructive Solid Geometry (CSG) approach within the COMSOL [COMSOL, 2014; Li et al., 2009] multi-physics modelling environment in Humbert et al. [2009]. The GeoCUBIT [Casarotti et al., 2008] branch of CUBIT developed for seismic inversion domains, and a plugin for Gmsh [Geuzaine and Remacle, 2009] to enable the creation of domains bounded by paths from the Global Self-consistent, Hierarchical, High-resolution Geography [GSHHG, Wessel and Smith, 1996]. Extensions to GIS [e.g. Candy et al., 2014] enable a flexible development of geoid surface boundary representations. Extensibility of these frameworks for the purposes of geophysical domain discretisation and model initialisation is limited, with for example GIS frameworks being built up from working on two-dimensional raster fields. Similarly, project files associated with GIS do not contain all of the information required to fully constrain a spatial discretisation problem, and moreover, it is not possible to include the high-level natural language functional descriptions proposed here. As Candy et al. [2014] demonstrates though, GIS methods can benefit geophysical domain development, and their role is included in the schematic figure 5.1.

Use of Spud enables a description of model option parameter space to be considered separately. This is constructed in a schema file, a machine readable specification of which options are expected, their type and context, and how they should be read: A formal grammar to be used to describe model constraints. The constraints 1 – 5 that fully describe the geophysical domain discretisation problem have been structured into a schema. A schematic of the included components and their relationship to required constraints is shown in figure 9.1.

This is a single hierarchical and formal description of the constraint space, and more generally the options available to the user in generating a mesh. As illustrated in figure 5.1, it is part of Shingle and is central to how components of the approach interact with BRML files that describe a particular meshing problem. At the simplest, highest level use of Shingle, this is transparent to the user. For more advanced use and development, it provides a centralised and language-based description of the constraint space that all other parts of Shingle, and the geophysical mesh generation process, depend.

9.4 Dynamic, hierarchical parameter description

Just like the case of a numerical model, there are a wide range of possible options in mesh generation, even when restricted to geophysical problems. The BRML schema builds on the general schema language for simulation models prepared in Ham et al. [2009], to give an option-complete language for the mesh generation problem. This is exactly the type of purpose Spud is intended for and other current models in development are adopting this approach to formally describe model constraint spaces, like for example the new TerraFERMA model of Wilson et al. [2016].

This caters for options which may be specified multiple times, at potentially varying levels of option hierarchy in multiple contexts. For example, as the block diagram of figure 9.1 highlights, a simulation domain can contain multiple geoid surfaces Γ', each with potentially multiple boundary representation components (e.g. simple orientated polylines with identification). BRML is an XML language, and by nature is hierarchical and extensible. With this structure, and guided by what the schema permits (itself representing the constraints 1 – 5), it is easy to dynamically add, repeat, expand and remove options and groups of options whilst in context.

As an example, use of the Spud framework immediately provides access to the Diamond GUI which enables easy editing and drafting of new domain discretisations. This GUI uses the schema file (see figure 5.1) to guide navigation of the option tree. Through this the GUI knows to expect at least one definition of a geoid surface Γ', for example, and a specification of a geoid metric $\mathcal{M}_h$ (and requires these from the user). Additional geoid surfaces or more feature-rich boundary representation components are easily added and built up at a later stage, dynamically increasing the complexity of the mesh generation problem.

9.5 Option tree cross-references

Options are structured into a hierarchical tree within the BRML description. The grouping of constraints 1 – 5 and decoupling (section 6.2) are naturally structured in this way, as figure 9.1 highlights. This is much like numerical simulation model options parameters, which motivated the development of Spud and adoption of an underlying XML-based language.

In some cases there exist dependencies across the option tree, and these are achieved through attribute names. For instance, the choice has been made to centralise source dataset definitions. These are named (e.g. 'GEBCO2014' in figures 9.2 and 9.3) and this name referred back to whenever the data is required. This is also used to assign potentially multiple boundary representation component sections to the same named boundary identification (e.g. the 'Coast' and 'OpenOcean' named identifications of figures 9.2 and 9.3).

This also allows component boundary representations sections to be used multiple times. This is required, for example, when distinct physical regions meet at an interface (e.g. the open ocean meets an ice sheet) and share a boundary. The component boundary representation section defining the interface can then be referred to out of the order defined by the hierarchy, and from potentially separate parent geoid surface representation Γ' (where for instance Γ'_o and Γ'_i are setup to represent neighbouring geoid surface representations for the ocean and ice, respectively).

9.6 Natural language descriptions

Domains for geophysical simulations are typically described with reference to bounding lines on orthodromes such as meridians and parallels, together with global or segments of contours such as a 0m coastline, for example. More generally, geographic features are identified with a similar combination. The Southern Ocean for example, is defined extending up from the Antarctic coast to the 60°S parallel, and the Atlantic and Indian Oceans divided at the 20°E meridian.

This is the natural way to identify bounds for geophysical models. Setting up these geographic bounds and including all features contained within in a format suitable for meshing algorithms can be a time consuming, difficult to edit and repeat, ad hoc process. Shingle automates this and from a basis of natural language definitions typically used in geophysical modelling studies.

The original consistent boundary representation generation approach described in Candy [2017] enabled sections of contours to be selected and domains extended meridionally to parallels. This has been generalised significantly to allow a wide range of arbitrary bounds described with natural language definitions. Moreover these can be defined multiple times, and in context with hierarchy available within the BRML description. In the example presented in figures 9.2 and 9.3 the boundary representation can be seen to include two components: a section of the Chilean coastline and a second extending the domain out to a meridian at 7°W, mirroring those in the description (∗).

9.7 Arbitrary and discrete descriptions

More flexible functional descriptions can be made within the BRML written directly in Python. This again in a relatively readable form, using primitives such as the positions 'longitude' and 'latitude', or Universal Transverse Mercator (UTM) coordinates 'x' and 'y'. This can be used to describe an arbitrary orthodrome, for example.

In addition to this, the natural language basis can be supplemented with raw discrete data types such as orientated polylines from the GSHHG database, mapping databases (e.g. the UK national Ordnance Survey OpenData resource) or those developed directly in a GIS as Candy et al. [2014] demonstrates, bounding a domain to the complex UK coastline together with the fine man-made structures of Portland Harbour. The high fidelity boundary representation is not only built up from components constructed on-the-fly from functional forms referencing geographic features, but also discretised forms containing an explicit description of domain constraints, if needed (see figure 5.1). These are available through the central dataset section of the option hierarchy (figure 9.1), and accessed from local or distributed resources.

9.8 Self-describing constraint options

The constraint space description developed in the BRML schema is self-describing, containing a verbose description of each option. This information can presented alongside options in the GUI (see the top right of figure 9.2, for example) or reported for any option errors occurring at run time, again from this centralised constraint space descriptor resource, the schema. In this way the schema, and as a result the GUI, act as a manual, directly supporting users as mesh options are made.

From the developer's perspective, this Spud based approach means new features can be added with minimal code changes. The XML based structure means codes focus on patterns of options. The schema defines what expected and the code loops through the hierarchy following well-defined patterns, picking up options from a corresponding in-memory dictionary tree.

For the user, mesh generation with real fractal-like boundaries can be as simple as selecting a coastline segment by a bounding box and on the other side a bounding orthodrome, with choice of element edge-length metric (see figure 11.2).

9.9 Provenance record

A complete description of the domain discretisation problem is a fundamental requirement if an accurate record of provenance is to be made, and this is provided by the BRML file. These BRML files alone are themselves easily parsable XML based problem description files, human-readable with structure. This is focused on a textual natural language problem description and is lightweight as a result such that changes are easily tracked with version control systems such as Git and SVN.

Together with the problem description, the BRML maintains details of authors responsible for their creation, contact details, comments including timestamped notes on past changes made in development (seen in figures 9.1 and 9.2). This is similar to the record kept within the global attribute metadata contained in NetCDF headers, which is supplemented through operations performed on the data with tools such as the Geospatial Data Abstraction Library [GDAL, 2016]. The ADCIRC hydrodynamic circulation model [Westerink et al., 2008] makes a record of this type of information in its NetCDF output, inherited from its initialisation namelist files. Shingle records this information in output where possible, notably the high fidelity boundary representation, supplementing it with a record of the library release version and unique repository abbreviated commit hash. Unique identifiers of other libraries are also recorded, such as the version of the meshing tool employed (e.g. Gmsh).

Chapter 10

Source data management

Data contributing to discrete domain characterisations can be large in size, difficult to distribute efficiently and computationally costly to process. The current version of the global bathymetry dataset GEBCO [2014] containing only elevation is currently 1.9 GB in size, for example. Efforts are growing to provide a complete provenance record of numerical model simulations, with direct instructions from research funders requiring a research data management plan [NWO Data Management Protocol, 2014] and in general, accountability from the public, it is important to detail data source origin and content accurately.

Options for the management of mesh generation source data range from:

1. Recast data into form suitable for distribution and share with BRML description.
2. Distribute processed datasets with BRML irrespective of size.
3. Begin from a standardised raw dataset, and conduct potentially computationally demanding processing as needed.
4. Refer to remote repositories of source data, such that data is downloaded and processed on demand.

Often this data processing stage of the mesh generation process is not well-described, and difficult to reproduce, with filtering, subsampling and agglomeration operations only loosely outlined.

Modern data descriptors support a record of provenance [such as the 'history' field embedded in NetCDF, Rew et al.], so it would be possible to record the filtering, subsampling and other processing here or within the BRML.

The purpose of the BRML description of constraints is to provide an accurate description of the meshing problem. It is not the intent to reinvent new standards for data description. Along this line of design, with a focus on provenance record and how data is handled, and noting the computational demands and connectivity speeds that affect options 3 and 4 above will continue to improve in the future, the approach is made to depend directly on raw, standard and potentially remote data sources.

10.1 OPeNDAP integration

The problem of efficient access to large remotely hosted data sources is tackled by Cornillon et al. [2003] which describes OPeNDAP (Open-source Project for a Network Data Access Protocol). The protocol has since been adopted by many organisations who host servers providing OPeNDAP services. This includes a large amount of environmental data in the Global Change Master Directory (GCMD) provided over OPeNDAP by NASA[1]. Other data libraries such as the NOAA National Oceanographic Data Center[2] and British Oceanographic Data Centre[3] are expanding the range of data delivered over OPeNDAP. Some OPeNDAP servers additionally maintain a catalogue of other servers worldwide such as the THREADDS host at Deltares[4]. Specific numerical simulation models too have their own dedicated servers to host output such as the ocean models HYCOM[5] and ROMS[6].

This has typically been applied to sharing geophysical model output data in combination with the [NetCDF Rew et al.] and Climate and Forecast [CF, Gregory, 2003] metadata standards [Hankin et al., 2010], for

[1]http://gcmd.gsfc.nasa.gov
[2]http://data.nodc.noaa.gov/opendap
[3]http://dods.bodc.ac.uk
[4]http://opendap.deltares.nl
[5]http://tds.hycom.org
[6]http://megara.tamu.edu:8080, http://tds.marine.rutgers.edu

intercomparisons and post-processing analysis. Here we apply OPeNDAP to model initialisation. In Shingle, this OPeNDAP negotiation is achieved using the standard Python library pydap. In this way Shingle can request fundamental operations are applied to distributed datasets before they are delivered for further processing, picking out required fields and regions of interest to reduce the size of data communicated. A description of further processing such as subsampling and filtering is then maintained in the BRML and executed through standardised Python wrappers to established geospatial tools such as GDAL [2016]. A reference in place for the GEBCO [2014] data source hosted on the NASA/JPL ECCO OPeNDAP server is made in figure 9.3, where the region of interest (for cropping on the remote server) is automatically established by its use further down in the tree.

Keeping the BRML focused on problem description, with references to source data, ensures it is lightweight and portable. Iterative adjustments to the mesh generation are also then made with changes to descriptions rather than data. Furthermore, these are then easily managed in version control systems.

This additionally ensures the verification test engine is lightweight and apart from a dependence on standard software libraries, and a connection to OPeNDAP servers, is self-sufficient and can be easily be setup and used independently.

Constraints built from distributed resources are encouraged, but to engage with existing mesh generation workflows and as a pragmatic solution, source files can be cached or local files used directly (see figure 5.1).

10.2 Self-consistent boundary representation development

Shingle applies the self-consistent approach to mesh generation developed in Candy [2017]. Within the BRML description this is emphasized through a central data source definition (seen in figures 9.1 – 9.3), rather than external sources brought in directly at different levels in the hierarchy and correspondingly in the generation process (figure 9.1). It is then easier to ensure datasets and their component fields undergo the same pre-processing to generate high fidelity constraints that are consistent, and a solution spatial discretisation that is self-consistent.

Data used to construct the spatial domain discretisation is commonly a DEM describing a surface through perturbations from a reference geoid surface (e.g. to establish a geoid surface boundary representation), but is not limited to this form, with for example Candy [2017] developing a mesh optimised to the mean track of the ACC, based on currents in the Southern Ocean.

Chapter 11

Shingle library framework

(a)

```
from shingle import SpatialDiscretisation, Dataset, Boundary
# Set up constraints
R = SpatialDiscretisation(name='NorthSea')
R.SetProjection('UTM', -3, 52) # alternatively zone='30U'
gebco = Dataset(type='raster', source='opendap', url='...', region=[-12,14,45,62])
coast = Boundary('coast', id=3)
S = R.AddSurface()
S.AddBoundaryComponent(source=gebco, contour='ocean0m', id=coast)
   ...
M = R.Discretisation()
M.Save('NorthSea.msh')
```

```
# Modify boundary representation output projection
import pyproj
p = pyproj.Proj('+proj=utm +zone=30U +ellps=WGS84 +datum=WGS84 +units=m")
R.SetProjection(p)
R.Save('NorthSea_UTM30U.brml')
```

```
# Simple parameter sweep example
from shingle import Load
R = Load('Weddell_Sea.brml')
S = R.GetSurface('SouthernOcean')
B = S.GetBoundaryComponent('OpenParallel')
for latitude in [float(x) for x in xrange(-75,-65,2)]:
  B.ExtendToParallel(latitude)
B.Save('Weddell_Sea_and_%.0f.brml' % latitude)
```

Figure 11.1: Example interactions with the Shingle Python library LibShingle. (a) Using natural language constructs native to Shingle, counterparts to BRML entries. (b) Together with objects native to external libraries. (c) Loading, extending and saving descriptions from BRML.

11.1 Built on standard libraries

The library LibShingle is written in Python and uses standard libraries for operations where possible. It can simply be used transparently through the Shingle executable to interpret the constraints specified in BRML file descriptions. For lower level more advanced use building up constraints for more complex setups or in prototyping natural language objects for automating the inclusion of new geographic features, interaction can be made directly with the LibShingle library as figure 5.1 illustrates.

Mirroring the BRML constraint description (overviewed in figure 9.1), the library contains natural language based objects that can be built up in code to construct components of a mesh generation problem, including boundary representations and element edge length metrics. The mesh problem can then be solved under these constructed constraints all within a Python context.

LibShingle uses the open source Python shapely library (refer to figure 5.1) to handle polyline imports and manipulations. The Scientific.IO library is relied on to efficiently process raster NetCDF files. The homeomorphic projections to the charts required in the mesh generation process [see Candy, 2017], such as ξ

of (6.1) are interpreted and managed by the Proj.4 Python library pyproj. Geospatial operations can be made by both high-level Shingle objects, or built up with GDAL operations through its Python osgeo interface.

Although the use of external libraries may require updates to Shingle in the future to maintain compatibility, this is minimal compared to the benefits of using standardised implementations (tenet 9), that have community effort to ensure ongoing support with operating systems and interaction with other software and methods.

11.2 Low-level interaction through Python objects

In addition to the ongoing support from standard libraries in high-level use, Shingle has been written to interact directly with external libraries. Objects such as pyproj projections, GDAL operations, surface and polyline descriptions can be used interchangeably with LibShingle. An example bringing in a UTM projection setup externally using the standard library pyproj is shown in figure 11.1(b). This supplements the high-level text-based natural language definitions available in the BRML, and a route to adding new high-level boundary representation BRML objects to LibShingle as needed.

11.3 Efficient parameter space exploration

In developing a new application study applying a numerical simulation model, it is common to iterate on a spatial discretisation until it is optimum and fit for purpose. This involves small changes in the constraints, exploring parameter space often through a loose bisecting binary search algorithm. This process can be rigorously implemented and automated with LibShingle, where modifications are guided by the schema describing the formal grammar of the constraint space through libspud. Figure 11.1(c) illustrates a simple template to modifying and generating a range of BRML mesh descriptions. The solution mesh discretised domains can be generated in the same way, and this could further be used to initiate numerical simulation runs.

This algorithmic formulation of constraints is easily extended to enable complex operations that are difficult to achieve with other approaches. For example, the loop of figure 11.1(c) is trivially extended to include a search algorithm exploring a parameter space to converge a domain discretisation on a required total number of nodes and hence degrees of freedom.

Being an XML based language, the BRML descriptions can also be simply interrogated and modified directly with standard XML libraries. This interaction is highlighted separately in figure 5.1.

11.4 Example applications of the library

The library has been used to develop spatial discretisations of:

- The global oceans.
- The Southern Ocean, directed by the mean position of the Antarctic Circumpolar Current (ACC).
- Ice and ocean domains of Antarctica, accurately meeting at common interfaces.
- Ice shelf ocean cavities, coupled to ice sheets (including Pine Island Glacier, Filchner-Ronne and the whole Antarctic region).
- Paleo ocean domains from reconstructed DEMs.
- North Sea coastal seas.
- Greenland and Antarctic ice sheets.

Please see section 1.4 for further details.

The library is additionally used to automatically review tests and collate images of output for a further verification. Example use can be found in the source files.

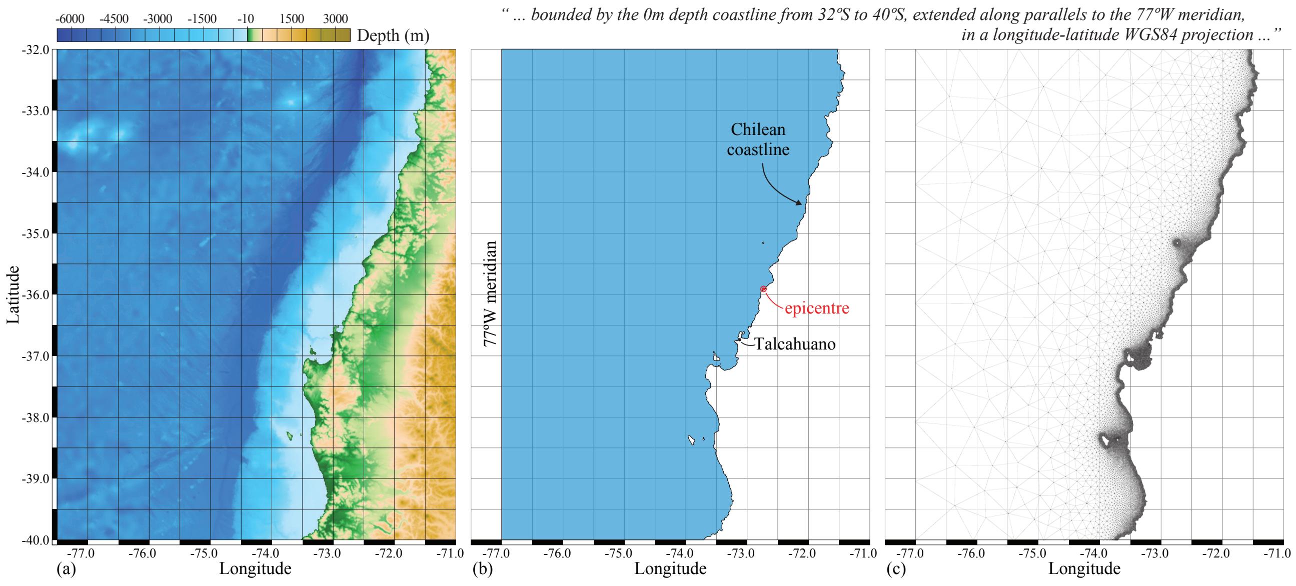

“ ... bounded by the 0m depth coastline from 32ºS to 40ºS, extended along parallels to the 77ºW meridian,
in a longitude-latitude WGS84 projection ... ”
Depth (m)
-6000 -4500 -3000 -1500 -10 1500 3000
Latitude
Longitude
(a)
(b)
(c)
Chilean coastline
epicentre
Talcahuano
77ºW meridian

Figure 11.2: Example simulation domain for modelling ocean wave propagation and tsunami inundation in the 2010 Chile M8.8 earthquake, centred at 35.9°S 72.7°W, approximately 100km north of Talcahuano. This domain is relatively simply described by (∗) in chapter 1 with constraints formally defined by the BRML of figure 9.3 (with some further description and corresponding formal BRML to constrain spatial resolution). Generation is a simple matter of translating the former into the latter under the formal grammar, with both being human-readable descriptions. Shingle automatically handles the details of defining a high fidelity boundary representation Γ' in (b) from the GEBCO [2014] DEM (a) and, notably here, includes island features to give a geoid surface representation with non-zero genus [following the approach of Candy, 2017], and further to automatically produce a simulation-ready meshed spatial discretisation $\mathcal{T}_h$ in (c).

11.5 Jupyter notebooks

Use of Shingle as a library is easily explored using a Jupyter notebook (`http://jupyter.org`) and several are provided in the source code as examples in `doc/jupyter`.

```
cd doc/jupyter
jupyter notebook
```

For further details on installing Jupyter, please refer to its documentation at:
`http://jupyter.readthedocs.io/en/latest/install.html`.

Chapter 12

Model, method and data interaction and interoperability

Shingle has been built with modules for high-level interactions, with established tools used in mesh generation. These are highlighted in figure 5.1, with a core link to the Gmsh library of meshing algorithms. Where possible interaction is achieved through standardised Python APIs, such as the Triangle Library Python Bindings [2014] for the Triangle [Shewchuk, 2002] library of Delaunay mesh algorithms. High fidelity boundary representation can be output in Gmsh format using a specific format writer developed within a collection of writer modules prepared within Shingle.

Similarly, fields supporting a meshed domain (e.g. initial full-field temperature state) can be output as unstructured VTK files, using a format writer extending standard VTK libraries. Data is written and stored efficiently in an XML based data format containing blocks of binary data compressed using the zlib library.

12.1 Model format writers

Models with non-standard data formats are supported through specific format writers. This modular approach enables new format writers (and readers) to be added as needed. As examples, Shingle includes modules to prepare initialisation files for the ADCIRC hydrodynamic circulation model and H2Ocean shallow water equation model.

As well as writing mesh solutions, the output writers are used for validation purposes and in the general purpose efficient prototyping (tenet 5). Output can be prepared for viewing alongside source data in geospatially valid context provided by GIS frameworks, with for example the resulting mesh and discrete bounds overlaid over DEMs directly within GIS [see Candy et al., 2014]. This is useful for a visual evaluation of conformity, to see how well geographic features are represented. For large discretisations, visualisations tools designed specifically for efficiently handling large unstructured datasets can be employed, such as Paraview[1], which is directly supported by Shingle using VTK.

Interaction at different levels is important to ensure a hierarchy of automation tenet 7. Particularly challenging meshing problems can, for example, easily be offloaded to more capable dedicated resources.

For quick visual inspection purposes, Shingle can automatically output an image of the geoid surface mesh discretisation.

12.2 Input readers

Parallel to the writer modules, Shingle includes readers. These are used to interact with meshing libraries where needed, loading in output mesh discretisations produced by Gmsh on-the-fly, for example. Additionally this can be used to support a wider range of data sources and initialisation. Standard data in NetCDF and shapefile form can be read. Readers here can import more complex heterogeneous data, including GIS projects with multiple layers containing a wide range of data types, for example.

[1]`http://www.paraview.org`

12.3 Embedding in model codes

As a Python library unifying boundary representation constraint and solution, LibShingle makes it possible to incorporate complex domain discretisation of real geophysical domains in overarching model control scripts, which is where development of new cutting-edge models is headed [see for example, Pelupessy et al., 2016; Rathgeber et al., 2015]. In this way the model supplements the problem constraints sent to LibShingle (see figure 13.3), dependent on numerical discretisations employed in the simulation model, and the BRML would be truly independent of specific models, a pure description of the boundary representation, resolution and identification. Moreover, interaction through the library enables models to handle the output discretisation directly as the Python objects constructed by Shingle, rather than an intermediate file object.

As Pelupessy et al. [2016] demonstrates, complex multi-model Earth system models can be created and coupled, and interactively monitored, on potentially a heterogeneous array of computational resources, all coordinated from a central a Python interface. LibShingle brings domain discretisation in real geometries to these type of extensible Earth system modelling frameworks.

Chapter 13

Verification and discretisation validation

A suite of verification tests are provided together with Shingle, along with the automated test engine detailed in section 13.2. A selection of geophysical domain discretisations described in BRML that form part of the test examples are shown in figures 1.1, 11.2, 13.1 and 13.2. Each test is evaluated using validation tests built into Shingle and their BRML descriptions, as outlined in section 13.1. The test engine can be used to verify a new install, and flexibly to support iterative mesh drafting and prototyping (tenet 5).

13.1 Self-validation

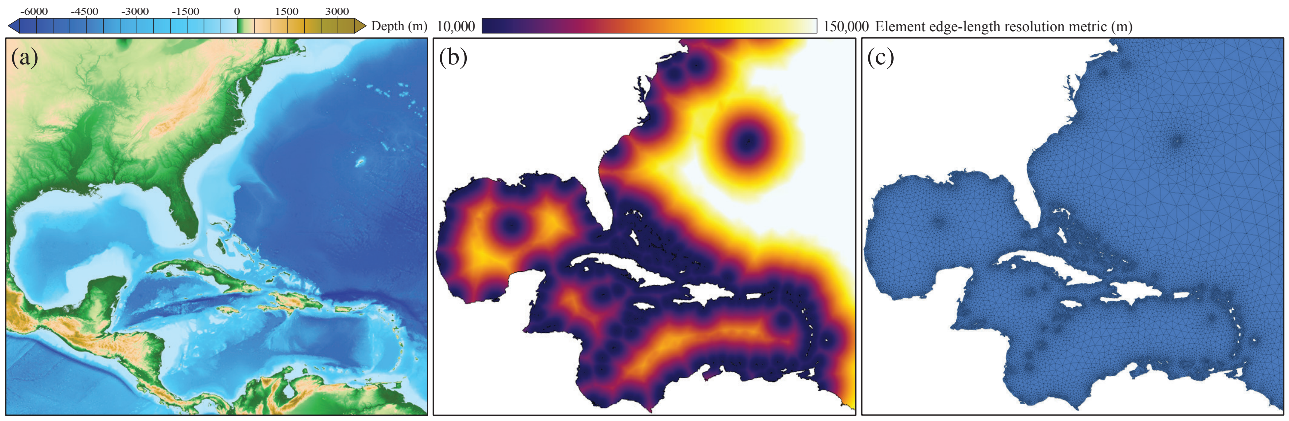


Figure 13.1: Simulation domain focused on the Caribbean Sea basin. (a) GEBCO [2014] DEM. (b) Surface geoid element edge-length resolution metric $\mathcal{M}_h$ developed as a function of (a). (c) Surface geoid boundary representation Γ' in blue, overlaid with multi-scale spatial discretisation $\mathcal{T}_h$.

Validation of the mesh generation process is achieved in four ways. Firstly, with reference to the formal grammar of the constraint space, a degree of self-validation can take place on-the-fly as mesh options are built up. Following rules described in the schema, only some options are available and certain combinations permitted. Unlike with namelist descriptions, or ad hoc collections of data, the user does not need to wait until running Shingle before receiving feedback on option validity. Available options are limited dynamically following the constraints and option selections. Moreover, with information from the schema on the mesh generation problem, it is possible to identify which options are required for the problem to be complete. The creation of a new BRML file immediately requires a name, type and options to be completed for at least one geoid surface representation and a geoid metric. The GUI highlights which required options remain to be completed (see figure 9.2). This is particularly useful to users new to mesh generation.

Secondly, the required 'type' option classifies the mesh and checks at runtime it is suitable for the intended simulation. A 'shallow water' model requires only a surface geoid discretisation $\mathcal{T}_h$ for example, whilst a full three-dimensional mesh is needed in other simulation types. This is a sanity check to ensure the mesh generation problem is fully constrained for the intended purpose, beyond the fundamental constraints 1 – 5.

Thirdly, a parsing stage following application of a meshing algorithm eliminates commonly found issues in output mesh descriptions, ensuring structural integrity. For example, additional lone, unconnected boundary

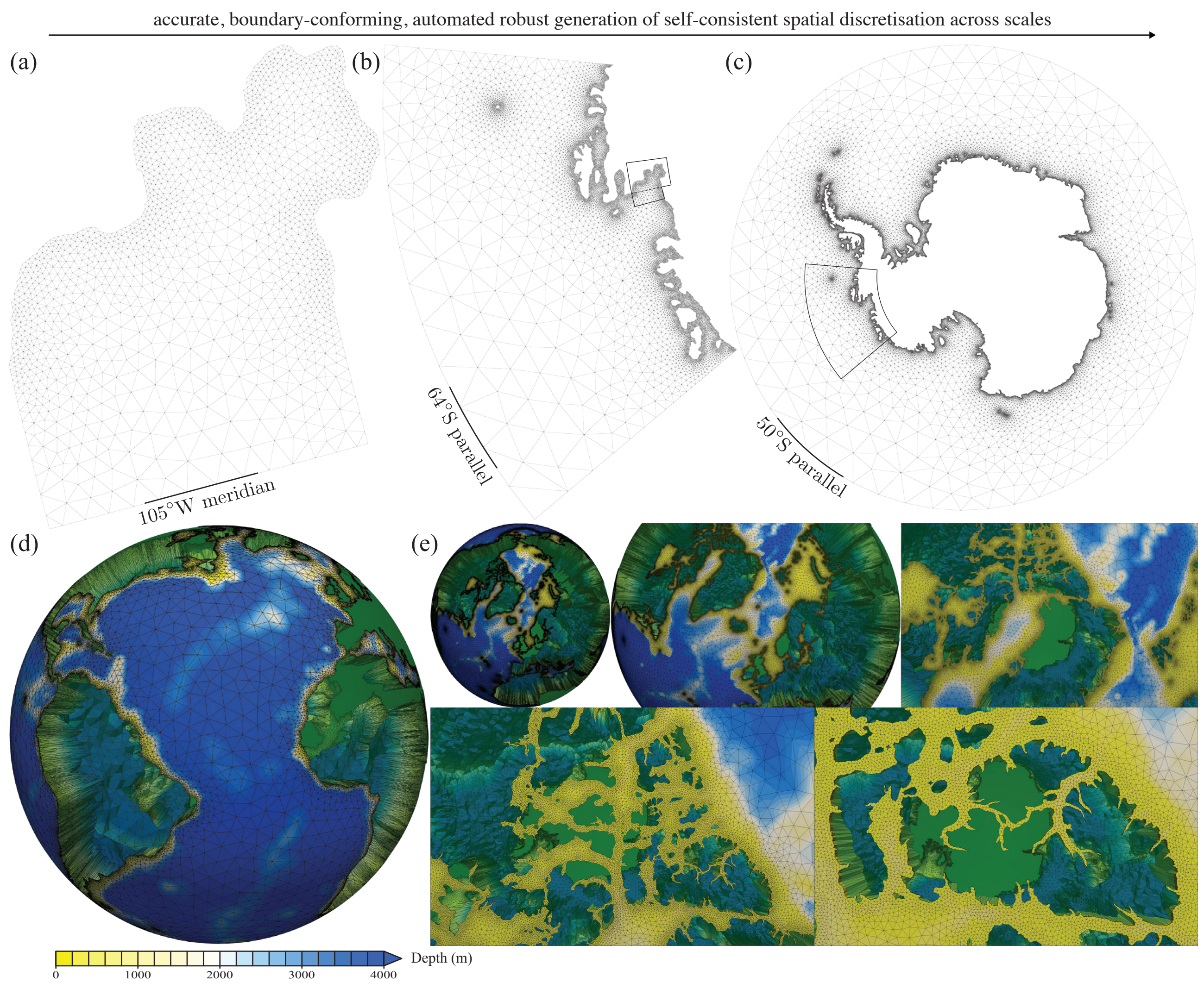

accurate, boundary-conforming, automated robust generation of self-consistent spatial discretisation across scales
(a)
(b)
(c)
105°W meridian
64°S parallel
50°S parallel
(d)
(e)
0
1000
2000
3000
4000
Depth (m)

Figure 13.2: A selection of further example geophysical domain discretisations straight-forwardly described in BRML and automatically constructed using Shingle. (a) $\mathcal{T}_h$ of the Pine Island Glacier ice shelf ocean cavity from ice-bedrock grounding line extended out to the 105°W meridian. (b) The Amundsen Sea region in West Antarctica extended out to the 64°S parallel. (c) The Southern Ocean Antarctic continent landmasses, from ice grounding line to 50°S parallel, built from a high fidelity boundary representation containing 348 automatically identified islands. (d) The full $\mathcal{T}$ of the global oceans, with a radial scaling of 300 to exaggerate the vertical extent of the discretised shell and land regions shaded green. (e) Zoomed in regions focusing on the complex Canadian Arctic Archipelago west of Greenland around Ellesmere and Baffin island. (a)–(c) are generated from the GEBCO [2014] DEM and presented under a orthographic projection centred on 90°S, and (d)–(e) from RTopo [Timmermann et al., 2010] and viewed in a Cartesian frame. These contain a multi-scale of spatial resolutions, with element edge-lengths parallel to the geoid in these examples, specified through $\mathcal{M}_h$, ranging from 2km to 500km. Vertical layers in (d), specified through $\mathcal{M}_v$, vary from 2m to 500m, under differing regimes in a generalised hybrid coordinate system described further in Candy [2017], and leads to a mesh containing 8,778,728 elements and 35,114,912 spatial degrees of freedom under its discontinuous Galerkin finite element discretisation. Along with other examples presented in figure 1.1(c)-(g), these are part of the test suite accompanying the library.

elements are removed in this step to ensure the discretised output mesh is as expected. Meshing algorithms do not usually possess information on underlying numerical discretisations, and it is also possible elements are generated that 'tied' to boundary conditions, with no independent free unknowns. This type of problem in the spatial discretisation is often difficult to identify, only being picked up at runtime, or through careful visual inspection. This parsing is an opportunity to identify and process these at this stage. Numerical simulation codes are sometimes accompanied with standalone mesh checking tools to support initialisation stages (e.g. the MechChecker.F90 utility for the ADCIRC model), and visual interfaces can be used for manual inspection and editing, such as the Show Me tool provided alongside Triangle [Shewchuk, 2002] and the GUI of Gmsh [Geuzaine and Remacle, 2009]. This is part of the mesh generation process and, if possible, better handled

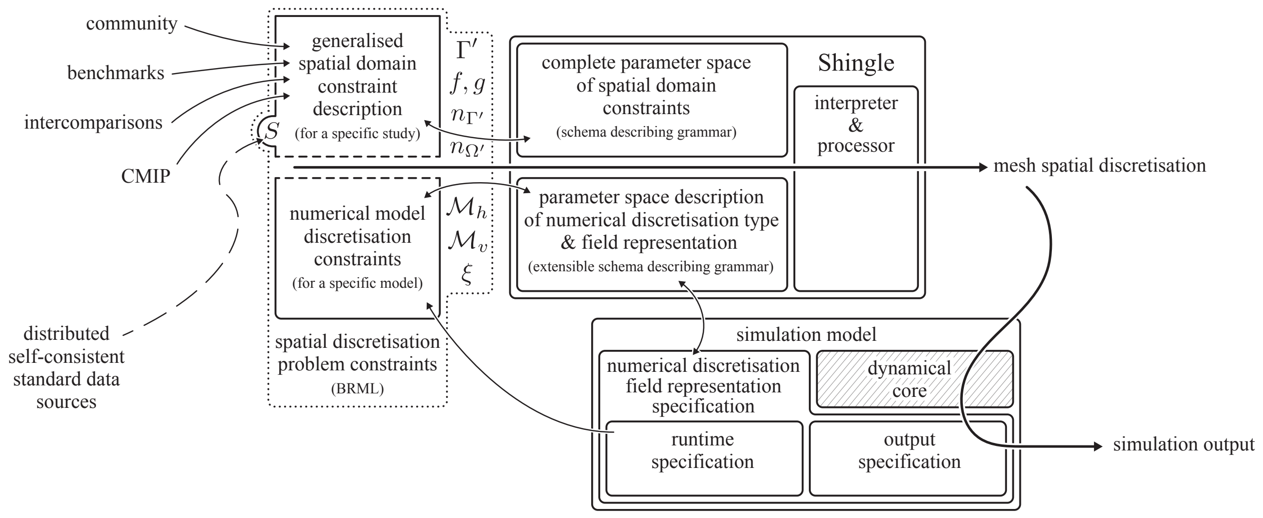

community
benchmarks
intercomparisons
CMIP
distributed self-consistent standard data sources
generalised spatial domain constraint description (for a specific study)
S
Γ'
f, g
$n_{\Gamma'}$
$n_{\Omega'}$
numerical model discretisation constraints (for a specific model)
$\mathcal{M}_h$
$\mathcal{M}_v$
ξ
spatial discretisation problem constraints (BRML)
complete parameter space of spatial domain constraints (schema describing grammar)
Shingle
interpreter & processor
mesh spatial discretisation
parameter space description of numerical discretisation type & field representation (extensible schema describing grammar)
simulation model
numerical discretisation field representation specification
dynamical core
runtime specification
output specification
simulation output

Figure 13.3: Framework for generalised spatial domain discretisation for geophysical model simulations. A formal spatial domain constraint description (a model-independent grouping of high-level directives describing key geospatial boundaries and features, required spatial resolution and source datasets) for a specific study (e.g. the geography to include in a CMIP intercomparison study) is joined with specific constraints from a simulation model, depending on its internal numerical discretisations and field representations (e.g. following Gridspec [Balaji et al., 2007], or a UFL description [Alnæs et al., 2014]). These constraints are used by the interpreter Shingle to produce, in a robust, automated, repeatable process, a model-specific mesh spatial discretisation. Moreover, the latter description is further used to specify numerical simulation output representation (as CMIP uses Gridspec).

automatically following tenet 7, as proposed.

Lastly, the fourth approach to validation is through explicitly defined expected boundary representation and discretised mesh characteristics. Like the initial consistent approach of Candy [2017], the intermediate high fidelity boundary representation is compared at a raw level. Being a deterministic process, deviations are only expected as a result of depending on Shingle library version and behaviour, source data and potential OPeNDAP response, machine precision and the originating BRML description. At this stage of the meshing process, this has been supplemented with a test on the area within the bounds of the high fidelity geoid boundary representation Γ'.

On the discretised output, the tests include simple lower and upper bounds on output geoid mesh node and element numbers, the number of boundary elements, and element circumspheres to check adherence to metric constraints. The degree of representation is examined comparing the high fidelity geoid boundary representation surface area to its corresponding discretised form. Boundary complexity is measured through the overall Minkowski fractal dimension.

This provides a means for users to easily specify what should be expected in the discretised output, to ensure the accuracy required in tenets 1 – 3. Testing built in to the mesh generation process, further automates the process. It is also important to ensure tenet 8: provenance, that the solution mesh is the same (within prescribed tolerances) as that that has been generated in the past, and potentially by others on different systems.

A self-validating description provides tenet 9: a standardisation of interaction with the descriptions themselves. Users can immediately begin building on and improving the work shared by others, having been able to check the descriptions give a solution expected by the creator. This eliminates ad hoc or purely qualitative measures of conformity and reinforces the provenance record of the mesh generation process.

This is important when these then form key components of critical studies, such as the coupled climate and Earth system models run for internationally coordinated model intercomparisons, such as CMIP and CORE [Griffies et al., 2014; Meehl et al., 2007; Taylor et al., 2012], that form the basis of reports compiled by the IPCC.

Models containing unstructured meshes with conforming boundaries are now starting to be used in such large-scale international research efforts [e.g. FESOM, Sidorenko et al., 2014]. This approach provides the full provenance, reproducibility and complete constraining descriptions of the significantly more complex spatial discretisations supported by these models.

13.2 Continuous verification

To ensure Shingle as a whole continues to behave as expected for all users and on all systems, it contains a verification test engine. This processes a suite of key meshing problems, which are then automatically evaluated following the validation tests defined in their BRML description. Since the BRML descriptions are self-validating, the addition of new tests to the suite is simply a matter of adding the problem description file to a test folder of the source code. Testing is often a secondary consideration to new feature implementation, so it is important the extension of testing suite is as simple as possible.

This can simply be run at the time of a new installation, following the upgrade of required libraries or the operating system, or routinely as part of a commit-hook buildbot with dedicated resources to continuously verify new code pushed to a Shingle development code repository [see, for example, Farrell et al., 2011]. Being built on standard libraries, it could further form part of an automated wider system framework validation, for the above climate intercomparison projects, for example, reproducing the entire process from initialisation to post-processing, on demand. Alternatively, the engine can be used to drive an efficient drafting and prototyping workflow (tenet 5) with updates to mesh generation problems automatically processed and tested, to support an iterative domain discretisation process.

The Shingle project is continuously verified on local systems and additionally online, where new code pushed to `https://github.com/shingleproject/Shingle` is automatically tested on Travis (`https://travis-ci.com`) with reports available on `https://travis-ci.org/adamcandy/Shingle`. An indication of the status of this continuous verification build is additionally presented on the project and Github pages.

Chapter 14

Summary

This library is a high-level abstraction to mesh generation for domains containing complex, fractal-like coastlines that characterise those in numerical simulations of geophysical dynamics, together with a compact, shareable and necessarily complete description of the domain discretisation.

The approach is designed to be accessible to a wide range of users and applications. This begins at a simple standalone GUI-driven one way workflow, where users are guided through the option parameters required to constrain the domain discretisation problem. Options are presented in context through the hierarchical tree structure with documentation automatically provided alongside. Moreover, the use of a human readable XML format and introduction of high-level natural language based geographical objects give BRML problem constraint descriptions that closely follow those presented in literature and shared by scientists. The example built up from the description ($*$), to BRML in figures 9.2 and 9.3, followed by the construction of the high fidelity boundary representation and resulting spatial discretisation shown in figure 11.2, highlights how the problem of generating a domain bounded by a complex coastline defined by a depth contour and three orthodromes, common in tsunami modelling studies, is trivially constructed and solved using Shingle.

This is easily built on and extended to larger and more complex problems. High-level objects automate processing of multiple, potentially complex geospatial features. BRML descriptions are easily shared and XML sections cut and pasted to combine descriptions and build up complexity. New high-level objects and processing can be prototyped directly in Python to later join the core LibShingle operations library. Corresponding natural language based objects are available through the Python API, meaning domain discretisation can be achieved directly and purely in native Python code, for complex setups, direct integration with numerical simulation codes, or interactive sessions or Jupyter notebooks[1]. Both the BRML file descriptor and modular LibShingle are extensible.

Extending the tsunami example shown in figure 11.2, this robust and automated approach could form part of a real time warning system using unstructured spatial discretisation, with a domain created on-the-fly, centred around the earthquake epicentre, in a direct response to measurement by GPS seismic monitors.

Recognising the domain discretisation process is becoming more challenging and more difficult to document completely, such that others can reproduce, has been central to steering this approach. Progress is focused on the nine tenets of geophysical mesh generation summarised in table 7.1. One result of this is that Shingle treats the mesh generation as a model problem. Strategies from numerical simulation model development have been adopted and modified to formalise the description of the heterogeneous geophysical mesh generation constraints, such that they provide an accurate and complete description (tenets 1 – 3) in a standardised language-based XML form (tenet 9). This compact text-based description easily affords a record of changes in the development of a domain discretisation (tenet 8) and through the BRML grammar ensures it is always a complete description and therefore reproducible. The model-based approach manages the range types of parameters (which have diversified with the use of flexible unstructured discretisations) and supports users in their preparation, to allow for efficient drafting and prototyping (tenet 5). With options managed in a structured hierarchical tree, complex discretisations can be built up logically (tenet 7) and scaled up (tenet 6).

The creation of the BRML file boundary representation description is not intended to reinvent standards. It is not a new data descriptor, for orientated vector paths or two-dimensional raster data, for example. There exist standards already that tackle these challenges well. It is rather a new problem descriptor, like those for Fluidity [Piggott et al., 2008] and the TerraFERMA model of Wilson et al. [2016], for fully describing the mesh generation problem specifically for geophysical model domains, following the approach that this requires solving the same types of challenges involved in numerical model setup, that makes significant progress in

[1] http://jupyter.org

meeting the tenets of table 7.1.

The consistent approach of Candy [2017] is adopted, with an emphasis on producing a self-consistent high fidelity description and resulting output domain discretisation. Consistency is additionally encouraged through a centralised definition of the source data and processing in the BRML description (see figures 9.1–9.3). Use of decentralised, distributed datasets, efficiently accessed using OPeNDAP, ensures the discretisation uses exactly the same source data on every processing instance.

Verification and discretisation validation is achieved at multiple points throughout the process. The formal grammar of the BRML, imposed by the schema, enforces valid inputs and provides initial option checking. This framework and interaction with the schema using the libspud library additionally enables new self-validating user interfaces to be written. With expected mesh validation measures included in the BRML descriptions, discretisations are automatically validated and continuous verification of the library is easily obtained.

With the dependable, robustly verified library LibShingle for high-level abstractions for geophysical mesh generation, it is easily applied to develop interactions with other frameworks and models, such as GIS, as described in Candy et al. [2014]. Critically, with the standalone LibShingle library, these are easier to maintain and better insulated to API changes in other codes.

It does not immediately solve the mesh generation constraint problem in general, since numerical simulation models use a wide range of mesh types and numerical discretisations. It has however, been designed with this in mind, with low-level structures that are extensible, to accommodate additional mesh types for example, and high-level constructs that are applicable to all geophysical models. Arguably the *'holy grail'* of domain initialisation for geophysical models, characterised by the constraints 1–5 following the development figure 9.1, is a grouping of high-level directives describing bounds (including key geospatial features to capture), required spatial resolution and source datasets that can be interpreted by any model, each dealing with the discretisation depending on the field representations within the model (figure 13.3). Shingle provides an extensible platform to achieve this, focusing on general, natural language based, model-independent descriptions of domain descriptions, that can be shared and used for different models. LibShingle additionally provides a means to interpret these descriptions such that this part of the process can be included in numerical simulation code, with the BRML constraints supplemented by those imposed by the simulation model, such as specific numerical discretisation choice (e.g. to use hexagonal over triangular prism elements), or ensuring a minimum degree of representation in maintained between bounds (e.g. within narrow river channel networks).

Chapter 15

Code availability, distribution and licensing

The Shingle computational research software library, developed as part of this study, is available at `https://github.com/shingleproject/Shingle`, with further details presented at `https://www.shingleproject.org`. This includes a suite of example domain discretisation BRML descriptions and the verification test engine presented in chapter 13.

All components of the Shingle package which have been under continued development since 2011 are free software, being released under the GNU Lesser General Public License version 3.0. Full details of the license, including the compatible copyright notices of third party routines included in the package, are included in COPYING in the source distribution.

Chapter 16

Appendix

Chapter 17

Verification test suite spatial discretisation solutions

The verification test engine can automatically generate an image of each test. Where provided, this follows the view orientation described in the option path `/output/orientation`. These are useful for quick visual inspection, in addition to the formal verification metrics examined as part of each test.

These images are collected by a Python code which links directly to the Shingle library to read each of the test BRML files to locate the output mesh images. This additionally reads the spatial discretisation problem name and comments.

This itself is an example application of the Shingle library to interrogate spatial discretisation problem descriptions.

The example outputs and their descriptions are presented below.
Note the mesh images for all active tests can be generated by:

```
make testimage
```

in the route source folder, or using the Shingle executable:

```
shingle -t path_to_test_folder -image
```

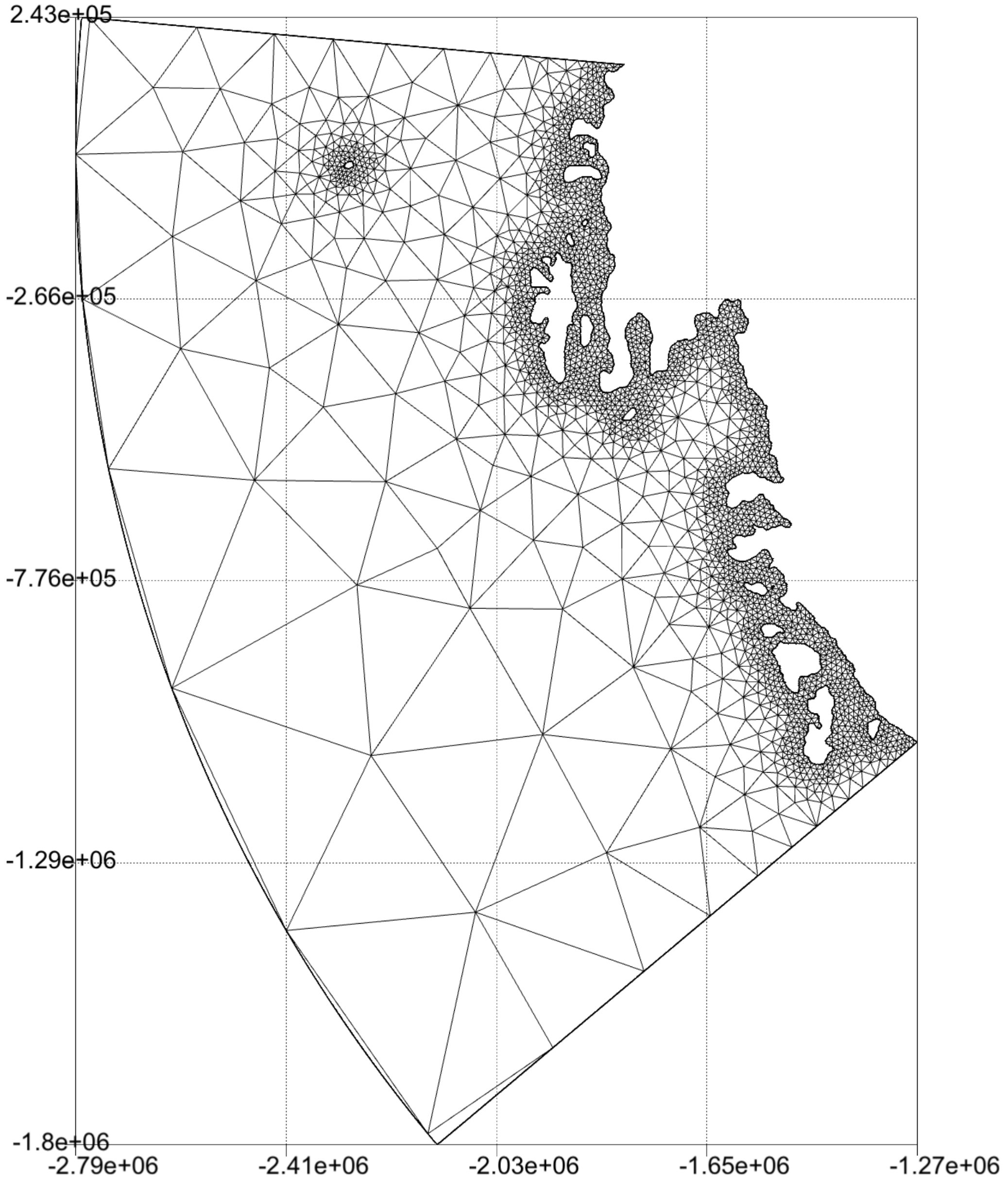


Figure 17.1: Amundsen Sea. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset, 50S version (RTopo105b 50S.nc), selecting the region [-130.0:-85.0,-85.0:-60.0] for the Amundsen Sea, extended up to the 64S parallel. Ice shelf ocean cavities are included.

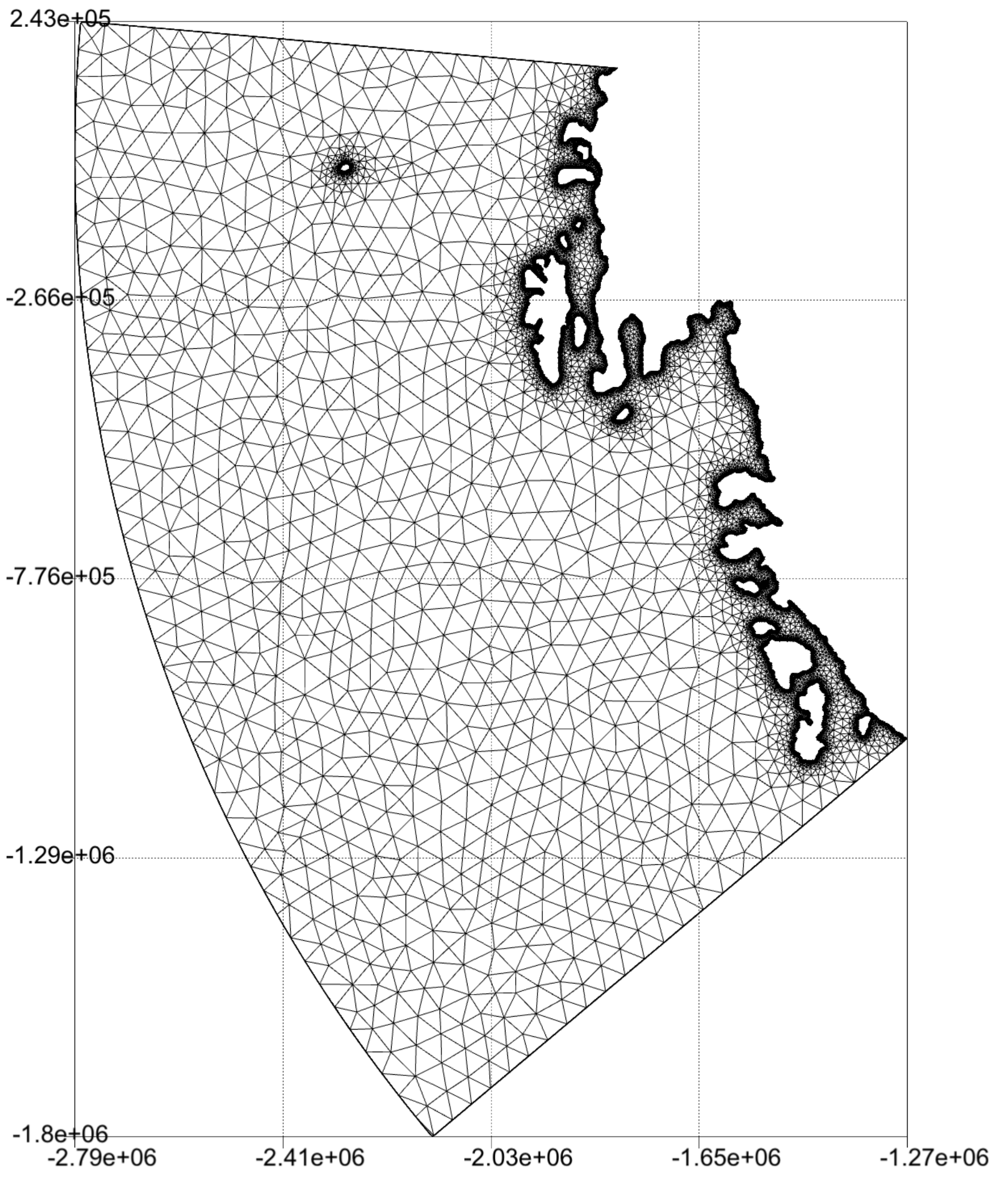


Figure 17.2: Amundsen Sea fine. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset, 50S version (RTopo105b 50S.nc), selecting the region [-130.0:-85.0,-85.0:-60.0] for the Amundsen Sea, extended up to the 64S parallel. Ice shelf ocean cavities are included.

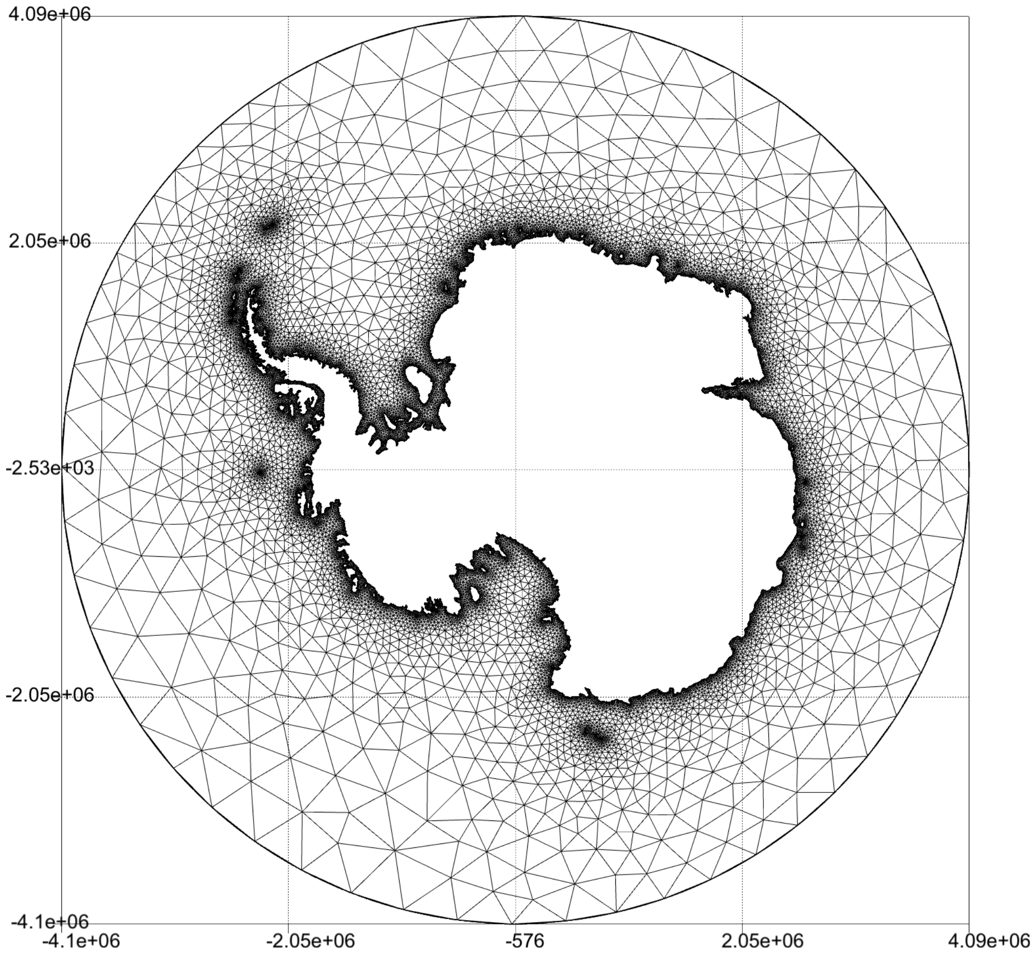


Figure 17.3: Antarctica all. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset (RTopo105b.nc), considering all land masses up to a latitude of 60S, extended up to the 50S parallel. Ice shelf ocean cavities are included. Spatial representation is specified along the open ocean boundary to ensure it is well-represented.

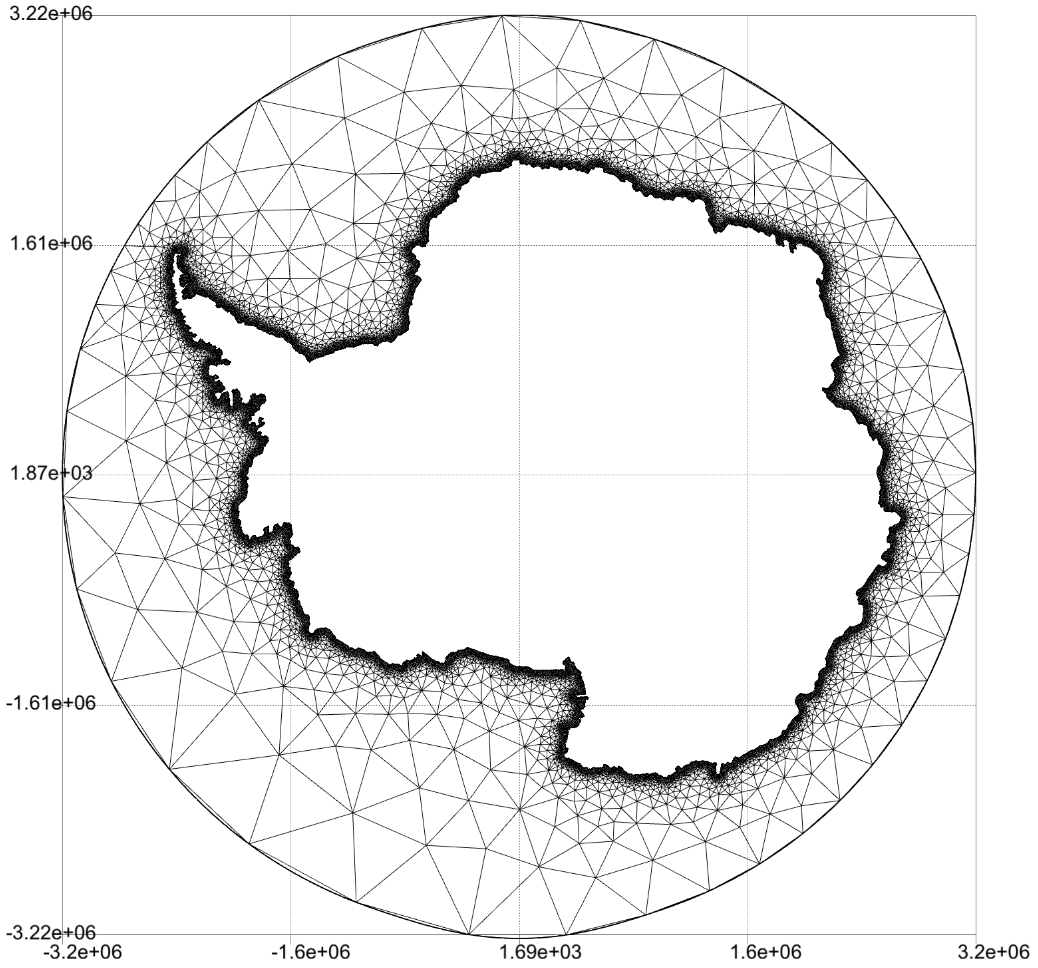


Figure 17.4: Antarctica main landicemass. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset (RTopo105b.nc), considering all land masses up to a latitude of 60S, extended up to the 60S parallel. Only the main largest landmass is included. Ice shelf ocean cavities are excluded. Spatial representation is specified along the open ocean boundary to ensure it is well-represented.

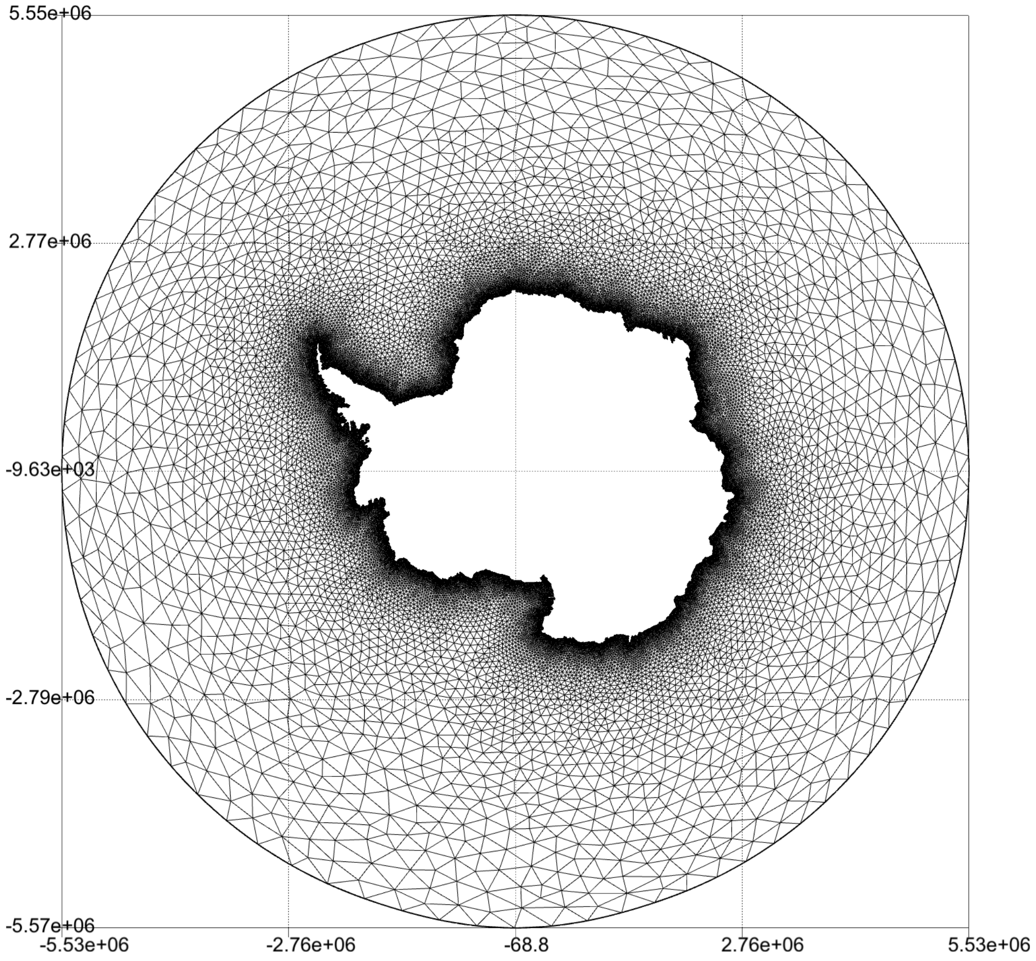


Figure 17.5: Antarctica main landicemass 30s. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset (RTopo105b.nc), considering all land masses up to a latitude of 60S, extended up to the 30S parallel. Only the main largest landmass is included. Ice shelf ocean cavities are excluded. Spatial representation is specified along the open ocean boundary to ensure it is well-represented.

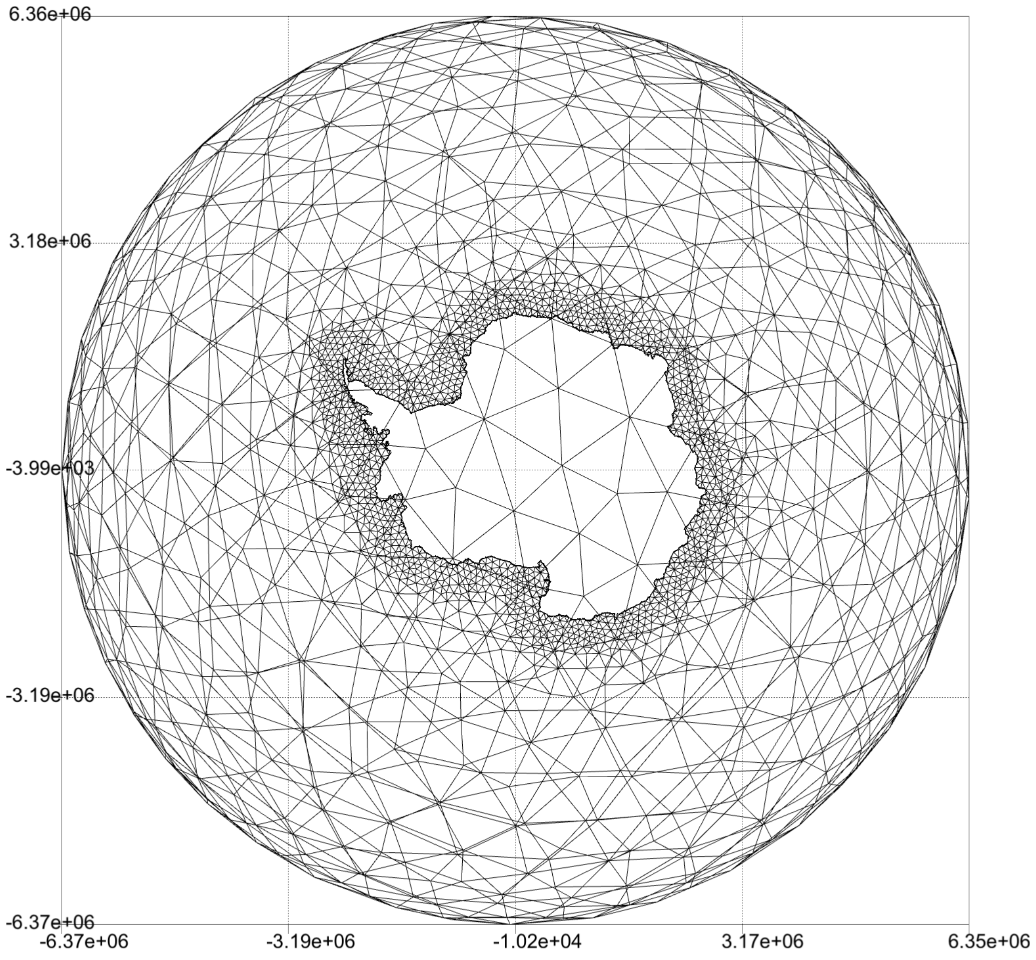


Figure 17.6: Antarctica main landicemass global. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset (RTopo105b.nc), considering all land masses up to a latitude of 60S, within a global domain. Only the main largest landmass is included. Ice shelf ocean cavities are excluded. Spatial representation is specified along the open ocean boundary to ensure it is well-represented. Minimum edge-length increased to 100km from 10km, minimum proximity to 400km from 40km, to reduce time for discretisation in verification test.

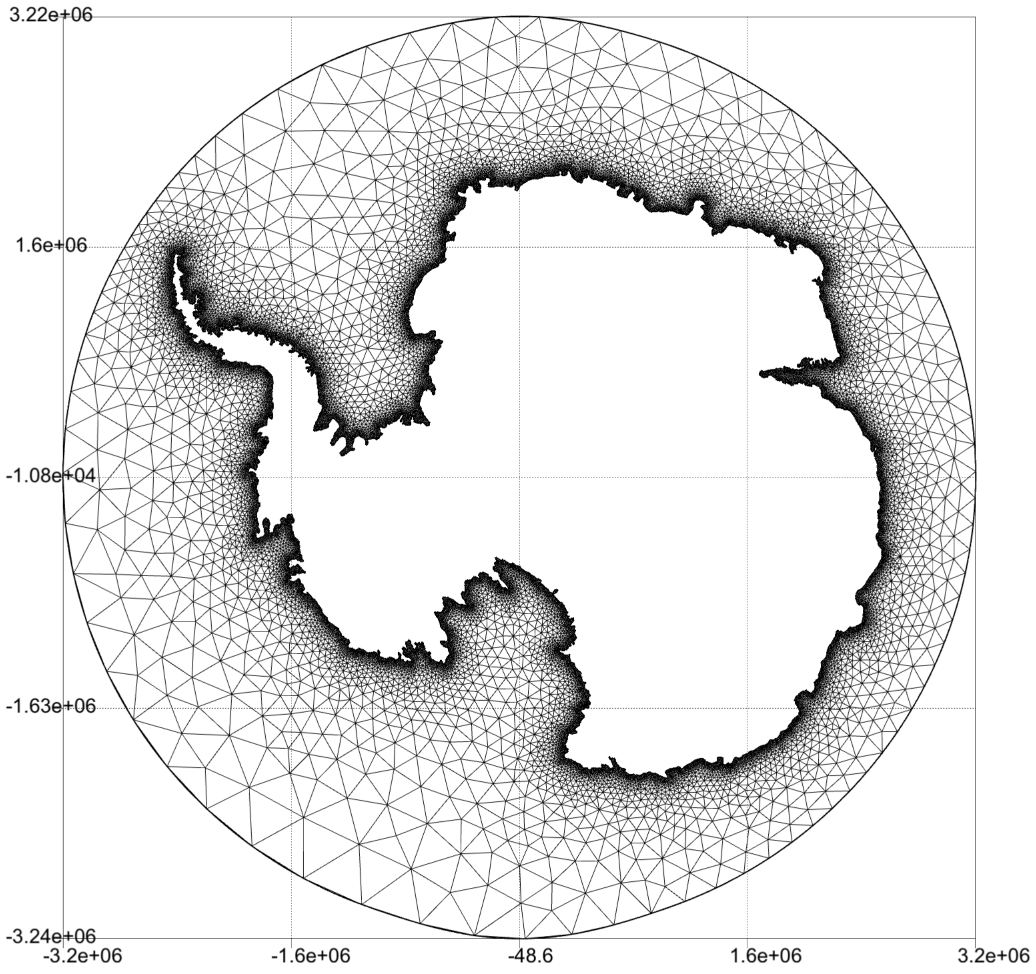


Figure 17.7: Antarctica main landmass. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset (RTopo105b.nc), considering all land masses up to a latitude of 60S, extended up to the 60S parallel. Only the main largest landmass is included. Ice shelf ocean cavities are included. Spatial representation is specified along the open ocean boundary to ensure it is well-represented.

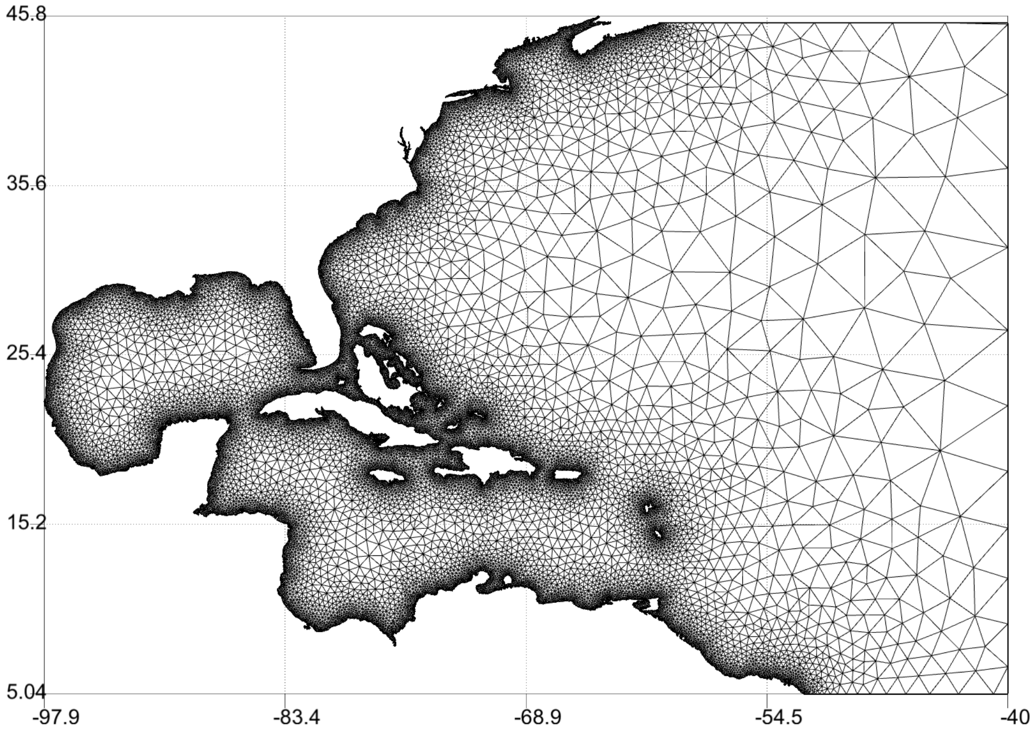


Figure 17.8: Caribbean. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Domain of the Caribbean Sea ocean basin, extended out to the 40W meridian.

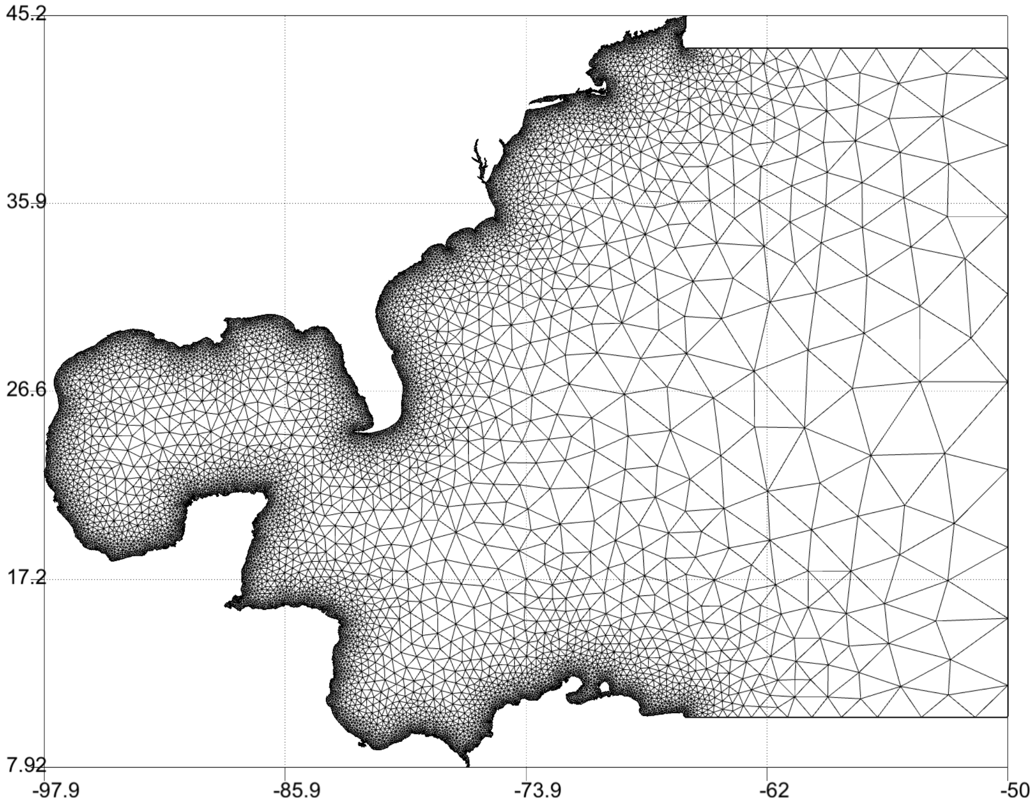


Figure 17.9: Caribbean main coastline. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Domain of the Caribbean Sea ocean basin, extended out to the 40W meridian. Developed from GEBCO. Contains only the main largest coastline.

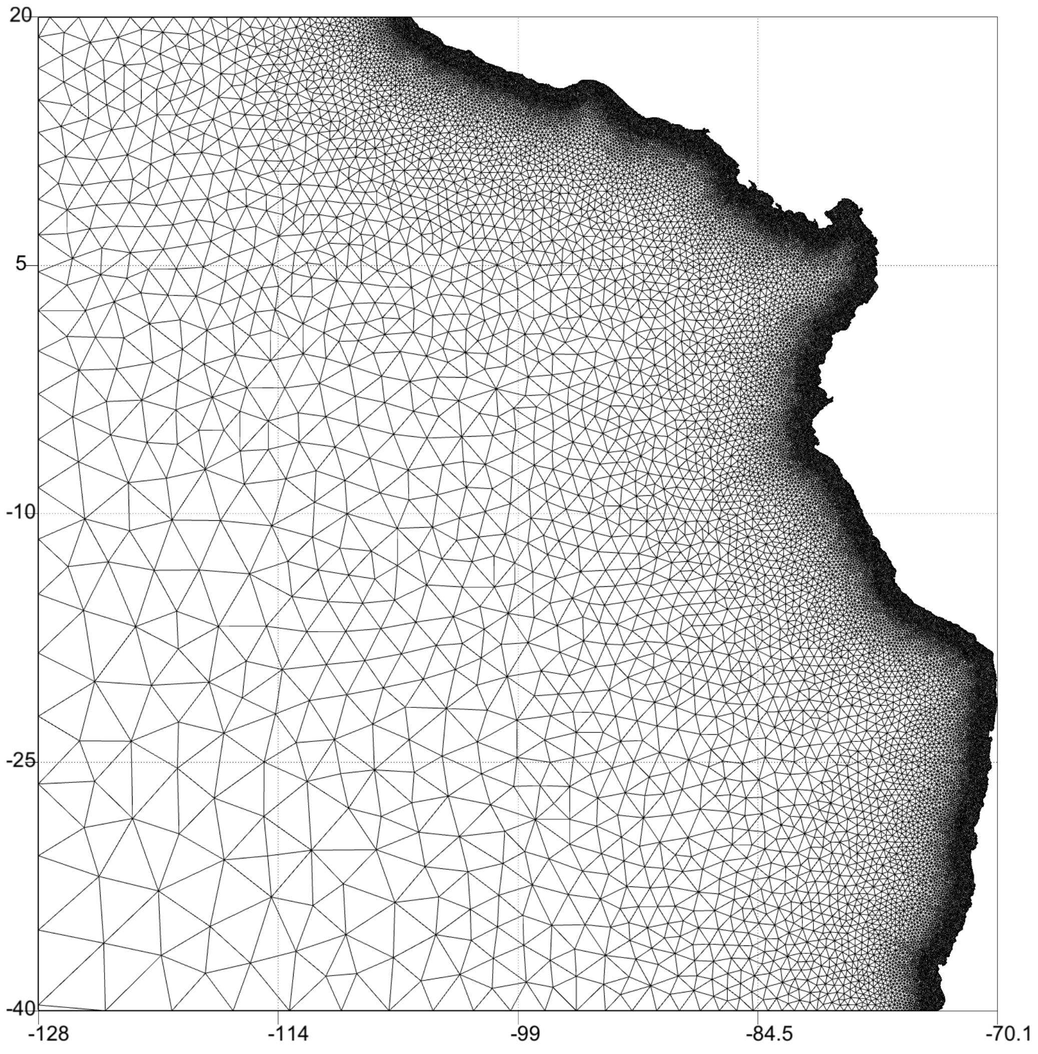


Figure 17.10: Chile Pacific wide. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: South East Pacific ocean up to the Chilean coastline. Developed from GEBCO, on the subset region 128W to 70W, 40S to 20N. In a latitude-longitude WGS84 projection.

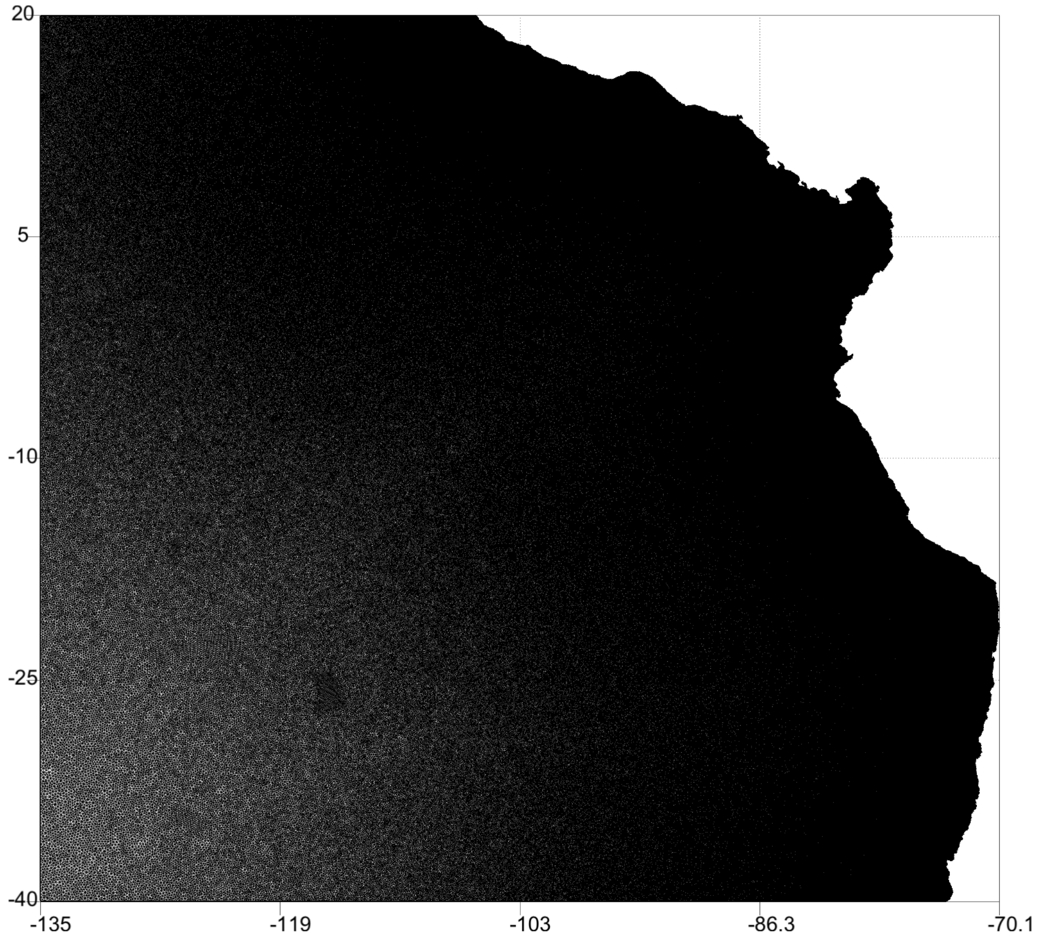


Figure 17.11: Chile Pacific wider. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: South East Pacific ocean up to the Chilean coastline. Developed from GEBCO, on the subset region 128W to 70W, 40S to 20N. In a latitude-longitude WGS84 projection.

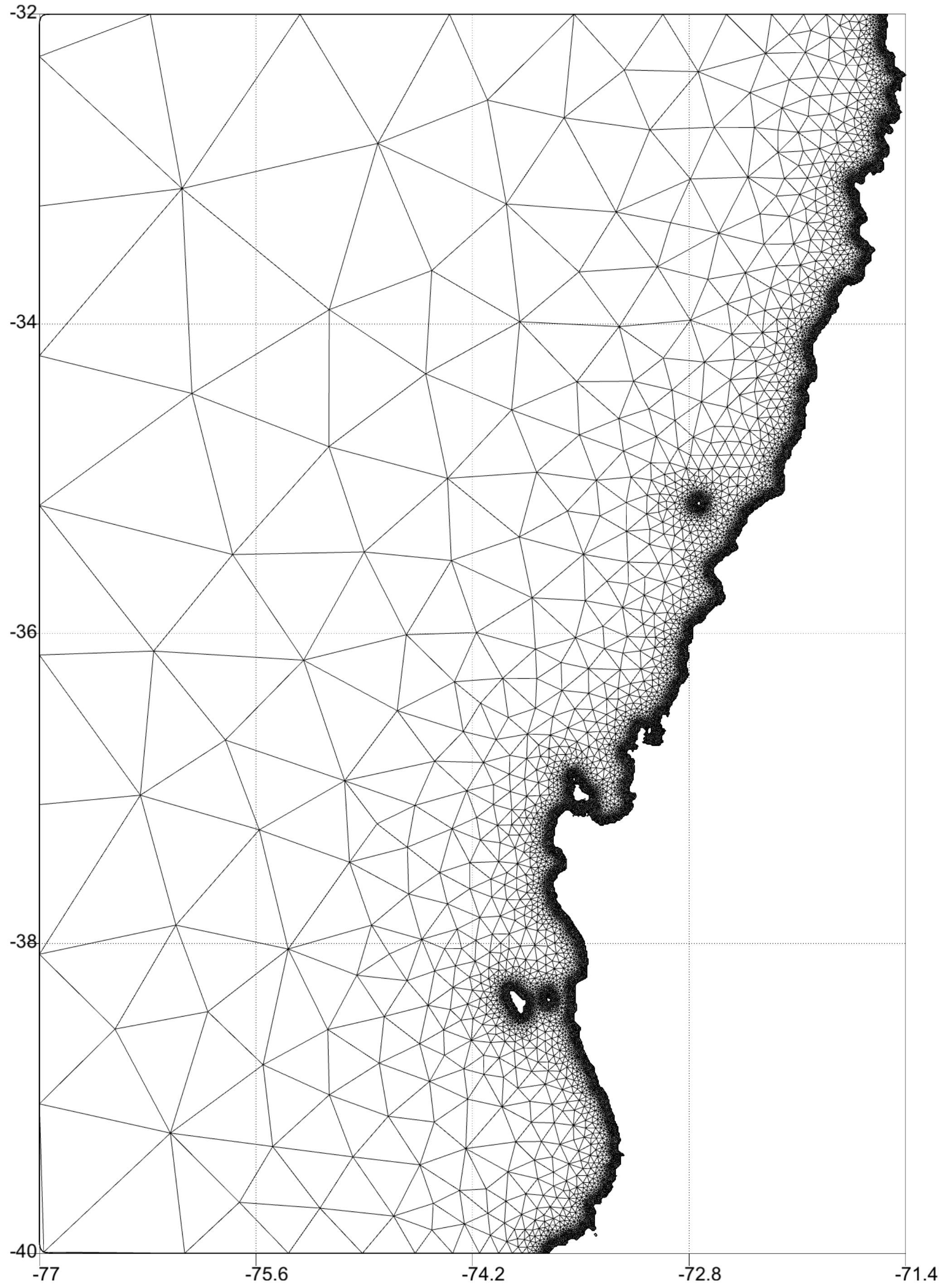


Figure 17.12: Chile Talcahuano. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Example simulation domain for modelling ocean wave propagation and tsunami inundation in the 2010 Chile M8.8 earthquake, centred at 35.9S 72.7W, approximately 100km north of Talcahuano. This is bounded by the 0m depth coastline from 32S to 40S, extended along parallels to the 77W meridian, in a latitude-longitude WGS84 projection.

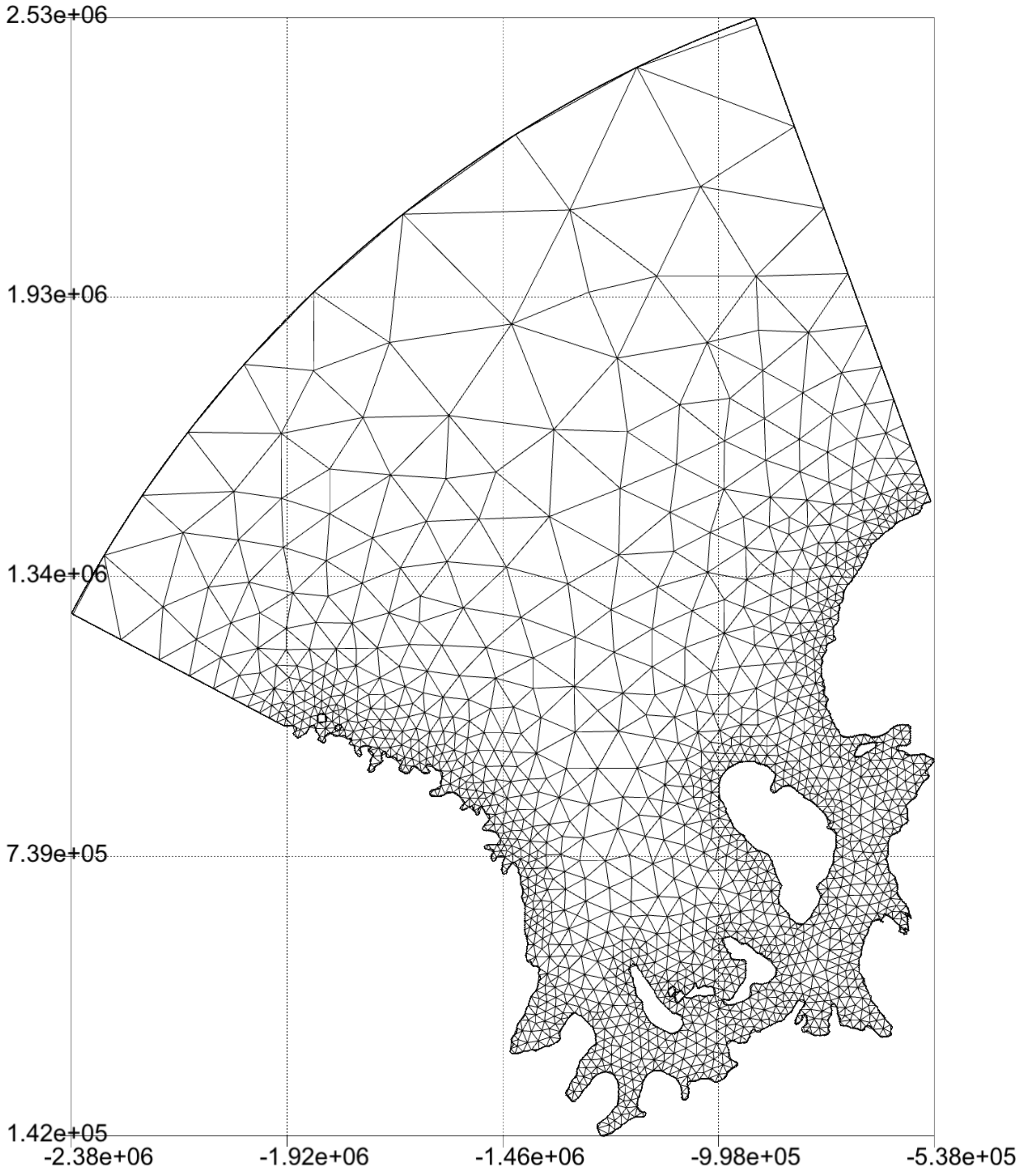


Figure 17.13: Filchner-Ronne. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset, 50S version (RTopo105b 50S.nc), selecting the Filchner-Ronne ice sheet ocean cavity region extended up to the 65S parallel. Ice shelf ocean cavities are included.

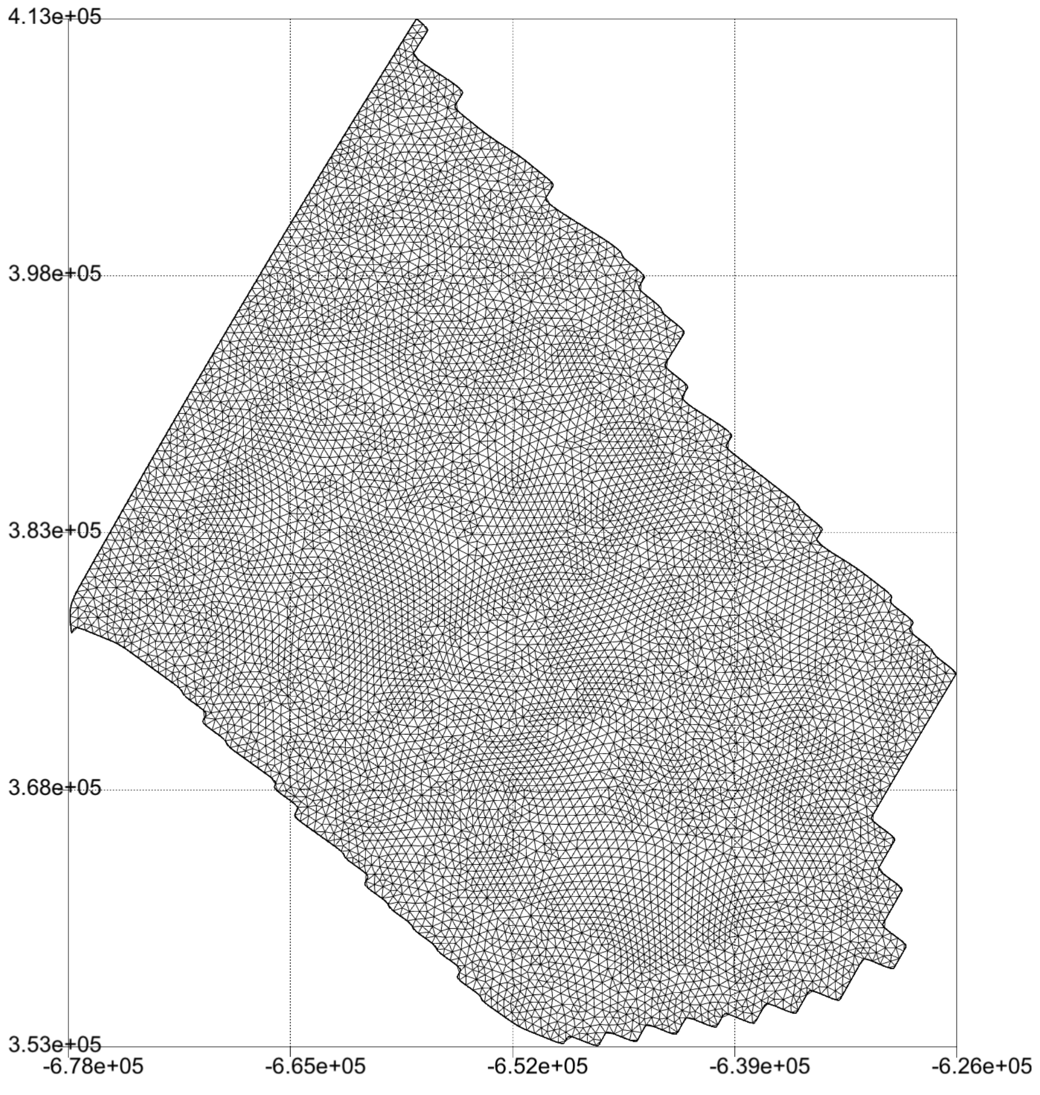


Figure 17.14: Filchner-Ronne back. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset, 50S version (RTopo105b 50S.nc), selecting the very back of the Filchner-Ronne ice sheet ocean cavity bounded by part of the grounding line, extended up to the 83S parallel. Ice shelf ocean cavities are included.

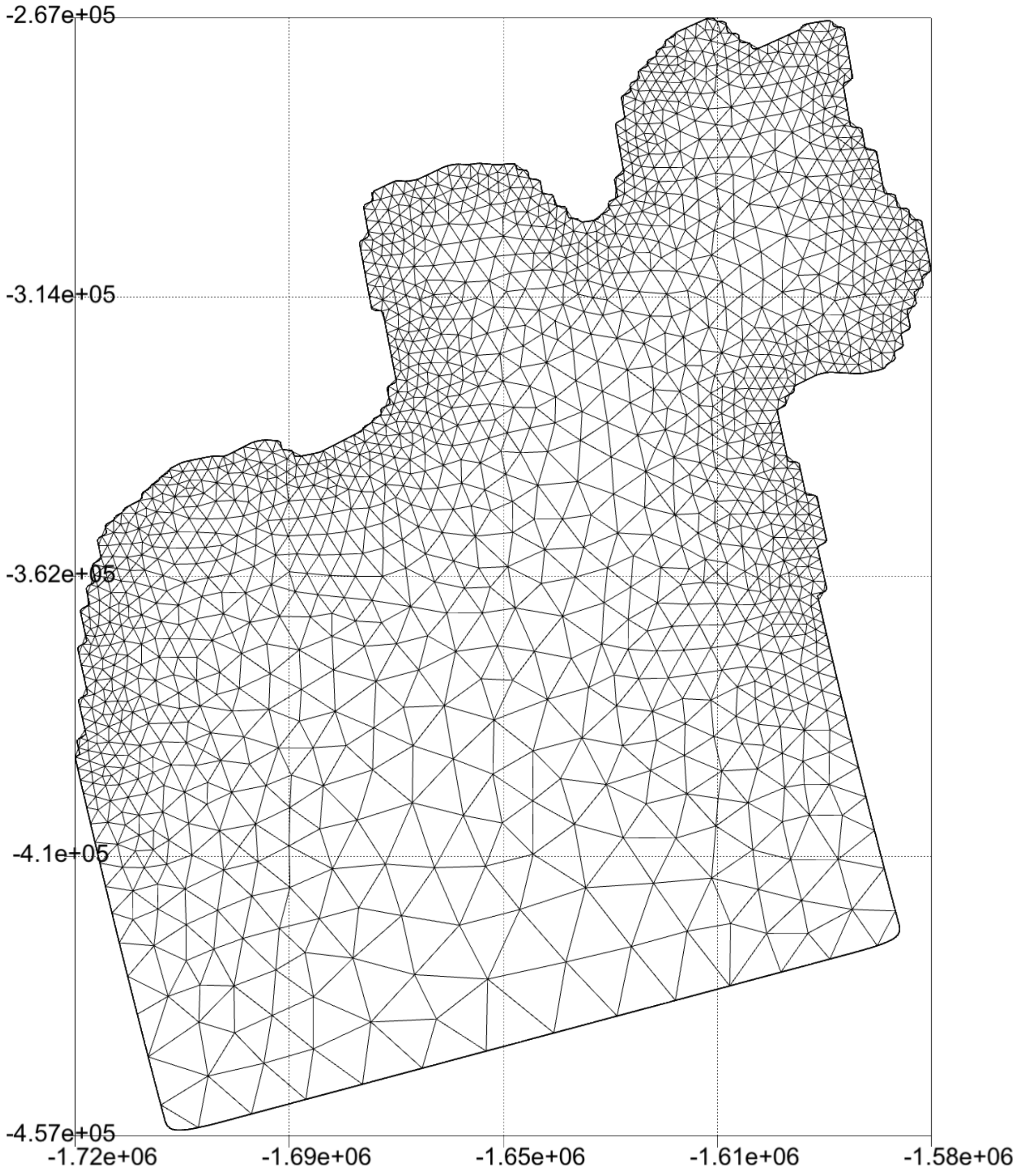


Figure 17.15: Pine Island Glacier. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: Use the RTopo dataset, 50S version (RTopo105b 50S.nc), selecting the region [-103:-99.0,-75.5:-73.9] for the Pine Island Glacier ice shelf ocean cavity, extended out to the 105W meridian. Ice shelf ocean cavities are included.

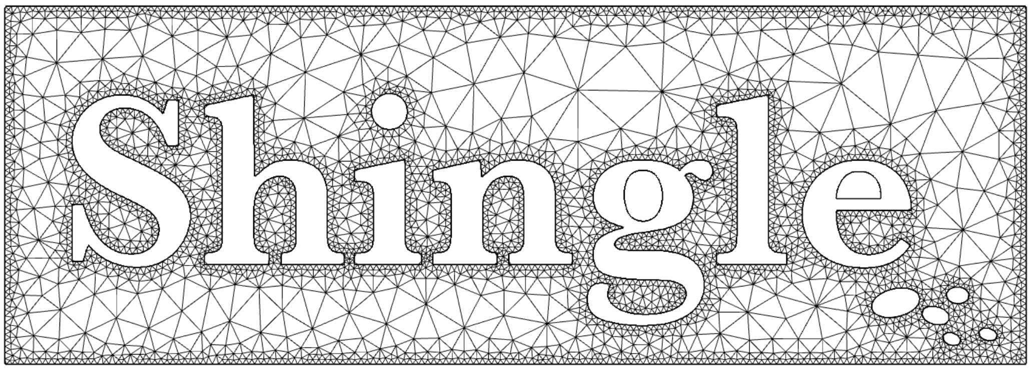


Figure 17.16: Shingle text. Image of output spatial discretisation automatically generated by the Shingle verification test engine. This test problem BRML description contains the following comment: A monochromatic raster image with the word 'Shingle' and five small islands developed as a mask.

References

M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, 40(2):9:1–9:37, Mar. 2014. ISSN 0098-3500. doi: 10.1145/2566630.

V. Balaji, A. Adcroft, and Z. Liang. Gridspec: A standard for the description of grids used in earth system models. In *Workshop on Community Standards for Unstructured Grids, Oct*, 2007.

A. S. Candy. A consistent approach to unstructured mesh generation for geophysical models. In review, 2017.

A. S. Candy and J. D. Pietrzak. Shingle 2.0: generalising self-consistent and automated domain discretisation for multi-scale geophysical models. Submitted, 2017.

A. S. Candy, A. Avdis, J. Hill, G. J. Gorman, and M. D. Piggott. Integration of Geographic Information System frameworks into domain discretisation and meshing processes for geophysical models. *Geoscientific Model Development Discussions*, 7:5993–6060, 2014. doi: 10.5194/gmdd-7-5993-2014.

E. Casarotti, M. Stupazzini, S. Lee, D. Komatitsch, A. Piersanti, and J. Tromp. GEOCUBIT, an HPC parallel mesher for spectral-element method seismic wave simulation. *SPE, EAGE*, 2008.

COMSOL. COMSOL Multiphysics. `http://www.comsol.com`, 2014.

P. Cornillon, J. Gallagher, and T. Sgouros. OPeNDAP: Accessing data in a distributed, heterogeneous environment. *Data Science Journal*, 2:164–174, 2003. doi: 10.2481/dsj.2.164.

P. E. Farrell, M. D. Piggott, G. J. Gorman, D. A. Ham, C. R. Wilson, and T. M. Bond. Automated continuous verification for numerical simulation. *Geoscientific Model Development*, 4(2):435–449, 2011. doi: 10.5194/gmd-4-435-2011.

GDAL. Geospatial data abstraction library. `http://www.gdal.org`, 2016.

GEBCO. General Bathymetric Chart of the Oceans (GEBCO). `http://www.gebco.net`, 2014.

C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, Sep 2009. doi: 10.1002/nme.2579.

J. M. Gregory. The CF Metadata Standard. 8, 4, 2003. URL `http://www.clivar.org/publications/exchanges/ex28/pdf/s28_gregory.pdf`.

S. M. Griffies, J. Yin, P. J. Durack, P. Goddard, S. C. Bates, E. Behrens, M. Bentsen, D. Bi, A. Biastoch, C. W. Böning, A. Bozec, E. Chassignet, G. Danabasoglu, S. Danilov, C. M. Domingues, H. Drange, R. Farneti, E. Fernandez, R. J. Greatbatch, D. M. Holland, M. Ilicak, W. G. Large, K. Lorbacher, J. Lu, S. J. Marsland, A. Mishra, A. G. Nurser, D. Salas y Mélia, J. B. Palter, B. L. Samuels, J. Schröter, F. U. Schwarzkopf, D. Sidorenko, A. M. Treguier, Y. Tseng, H. Tsujino, P. Uotila, S. Valcke, A. Voldoire, Q. Wang, M. Winton, and X. Zhang. An assessment of global and regional sea level for years 19932007 in a suite of interannual CORE-II simulations. *Ocean Modelling*, 78:35 – 89, 2014. doi: 10.1016/j.ocemod.2014.03.004.

D. A. Ham, P. E. Farrell, G. J. Gorman, J. R. Maddison, C. R. Wilson, S. C. Kramer, J. Shipton, G. S. Collins, C. J. Cotter, and M. D. Piggott. Spud 1.0: generalising and automating the user interfaces of scientific computer models. *Geosci. Model Dev.*, 2(1):3342, Mar 2009. ISSN 1991-9603. doi: 10.5194/gmd-2-33-2009.

S. C. Hankin, J. D. Blower, T. Carval, K. S. Casey, C. Donlon, O. Lauret, T. Loubrieu, A. Srinivasan, J. Trinanes, O. Godoy, et al. NetCDF-CF-OPeNDAP: Standards for ocean data interoperability and object lessons for community data standards processes. In *Proceedings of OceanObs: Sustained Ocean Observations and Information for Society*. European Space Agency, dec 2010. doi: 10.5270/oceanobs09.cwp.41.

J. Holt, P. Hyder, M. Ashworth, J. Harle, H. T. Hewitt, H. Liu, A. L. New, S. Pickles, A. Porter, E. Popova, J. I. Allen, J. Siddorn, and R. Wood. Prospects for improving the representation of coastal and shelf seas in global ocean models. *Geoscientific Model Development*, 10(1):499–523, feb 2017. doi: 10.5194/gmd-10-499-2017.

A. Humbert, T. Kleiner, C.-O. Mohrholz, C. Oelke, R. Greve, and M. A. Lange. A comparative modeling study of the Brunt Ice Shelf/Stancomb-Wills Ice Tongue system, East Antarctica. *Journal of Glaciology*, 55(189): 53–65, Feb 2009. doi: 10.3189/002214309788608949.

S. Kimura, A. S. Candy, P. R. Holland, M. D. Piggott, and A. Jenkins. Adaptation of an unstructured-mesh, finite-element ocean model to the simulation of ocean circulation beneath ice shelves. *Ocean Modelling*, 67: 39 – 51, 2013. ISSN 1463-5003. doi: 10.1016/j.ocemod.2013.03.004.

Q. Li, K. Ito, Z. Wu, C. S. Lowry, and S. P. Loheide II. COMSOL Multiphysics: A novel approach to ground water modeling. *Ground Water*, 47(4):480–487, 2009. ISSN 1745-6584. doi: 10.1111/j.1745-6584.2009.00584.x.

G. A. Meehl, C. Covey, K. E. Taylor, T. Delworth, R. J. Stouffer, M. Latif, B. McAvaney, and J. F. B. Mitchell. THE WCRP CMIP3 multimodel dataset: A new era in climate change research. *Bulletin of the American Meteorological Society*, 88(9):1383–1394, sep 2007. doi: 10.1175/bams-88-9-1383.

NWO Data Management Protocol. Netherlands organisation for scientific research (NWO): Data management protocol. `http://www.nwo.nl/en/policies/open+science/data+management`, 2014.

Ordnance Survey OpenData. Ordnance Survey OpenData. Available online: `https://www.ordnancesurvey.co.uk/opendatadownload`.

I. Pelupessy, B. van Werkhoven, A. van Elteren, J. Viebahn, A. Candy, S. Portegies Zwart, and H. Dijkstra. OMUSE: The oceanographic multipurpose software environment. *Geoscientific Model Development Discussions*, 2016:1–36, 2016. doi: 10.5194/gmd-2016-178.

M. D. Piggott, G. J. Gorman, C. C. Pain, P. A. Allison, A. S. Candy, B. T. Martin, and M. R. Wells. A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes. *International Journal for Numerical Methods in Fluids*, 56(8):1003–1015, Mar 2008. doi: 10.1002/fld.1663.

Quantum GIS Development Team. *Quantum GIS Geographic Information System*. Open Source Geospatial Foundation, 2016. URL `http://qgis.osgeo.org`.

F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. McRae, G. Bercea, G. R. Markall, and P. H. J. Kelly. Firedrake: automating the finite element method by composing abstractions. *CoRR*, abs/1501.01809, 2015. URL `http://arxiv.org/abs/1501.01809`.

R. Rew, G. Davis, S. Emmerson, H. Davies, E. Hartnett, D. Heimbigner, and W. Fisher. Network common data form (NetCDF). `http://www.unidata.ucar.edu/netcdf`.

T. Ringler, M. Petersen, R. L. Higdon, D. Jacobsen, P. W. Jones, and M. Maltrud. A multi-resolution approach to global ocean modeling. *Ocean Modelling*, 69:211–232, 2013. doi: 10.1016/j.ocemod.2013.04.010.

J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry*, 22 (1):21–74, 2002.

Shingle. Computational research software library for high-level abstraction to spatial discretisation for domains containing complex, fractal-like coastlines that characterise those in numerical simulations of geophysical dynamics. `http://shingleproject.org`, library repository at `http://github.com/adamcandy/Shingle`, 2011–2017.

D. Sidorenko, T. Rackow, T. Jung, T. Semmler, D. Barbi, S. Danilov, K. Dethloff, W. Dorn, K. Fieg, H. F. Goessling, D. Handorf, S. Harig, W. Hiller, S. Juricke, M. Losch, J. Schröter, D. V. Sein, and Q. Wang. Towards multi-resolution global climate modeling with ECHAM6–FESOM. Part I: Model formulation and mean climate. *Climate Dynamics*, 44(3-4):757–780, aug 2014. doi: 10.1007/s00382-014-2290-6.

K. E. Taylor, R. J. Stouffer, and G. A. Meehl. An overview of CMIP5 and the experiment design. 93:485–498, 2012. doi: 10.1175/BAMS-D-11-00094.1.

R. Timmermann, A. Le Brocq, T. Deen, E. Domack, P. Dutrieux, B. Galton-Fenzi, H. Hellmer, A. Humbert, D. Jansen, A. Jenkins, and et al. A consistent data set of Antarctic ice sheet topography, cavity geometry, and global bathymetry. *Earth System Science Data*, 2(2):261–273, Dec 2010. doi: 10.5194/essd-2-261-2010.

Triangle Library Python Bindings. `https://github.com/drufat/triangle`, 2014.

P. Wessel and W. H. F. Smith. A global, self-consistent, hierarchical, high-resolution shoreline database. *Journal of Geophysical Research*, 101:8741–8743, Apr. 1996. doi: 10.1029/96JB00104.

J. Westerink, R. Luettich, J. Feyen, J. Atkinson, C. Dawson, H. Roberts, M. Powell, J. Dunion, E. Kubatko, and H. Pourtaheri. A basin to channel scale unstructured grid hurricane storm surge model applied to Southern Louisiana. *Monthly Weather Review*, 136(3):833–864, 2008.

C. R. Wilson, M. Spiegelman, and P. E. van Keken. TerraFERMA: the Transparent Finite Element Rapid Model Assembler for multi-physics problems in Earth sciences. *Geochemistry, Geophysics, Geosystems*, 2016. *In review*.