AlgoMesh User Guide

AlgoMesh User Guide

Month of publication: March 2016

Table of Contents

1 Introduction

Welcome to the AlgoMesh user guide. This guide contains an overview of AlgoMesh and what it does, reference documentation on its features, and a comprehensive set of practical "how-to" tutorials to help you get up and running with the software.

This introduction section gives a brief background of AlgoMesh, introducing the mesh generation techniques it incorporates and some of the key features of its user interface.

If you would like to get started using the software immediately, take a look at the Getting Started section, which will point you to appropriate How To guides for learning the software, depending on the mesh generation or modelling task you would like to perform. Once you are confident with the basics, you may wish to review the Advanced Topics section, which details a number of tips and tricks for using AlgoMesh effectively, as well as some advanced features that are particularly effective in building certain types of models.

Specific reference material is provided in the User Interface and File Formats sections, which detail individual aspects of the AlgoMesh user interface and a number of file formats available for input and output in AlgoMesh, respectively.

1.1 Background

Finite-difference simulation code such as MODFLOW traditionally uses a structured, rectangular finite-difference grid to spatially discretise the model domain. With newer control-volume finite-difference (CVFD) formulations such as MODFLOW-USG (Panday et al., 2012), however, as well as with finite element simulation code, the use of prismatic unstructured grids is possible. In an unstructured grid – also called a *mesh* – model cells may be of different shapes, and of varying sizes and orientations. This gives several potential advantages over traditional structured grids.

First, cell geometry may be made to closely follow important geographical, geological, hydrological or man-made features, such as streams, lakes, wells, faults and mine boundaries. This enables a more accurate representation of the physical system being modelled, reducing one source of model errors and often mitigating convergence difficulties. Secondly, model resolution may be focused more heavily in areas that are identified as important in the conceptualisation phase, and reduced in other areas that are unlikely to significantly affect the model outputs. This has the effect of reducing errors due to discretisation (where resolution is increased), while at the same time allowing faster model run times by reducing resolution where it is not needed, or indeed even removing cells entirely from areas that are unneeded in the model. Finally, the flow faces between model cells may be aligned to more accurately represent directions of preferential flow, which can be beneficial when incorporating surface water components such as two-dimensional river flow in an integrated or coupled model.

Building an unstructured grid to make the most of these advantages is not trivial, however. A grid for a typical model may contain hundreds of thousands or millions of cells, and in general, producing such a grid manually is not practical. Algorithms and software tools are needed to automate the creation of grid cells that are sized and shaped appropriately to input constraints provided by the modeller. AlgoMesh provides such tools, greatly simplifying the building of unstructured grid models.

1.2 What is AlgoMesh?

AlgoMesh is a software tool providing implementations of two complementary algorithms for generation of unstructured triangular and Voronoi grids. It incorporates a comprehensive, easy-to-use graphical user interface (GUI) to assist in the model-building process.

AlgoMesh's primary focus as a model-building tool is producing input files for MODFLOW-USG simulations. However, its two-dimensional grid generation capabilities are quite general and it may be used as a powerful standalone mesh generator for any purpose. The following sections of this guide present a summary of the grid generation capabilities of AlgoMesh and its user interface functionality.

1.3 Grid Generation with AlgoMesh

AlgoMesh’s grid generation capabilities are based on two-dimensional Delaunay triangulations. A Delaunay triangulation comprises a set of triangles with the geometric property that no vertex of any triangle is contained inside the *circumcircle* of any other triangle (see Figure 1). This geometric property, applied to triangular grids, maximises the minimum angle of any triangle in the grid, in many cases avoiding poor-quality “narrow” triangles.

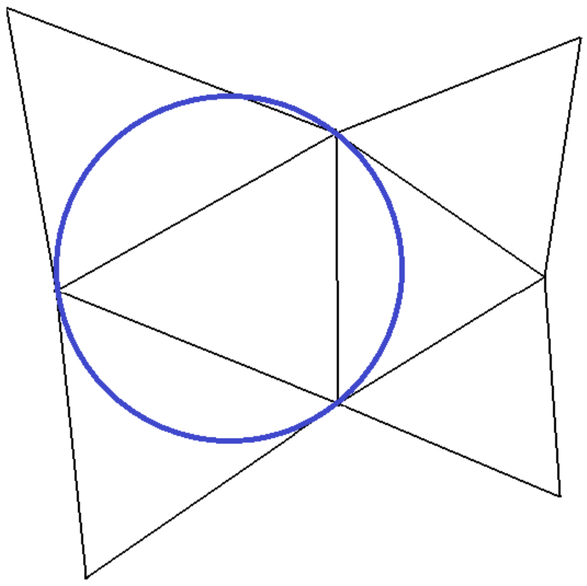


Figure 1: Part of a Delaunay triangulation showing one triangle's circumscribed circle (circumcircle), which is free of all other triangles' vertices.

For the cell-centred control-volume finite-difference approach used in MODFLOW-USG, we are particularly interested in Voronoi tessellations. These are a mathematical dual of Delaunay triangulations, in which the centre of each triangle’s circumcircle (the *circumcentre*) becomes a vertex, and adjacent triangles’ circumcentres are joined to create the edges of Voronoi polygons. Figure 2 shows an example of a Voronoi grid overlaid on its dual Delaunay triangulation.

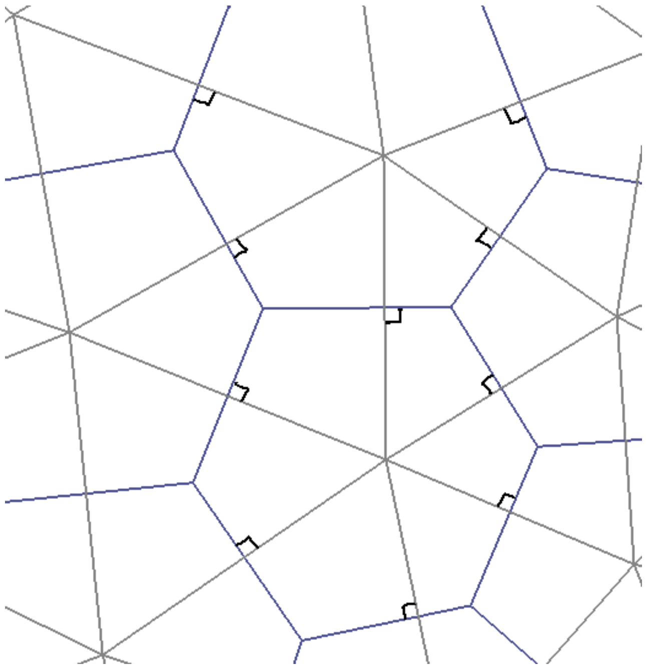


Figure 2: Part of a Delaunay triangulation and its dual Voronoi tessellation.

Voronoi grids are useful for building MODFLOW-USG models, as they guarantee one of the underlying geometric assumptions of the control-volume finite-difference formulation – that a line drawn between adjacent cell centres is perpendicular to the shared edge between them. The second geometric assumption of this formulation – that such a line precisely bisects the shared edge – is not guaranteed, but quality Voronoi grids often come very close to this, which helps to minimise errors arising due to the discretisation.

AlgoMesh provides implementations of two Delaunay-based triangular grid generation algorithms. Optionally, triangular grids that are produced by these methods are internally converted to their dual Voronoi representation. As a user of AlgoMesh, you only need to decide whether to work with triangular or Voronoi grids, and the necessary conversions will be performed fully automatically behind the scenes.

1.3.1 Delaunay Refinement

The first algorithm implemented in AlgoMesh is a so-called *Delaunay refinement* approach (Cheng et al., 2012). This type of approach has been used by many tools in the past to generate finite-element meshes – notably the popular mesh generator *Triangle* (Shewchuk, 1996), which has been integrated in several groundwater modelling user interfaces.

The overall process of producing a mesh by Delaunay refinement is illustrated in Figure 3. Delaunay refinement starts by producing a Constrained Delaunay Triangulation (CDT) of all input geometry that is given (points and line segments). A CDT is a triangulation that respects the Delaunay geometric property everywhere except for triangles adjacent to certain specified line segments (*edge constraints*), which must be present as edges in the triangulation. It then identifies and maintains a queue of “bad” triangles – where “bad” is defined by some geometric criteria such as having an angle below a desired minimum angle, having an area of edge length greater than some desired maximum, or being non-Delaunay due to an input edge constraint. These criteria are generally controllable by the modeler providing the input to the grid generation process.

Each bad triangle is “split” in turn by introducing a new point at an appropriate position within the triangle’s circumcircle, and adjusting the triangulation accordingly. Which point is chosen for insertion depends on the geometry of the triangle in question and its neighbors in the triangulation; the reader is referred to the literature for further detail and discussion (Cheng et al., 2012). When no more bad triangles are present, mesh generation is complete. Figure 3 illustrates the overall process, from the initial input geometry to producing a triangular mesh.

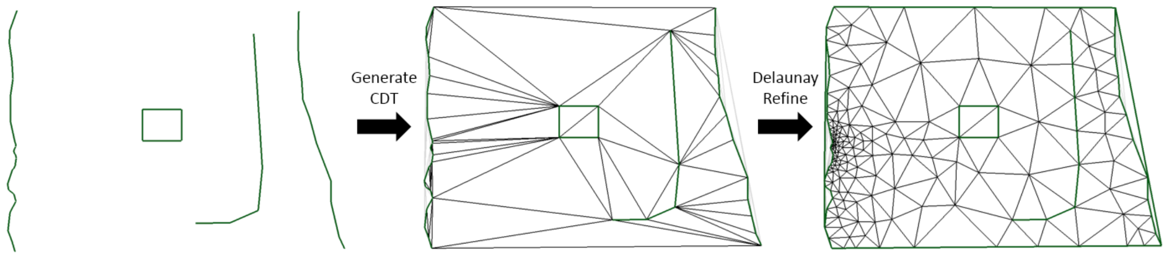


Figure 3: Producing a triangular mesh using Delaunay refinement.

Delaunay refinement can be performed relatively quickly, producing grids containing on the order of hundreds of thousands of cells in a few seconds. It also has the benefit of being able to provide a hard guarantee on minimum angle in triangular grids (up to a limit of about 35 degrees), for situations where that is important. The quality of the grids produced is reasonable, but not perfect: there is often unnecessary variation in cell shape, resulting in undesirable deviation from the CVFD bisection criterion when converted to Voronoi grids, and the number of cells produced for a given set of geometric constraints is often higher than the optimal. It is, however, very useful as a post-processing step on a higher-quality grid produced by the multi-level optimisation method.

1.3.2 Multi-level Optimisation

The multi-level optimisation algorithm implemented in AlgoMesh aims to produce higher-quality meshes with fewer cells than pure Delaunay refinement. It is an approach utilising a combination of Delaunay refinement and a weighted Lloyd smoothing technique (Tournois et al., 2008). We give only high-level notes on the workings of the algorithm here, focusing instead on its application to mesh generation for groundwater models in AlgoMesh. Details on the algorithm itself are given at length in the literature.

Strictly speaking, this algorithm adopts a multi-level *variational* approach, as opposed to a global optimisation. Execution is broken into multiple levels. At each level, new vertices are introduced using a limited Delaunay refinement process, and then all vertices of the mesh are moved towards their optimal positions according to an energy function, until some relative movement threshold is met. The number of vertices introduced at each level, and ultimately the number of levels, is controllable by the modeller through a parameter. This parameter acts as a trade-off between the quality of the optimisation – minimising the number of cells within the given constraints – and its speed of execution.

The energy function used to determine optimal relative vertex positions is derived from an input cell sizing field over the model domain. This sizing field itself is generated based on the mathematical concept of *local feature size* (Amenta and Bern, 1998), a function that smoothly varies from smaller values at boundaries and where there are many input points and line segments, to larger values in open areas of the model. This provides a “natural” grading of cell size from small to large throughout the model domain, but it can be controlled by the modeller both globally – through parameters influencing the local feature size calculations – and locally – by adding polygonal regions in which maximum cell sizes are specified, for example, or re-sampling polyline features such as streamlines with points at different spacing intervals. Figure 4 shows an example of a triangular mesh being produced with the multi-level optimisation, from initial CDT to one of the intermediate levels, and then the final mesh.

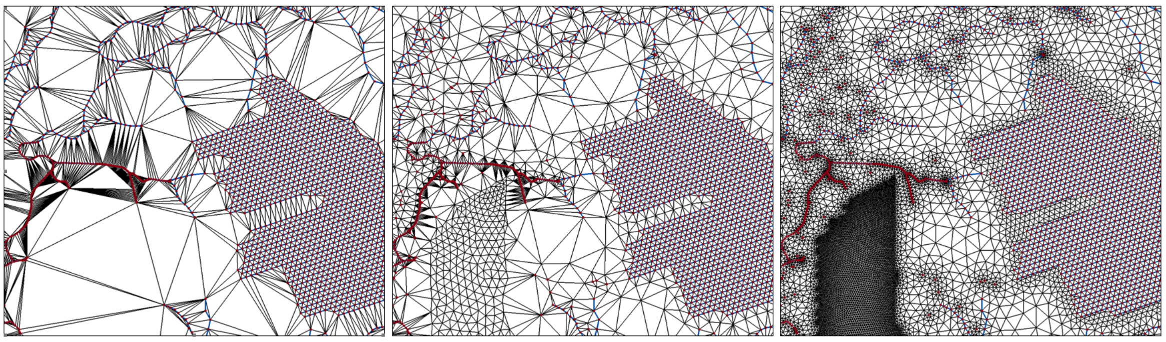


Figure 4: Progression of triangular mesh generation using the multi-level optimisation method.

A limitation of the optimisation approach is that it does not provide any guarantees on the minimum angle within triangles, which means that in certain areas some narrow triangles may be generated. Furthermore, it does not guarantee that all triangles incident to edge constraints will satisfy the Delaunay property, which can create concave cells when the triangular mesh is converted to its dual Voronoi mesh.

Combining the optimisation with a step of standard Delaunay refinement, however, allows these limitations to be overcome. The optimisation process is run first, producing the majority of triangles in the mesh. Then a Delaunay refinement step is run on this mesh, creating additional triangles to enforce the Delaunay criterion and eliminate exceptionally narrow triangles in highly-constrained areas of the model. Typically the number of additional triangles created is quite small – between a few hundred and a few thousand in meshes containing tens of thousands of triangles. Once the triangular mesh has been completed, a Voronoi mesh may be produced by converting the mesh to its dual representation. Figure 5 shows the completed Voronoi mesh generated from the geometry of the above example.

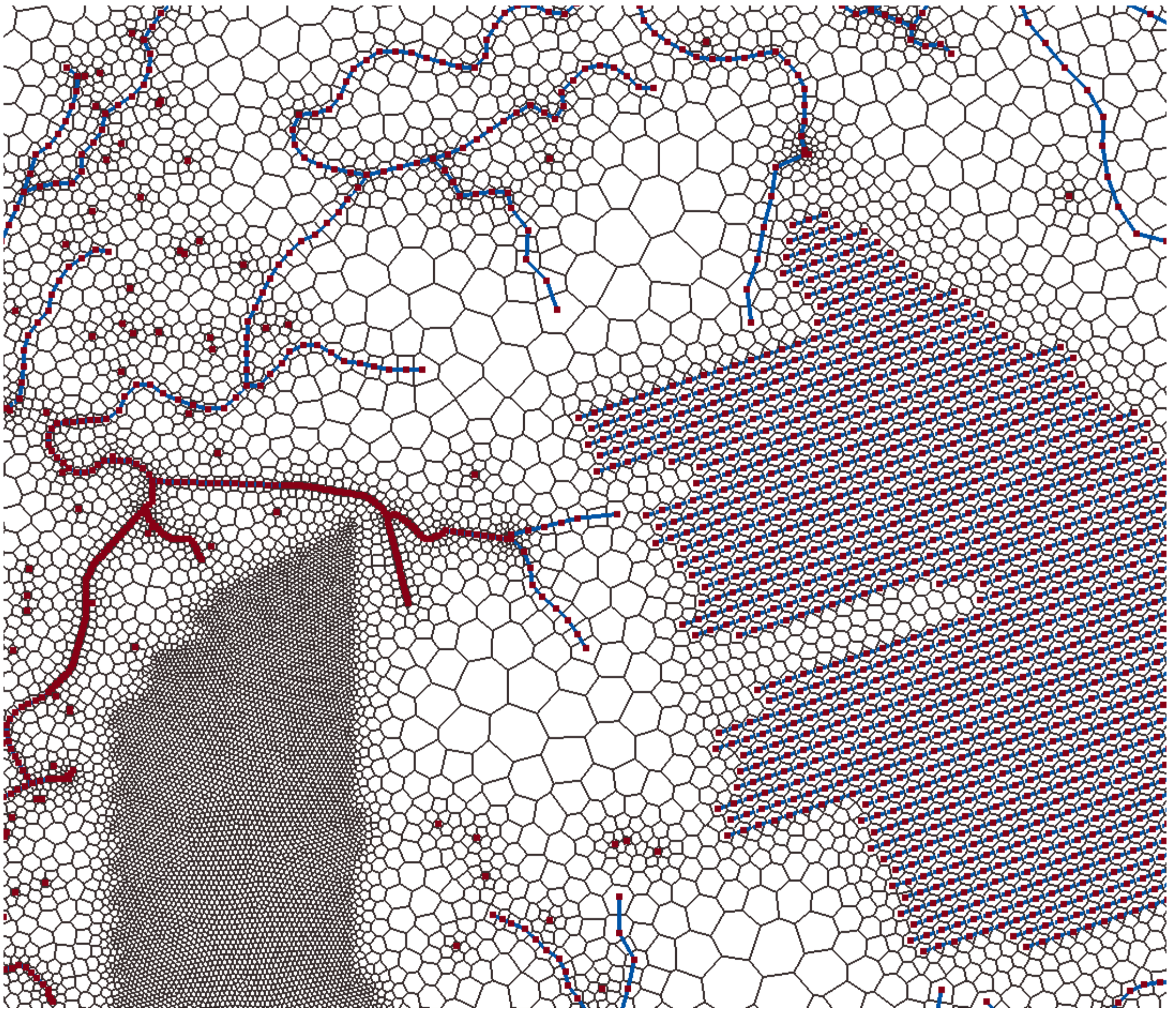


Figure 5: A zoomed-in snapshot of a complete Voronoi mesh generated in AlgoMesh. Selective refinement can be seen in a polygonal region in the bottom-left, and an induced perfect hexagonal sub-grid is present on the right-hand side.

1.4 User Interface Capabilities

The AlgoMesh GUI is a Windows desktop application running on the Microsoft .NET Framework. The GUI provides a fast, easy-to-navigate two-dimensional mesh display and allows execution of the mesh generation algorithms. In addition, it has a number of useful tools to aid in building Voronoi and triangular mesh MODFLOW-USG models. Point, polyline and polygon geometry may be brought in from GIS and CAD files in many common formats – most notably SHP, MID/MIF and DXF – and generated meshes may be exported to GIS for use with external tools.

Polyline and polygon features may be strictly linear, or may be fit to cubic spline curves to allow smoothly curving features to be better approximated. In both cases, AlgoMesh allows these features to be re-sampled with user-defined spacing between points, which is an important mechanism for controlling cell size (mesh resolution) along linear features.

Maximum cell sizes may be defined within polygons to enforce more refinement in the generated mesh in certain areas of a model (Figure 6).

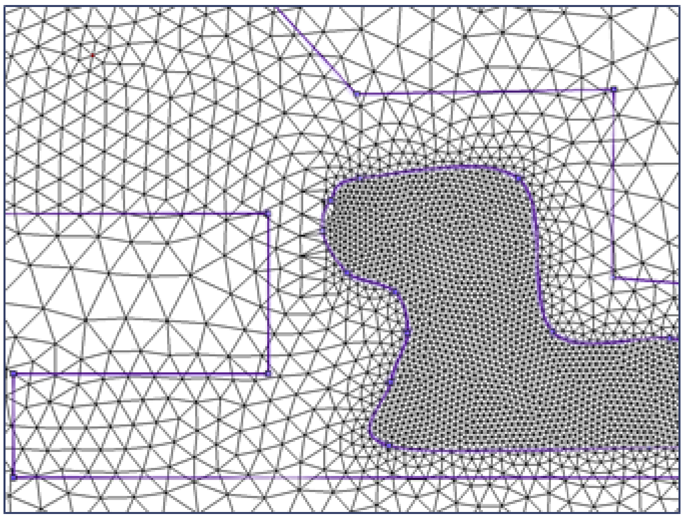


Figure 6: Nested refinement areas.

Furthermore, a polygon may be used as the boundary of a structured sub-grid to be inserted in a mesh, with unstructured cells grading out from the structured area (Figure 7). Structured sub-grid cells may be rectangles or perfect hexagons in Voronoi grids, or equilateral triangles in triangular grids. These structured sub-grid cells promote accuracy locally by ensuring conformance with CVFD assumptions within specified areas of a model; for example, they have been used to closely match longwall boundaries in underground coal mining scenarios, and to produce cells precisely at a desired resolution in these areas.

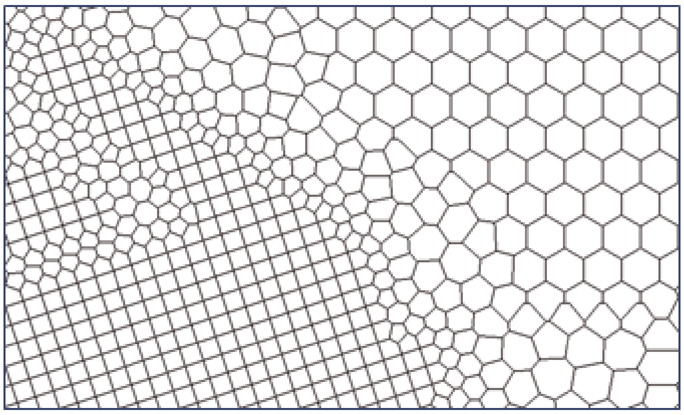


Figure 7: A retangular structured sub-grid within a Voronoi grid.

Material properties and boundary conditions may be specified in AlgoMesh by interpolation from a triangulated point set (TIN), by mapping values from a GIS polygon field, or by direct mapping from a CSV file indexed by cell number. AlgoMesh outputs core discretization and groundwater flow files for MODFLOW-USG (notably DISU, BAS6 and LPF), as well as, optionally, several boundary condition packages: CHD, DRN, EVT, GHB, RCH, RIV, TVM and WEL. Grid specification files (GSF) and Visualization Toolkit (VTK) files are also written to permit interoperation with other MODFLOW-USG tools.

A comprehensive reference of the various sections of the AlgoMesh user interface is given in the User Interface section of this guide.

2 Getting Started

AlgoMesh is a fully-featured model builder for MODFLOW-USG, and may also be used as a two-dimensional mesh generator for HydroGeoSphere™, FEFLOW® or other models that are able to accept two-dimensional triangular or polygonal (Voronoi) grids in shapefile format.

The following sections describe what is required to use AlgoMesh on your system, how to install and license AlgoMesh to get it up and running, and how to use AlgoMesh with particular simulation models, linking to the most appropriate How To tutorials along the way to get you started.

2.1 System Requirements

Before purchasing or installing AlgoMesh, please note the following system requirements and recommendations.

Operating system: AlgoMesh is supported for execution on Windows 7, Windows 8 and Windows 10 operating systems. Both a 64-bit (x64) and a 32-bit (x86) version of the software are available; 64-bit is recommended when running on a 64-bit operating system.
CPU: There are no specific CPU requirements - any x86 or x64 architecture CPU is sufficient - but a faster CPU with more cores is recommended to allow meshes to be generated more rapidly.
RAM: A minimum of 1GB of RAM is required for basic operation, but building large models may require significantly more than this; at least 8GB RAM is recommended.
Hard drive space: Less than 1GB of available hard drive space is required for the installation of the software; however, additional storage space is required to store mesh and model data.
Graphics: A dedicated 3D graphics card (e.g. NVIDIA or AMD) is not required, but will be used if present to accelerate the OpenGL graphics rendering used in AlgoMesh.
Screen resolution: AlgoMesh should be run at a resolution of 1280x1024 or higher to fit all user interface elements on the screen; 1920x1080 or higher is recommended.
Remote Desktop: Running AlgoMesh while accessing a machine using Microsoft's Remote Desktop Protocol (RDP) is supported, but accelerated OpenGL graphics rendering is not available via RDP, and viewing and manipulating meshes in AlgoMesh may be somewhat slower. Third-party remote access solutions such as TeamViewer or VNC may allow this limitation to be bypassed. *Please note that the software licence may restrict concurrent use of the software on a single machine; please read the conditions carefully and contact HydroAlgorithmics Support with any queries.*

2.2 Installing AlgoMesh

Upon purchasing or requesting a trial version of AlgoMesh, you will be provided with a link from which you can download installation packages for the software. There are two installation packages - one for 32-bit operating systems and one for 64-bit operating systems:

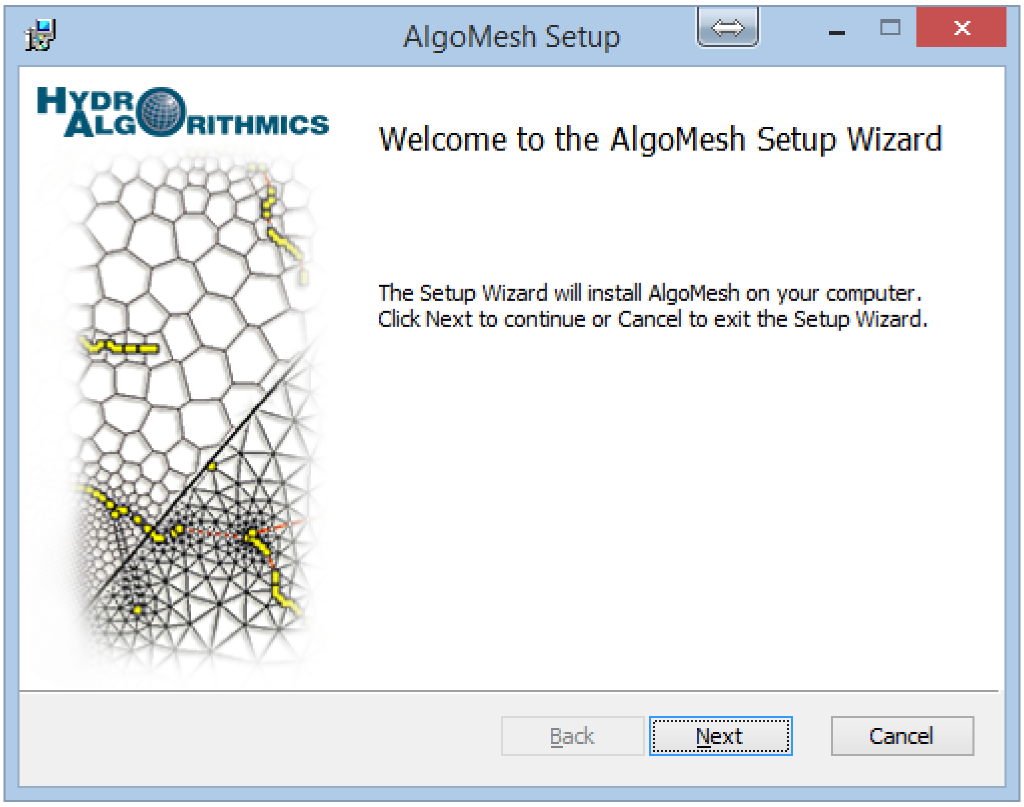
- **AlgoMesh-<version>-x64.msi** - run this to install AlgoMesh on a 64-bit (x64) Windows operating system.
- **AlgoMesh-<version>-x86.msi** - run this to install AlgoMesh on a 32-bit (x86) Windows operating system.

Note that the ***<version>*** tag in these file names is replaced with the actual AlgoMesh software version installed by the package, and this may vary depending on when you obtain it.

If you are unsure whether you have a 64-bit or a 32-bit operating system, look under the System properties in Control Panel, or ask your IT administrator.

Note that you will require administrator privileges to install AlgoMesh on your system. If you do not have these, contact your IT administrator to arrange installation of the software.

Once you run the installation package, the setup wizard will take you through the process of installing AlgoMesh on your machine (see Figure 8). Follow the instructions, clicking **Next** as it prompts you. You will be asked to accept the software licence agreement, to confirm the location in which AlgoMesh is installed, and to confirm which components of AlgoMesh you wish to install. By default, AlgoMesh will install to your *Program Files* folder (normally) or the *Program Files (x86)* folder (if installing the 32-bit version on a 64-bit operating system), and all components of the software will be included. If you are uncertain as to what or where to install, it is usually fine to accept the defaults.

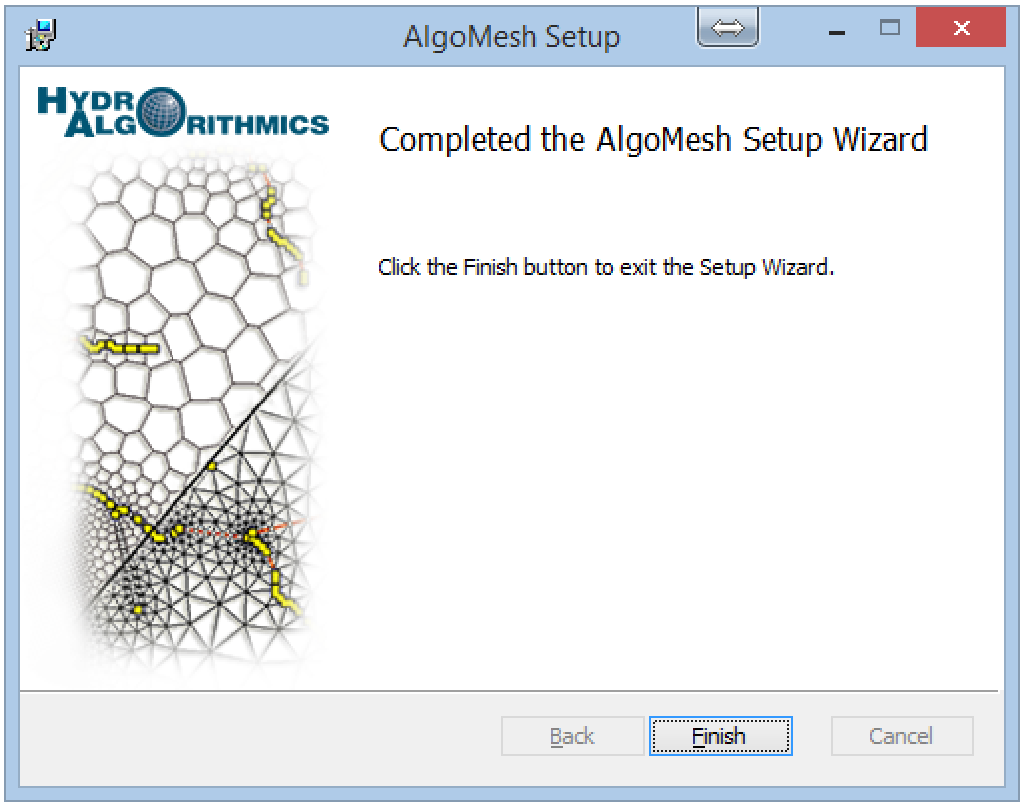


Welcome to the AlgoMesh Setup Wizard

The Setup Wizard will install AlgoMesh on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Figure 8: The AlgoMesh setup wizard.

After you have stepped through the installation wizard and clicked the **Install** button, installation will proceed. You may be asked whether you wish to allow the installation to make changes to your computer; click **Yes** to allow this so that installation can proceed. Once done, you will receive a notification saying that the installation has completed, as in Figure 9. Click **Finish** to close the setup wizard.



Completed the AlgoMesh Setup Wizard

Click the Finish button to exit the Setup Wizard.

Figure 9: Completion of the AlgoMesh installation.

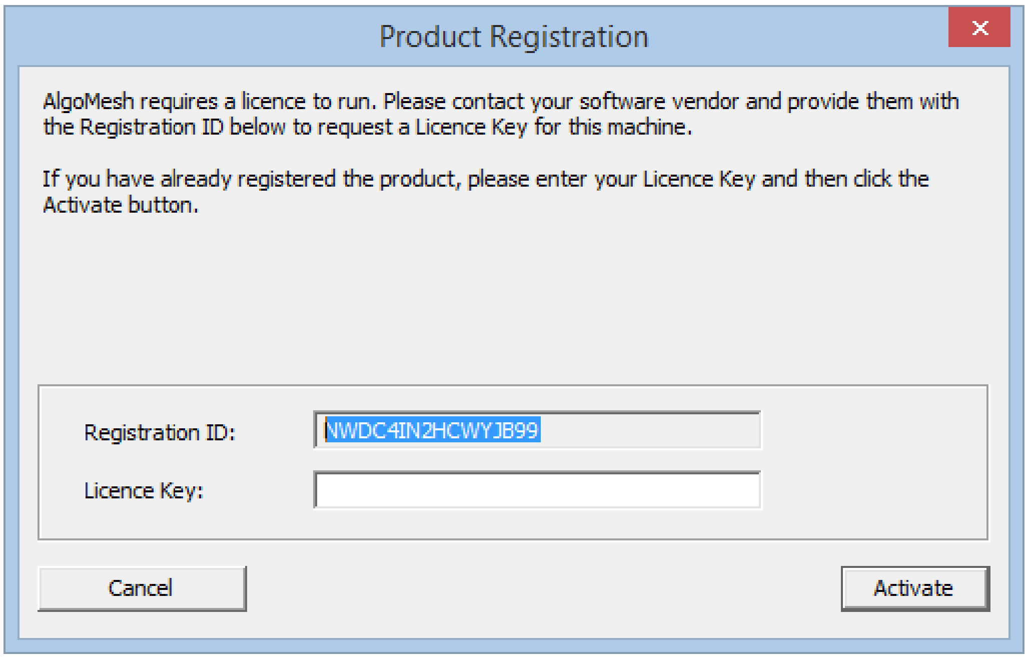
Once AlgoMesh is installed, look for the AlgoMesh shortcut in the list of applications on your Windows Start screen, and run it. The first time you run the software, you will be prompted to enter a licence code before the user interface will load; see Licensing AlgoMesh for more information.

Uninstalling AlgoMesh

Should you wish to uninstall AlgoMesh, go to **Programs and Features** in **Control Panel**, find **AlgoMesh** in the list of installed programs, and click the **Uninstall** button. Click **Yes** when prompted if you are sure you wish to uninstall, and **Yes** again if prompted to allow the installation package to make changes to your computer. The uninstallation process will complete, and the AlgoMesh entry should be removed from the list of installed programs.

2.3 Licensing AlgoMesh

The first time you run AlgoMesh on your computer, a Product Registration window will appear (Figure 10).



Product Registration

AlgoMesh requires a licence to run. Please contact your software vendor and provide them with the Registration ID below to request a Licence Key for this machine.

If you have already registered the product, please enter your Licence Key and then click the Activate button.

Registration ID: NWDC4IN2HCWYJB99

Licence Key:

Figure 10: The Product Registration window appears when you first run AlgoMesh, requesting your licence key.

This window provides you with a *Registration ID*, which is specific to your computer. Copy and paste this key into an email to HydroAlgorithmics Support (support@hydroalgorithmics.com) to receive your permanent *Licence Key* if you have purchased AlgoMesh, or to request a time-limited trial *Licence Key*, if you are evaluating AlgoMesh.

You can press the **Cancel** button to close AlgoMesh while waiting for your Licence Key.

Once you receive an email containing your Licence Key, start AlgoMesh again, and **copy and paste it into the Licence Key text box**, then click **Activate**. This will activate AlgoMesh for use on your computer. If you have purchased and activated a permanent Licence Key, you will not need to enter it again when you next run AlgoMesh. If you have activated a trial Licence Key, the Product Registration window will appear again once your trial period has expired, after which you may contact HydroAlgorithmics again to purchase a licence if desired.

2.4 Using AlgoMesh with MODFLOW-USG

MODFLOW-USG (Panday et al., 2012) is a control-volume finite difference groundwater flow simulator released by the U.S. Geological Survey. At the time of writing, information about MODFLOW-USG as well as executables and source code are available from the USGS' MODFLOW-USG website here: http://water.usgs.gov/ogw/mfusg/

Building MODFLOW-USG models with AlgoMesh is typically done using Voronoi grids, as these possess geometric qualities that particularly suit the underlying mathematical formulation of MODFLOW-USG (see the Grid Generation with AlgoMesh section). The process of building a model starts with geometry import and mesh generation, then moves onto model setup and applying material properties and boundary conditions to the generated mesh, after which MODFLOW-USG model files may be exported to run the model. Finally, although AlgoMesh only provides limited post-processing support and is not a visualisation package in itself, you can use it to read in binary head, drawdown and budget results produced by a MODFLOW-USG simulation, and output them in a textual CSV format for reading into other post-processing tools.

To get started building MODFLOW-USG models, follow the how-to tutorials in the recommended order below. These are divided into mesh generation and model building topics.

Mesh Generation

1. Start by learning about the basic Voronoi mesh generation process in AlgoMesh: Create a Simple Voronoi Mesh Using Delaunay Refinement
2. Learn how to utilise the recommended mesh generation method - multi-level optimisation - to produce high-quality Voronoi grids: Generate a High-Quality Voronoi Mesh using Multi-Level Optimisation
3. In a real project, you will need to import geometry prior to mesh generation from GIS shapefiles: Import Geometry from GIS
4. One of the most powerful and useful features of AlgoMesh is the ability to control mesh resolution arbitrarily in different areas of your model domain. The Use Property Polygons to Control Mesh Cell Sizing how-to explains how you can do this.

Model Building

5. Build your first AlgoMesh MODFLOW-USG model: Set up a Steady-State MODFLOW-USG Model
6. Learn how to pinch out cells or remove cells from inactive areas in your model to minimise cell counts: Remove or Pinch Out Unnecessary MODFLOW-USG Model Cells
7. Use AlgoMesh to read in binary head results and convert them to a form usable in post-processing and visualisation applications: Load MODFLOW-USG Results
8. Convert a steady-state model into a transient model, with boundary conditions varying throughout the simulation: Set up a Transient MODFLOW-USG Model

Further Reading

The Manually Digitise and Edit Splines and Manually Modify a Mesh how-to sections provide some more options for manipulating input geometry and customising the final mesh for your model.

Once you are comfortable with the basic mesh generation and model building workflow in AlgoMesh, take a look at the Advanced Topics section to learn about some additional features of AlgoMesh that may come in handy when using the software in your own projects.

2.5 Using AlgoMesh with HydroGeoSphere™

HydroGeoSphere™ (HGS) is a fully-integrated control-volume finite element simulator developed by Aquanty.

AlgoMesh may be used to generate two-dimensional triangular meshes for use in HGS models. To get started building 2D triangular meshes, follow the how-to tutorials in the recommended order below.

1. Start by learning about the basic triangular mesh generation process in AlgoMesh: Create a Simple Triangular Mesh Using Delaunay Refinement
2. Learn how to utilise the recommended mesh generation method - multi-level optimisation - to produce high-quality triangular grids: Generate a High-Quality Triangular Mesh using Multi-Level Optimisation
3. In a real project, you will need to import geometry prior to mesh generation from GIS shapefiles: Import Geometry from GIS
4. One of the most powerful and useful features of AlgoMesh is the ability to control mesh resolution arbitrarily in different areas of your model domain. The Use Property Polygons to Control Mesh Cell Sizing how-to explains how you can do this.

The Manually Digitise and Edit Splines and Manually Modify a Mesh how-to sections provide some more options for manipulating input geometry and customising the final mesh for your model.

Once you have a mesh generated, use **File**->**Triangular Mesh**->**Export 2D Mesh to HydroGeoSphere** to produce an .AH2 file, which can be read in for use in a HGS model using a grok command.

2.6 Using AlgoMesh with Other Simulation Models

AlgoMesh may be used to generate two-dimensional triangular or Voronoi meshes for use with any simulation model that can read in a 2D mesh from a polygon shapefile. To get started building meshes, follow the how-to tutorials in the recommended order below.

1. Start by learning about the basic triangular or Voronoi mesh generation process in AlgoMesh, depending on the type of grid you need: Create a Simple Triangular Mesh Using Delaunay Refinement or Create a Simple Voronoi Mesh Using Delaunay Refinement
2. Learn how to utilise the recommended mesh generation method - multi-level optimisation - to produce high-quality triangular or Voronoi grids: Generate a High-Quality Triangular Mesh using Multi-Level Optimisation or Generate a High-Quality Voronoi Mesh using Multi-Level Optimisation
3. In a real project, you will need to import geometry prior to mesh generation from GIS shapefiles: Import Geometry from GIS
4. One of the most powerful and useful features of AlgoMesh is the ability to control mesh resolution arbitrarily in different areas of your model domain. The Use Property Polygons to Control Mesh Cell Sizing how-to explains how you can do this.

The Manually Digitise and Edit Splines and Manually Modify a Mesh how-to sections provide some more options for manipulating input geometry and customising the final mesh for your model.

Once you have a mesh generated, use **File**->**Triangular Mesh**->**Save Cell Geometry** for a triangular grid, or **File**->**Voronoi Mesh**->**Save Cell Geometry** for a Voronoi grid, to export a polygon shapefile containing your two-dimensional mesh, which can then be brought into your target simulation model.

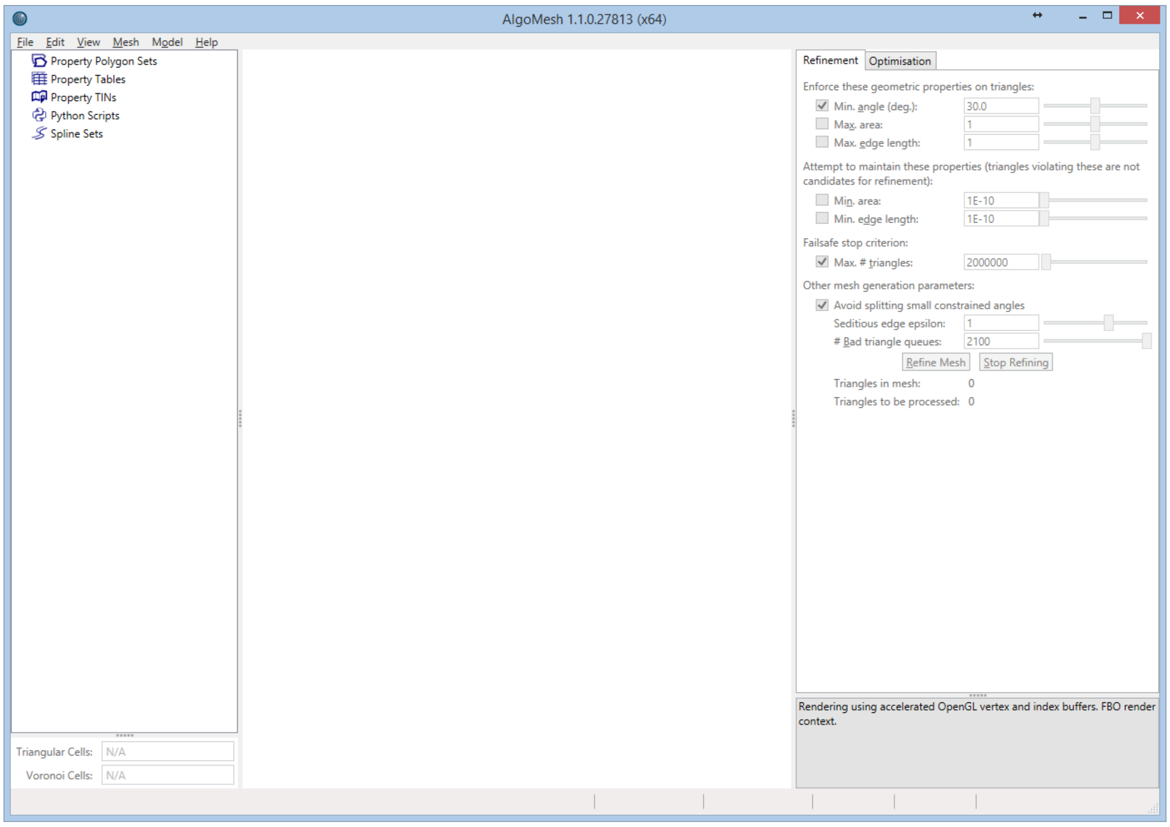
3 How To...

This section contains tutorials on performing a number of important tasks in AlgoMesh. You can follow along with these in the order they appear, or read the Getting Started section for suggestions on which tutorials to read for particular modelling tasks.

3.1 Create a Simple Triangular Mesh Using Delaunay Refinement

In this how-to, you will load an example AlgoMesh project file containing geometry for a simple model domain, and use the Delaunay refinement method to produce a triangular mesh from it. You will explore the effect of setting minimum angle and maximum edge length constraints on the refinement process. Note that there is an alternative version of this how-to section using Voronoi grids instead of triangular grids; follow whichever of these is relevant to the type of grid you want to create.

Start by launching the AlgoMesh user interface. You should be presented with a screen similar to that shown in Figure 11.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 0
Triangles to be processed: 0

Rendering using accelerated OpenGL vertex and index buffers. FBO render context.

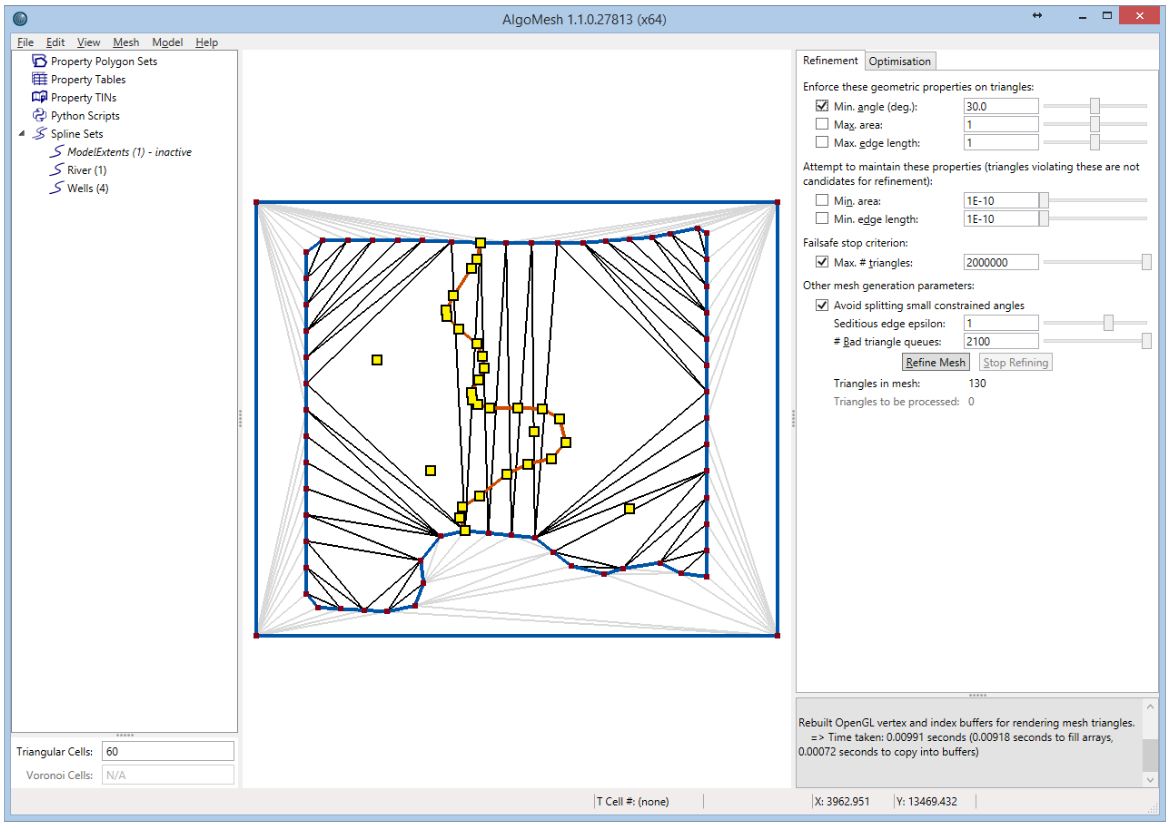
Triangular Cells: N/A
Voronoi Cells: N/A

Figure 11: The AlgoMesh user interface.

Now load the project file river-mesh-domain.amproj.

➢ Go to the **File** menu and choose **Open Project**.
➢ Browse to the **Samples** folder of your AlgoMesh installation.
➢ Select **river-mesh-domain.amproj** and click **Open**.

AlgoMesh will load the geometry for our mesh domain from the project file, including a bounding box, a model extents polygon, a river polyline and four points representing well locations. Your screen should look like Figure 12.



Triangular Cells: 60
Voronoi Cells: N/A

Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00991 seconds (0.00918 seconds to fill arrays,
0.00072 seconds to copy into buffers)
```

T Cell #: (none) X: 3962.951 Y: 13469.432

Figure 12: The initial CDT of the mesh domain.

All of this geometry was initially brought into AlgoMesh from GIS shapefiles, and later saved into the AlgoMesh project (.amproj) file we just loaded. The Import Geometry from GIS how-to guide details this process.

There are a few things to note in the initial view of the mesh domain. Firstly, note that AlgoMesh shows triangles connecting all of the points of the model extents boundary. This is the Constrained Delaunay Triangulation (CDT) of the input geometry, which is automatically produced by AlgoMesh whenever new geometry is brought in. The CDT is merely a starting point for a mesh to be generated in AlgoMesh; it is not suitable as a simulation grid itself.

Secondly, note the triangles coloured grey around the edges of the model extents. These triangles are grey because they have been removed from the mesh domain, i.e. they will not be considered part of the geometry to be meshed, nor part of the final model.

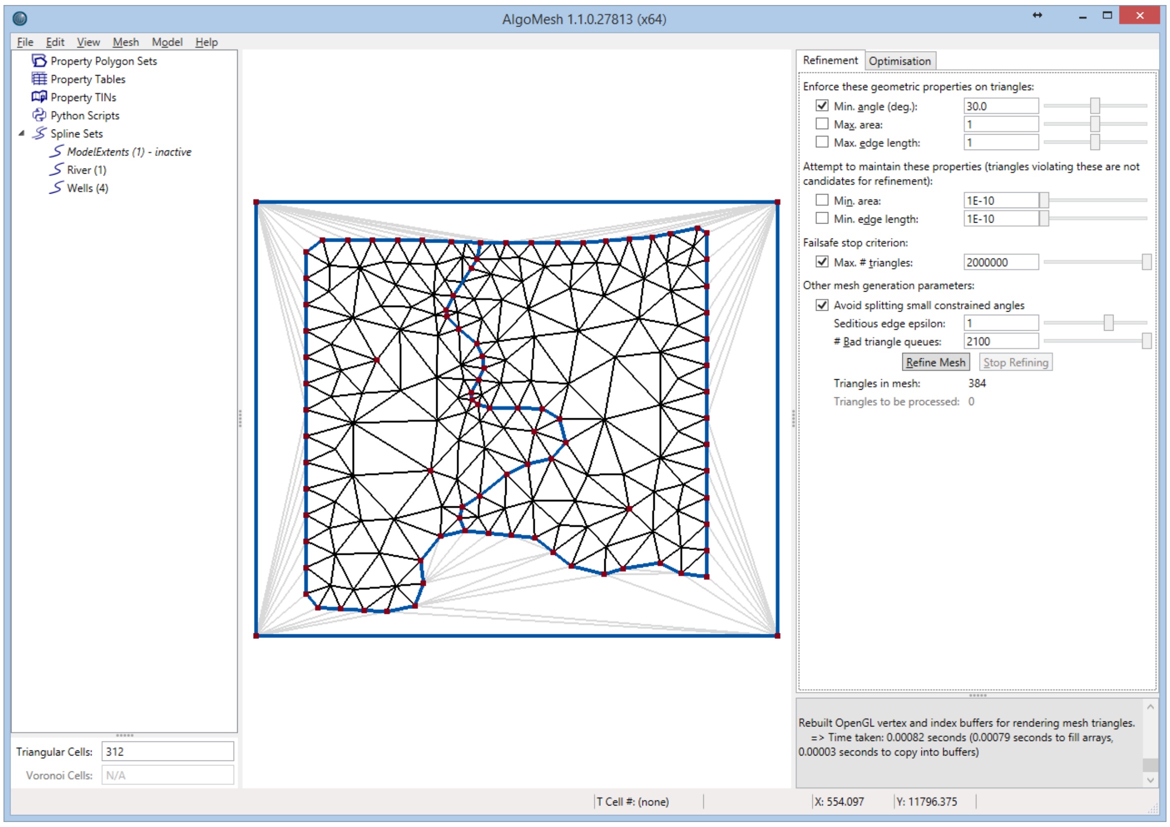
Finally, note that the river and well features have not yet been added into the mesh geometry. Instead, they are displayed on top of the CDT in yellow and orange. These will be added automatically into the mesh by AlgoMesh as soon as the mesh generation process is executed. Keeping these features independent of the mesh is useful, as it allows us to move or re-sample them easily and regenerate the mesh to adjust the input geometry as needed, without having to re-import them each time.

We are now ready to generate the mesh using the Delaunay refinement method.

- ➢ On the right-hand side of the AlgoMesh window, ensure the **Refinement** tab is selected, and leave the settings at their defaults (30 degree minimum angle).
- ➢ Click the **Refine Mesh** button.

- ➢ Once the display updates to show the generated mesh, go to the **View** menu and turn off **Splines**. This hides the yellow and orange river and well features to make it easier to see the resulting mesh underneath.

Your AlgoMesh window should now look like Figure 13. Note that all of the points and line segments that were present in the river and well geometry are included as points and edges in the generated triangular grid.



Enforce these geometric properties on triangles:
[✓] Min. angle (deg.): 30.0
[ ] Max. area: 1
[ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
[ ] Min. area: 1E-10
[ ] Min. edge length: 1E-10

Failsafe stop criterion:
[✓] Max. # triangles: 2000000

Other mesh generation parameters:
[✓] Avoid splitting small constrained angles
Seditious edge epsilon: 1
Bad triangle queues: 2100

Triangles in mesh: 384
Triangles to be processed: 0

Triangular Cells: 312
Voronoi Cells: N/A

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00082 seconds (0.00079 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

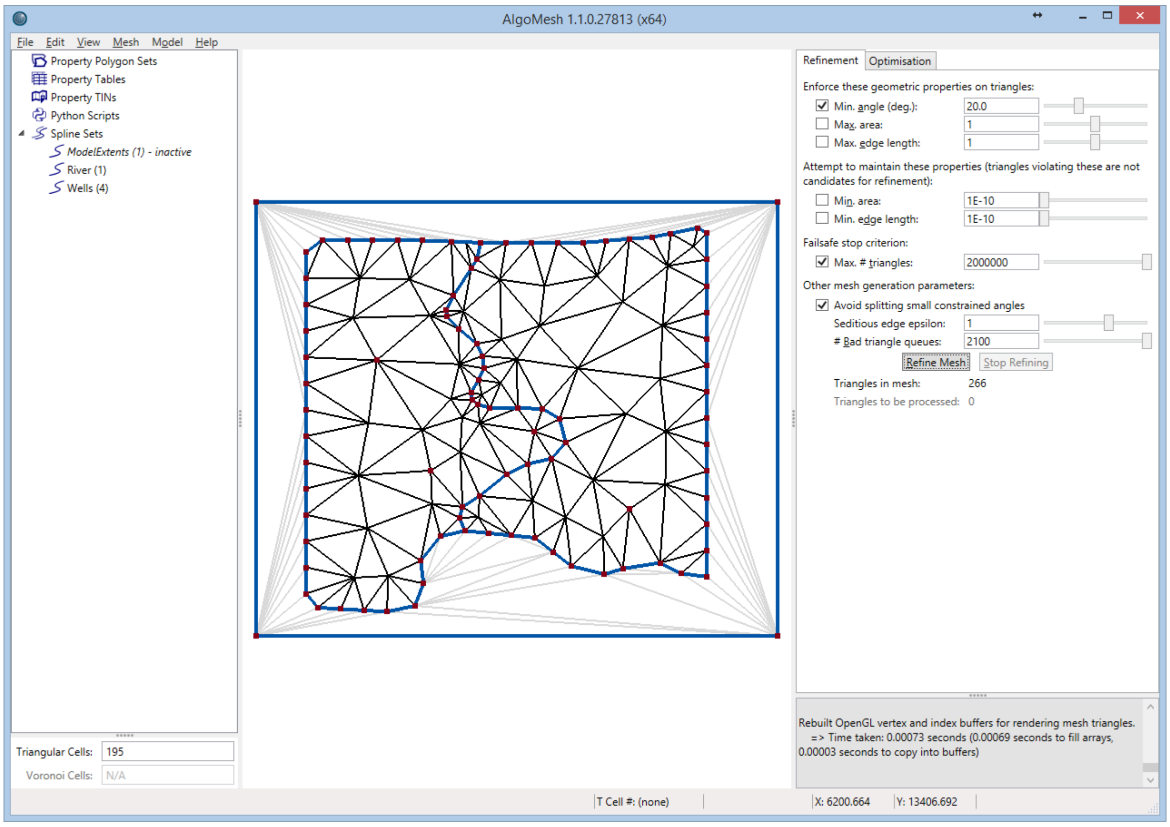
T Cell #: (none) | X: 554.097 | Y: 11796.375

Figure 13: The triangular mesh after refinement with a 30 degree minimum angle.

The result is a complete triangular mesh, incorporating our input geometry and the 30 degree default minimum angle constraint on triangles. The minimum angle constraint can have a significant effect on the resulting mesh. To see this, try changing its value and regenerating the mesh.

- ➢ In the Refinement tab settings, change **Min. angle (deg.)** to **20.0**. Ensure its checkbox remains checked.
- ➢ Click **Refine Mesh**.

AlgoMesh regenerates the mesh from the initial geometry; it does not start from the existing generated mesh. The result should be similar to Figure 14. Note how the mesh now contains fewer triangles, but some of them are quite narrow.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 20.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 266
Triangles to be processed: 0

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.00073 seconds (0.00069 seconds to fill arrays, 0.00003 seconds to copy into buffers)

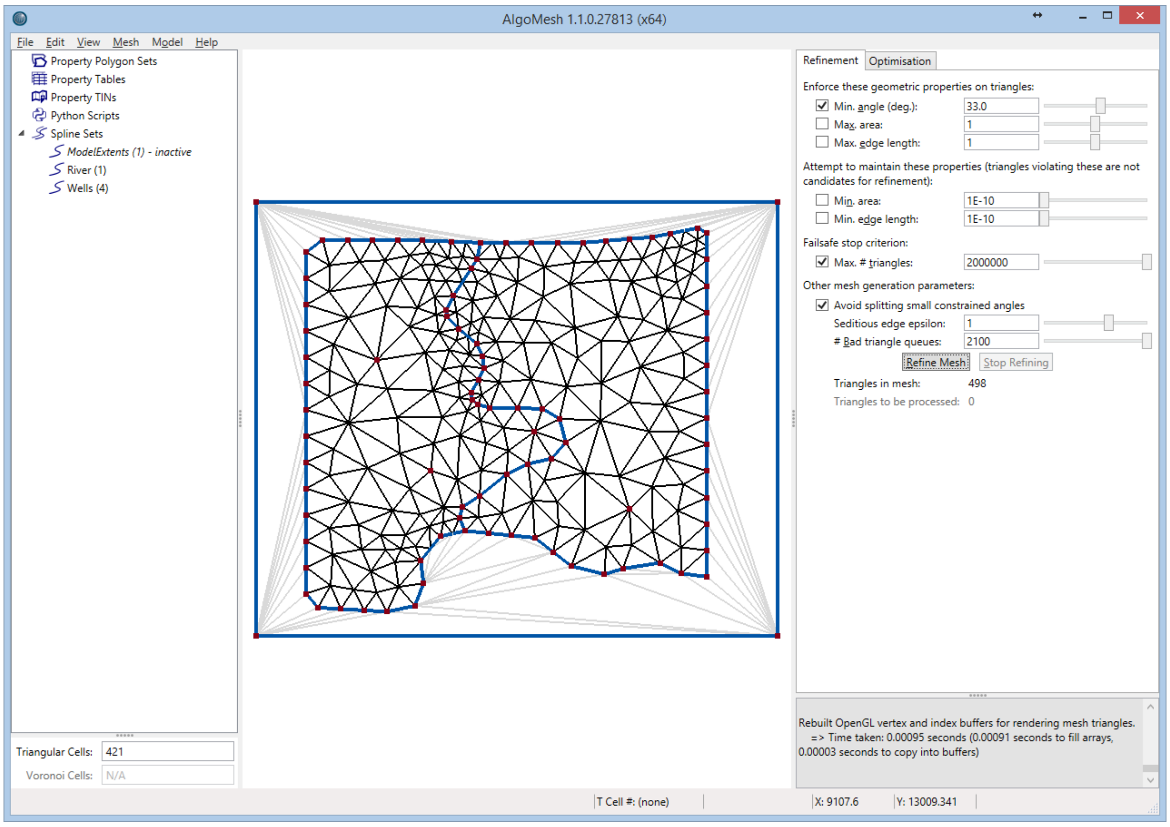
Triangular Cells: 195
Voronoi Cells: N/A

Figure 14: The triangular mesh after refinement with a 20 degree minimum angle.

Now try a higher minimum angle.

- ➢ In the Refinement tab settings, change **Min. angle (deg.)** to **33.0**. Ensure its checkbox remains checked.
- ➢ Click **Refine Mesh**.

The result should be similar to Figure 15. Note how the mesh now contains more triangles, but with many that are closer to ideal equilateral triangles than the previous mesh. It is not recommended to increase the minimum angle too far beyond this point; with a high setting for minimum angle, the Delaunay refinement method may over-refine in some parts of the mesh, and may not even be able to complete, stopping only when the maximum number of triangles is reached.



Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 33.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 498
Triangles to be processed: 0

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.00095 seconds (0.00091 seconds to fill arrays, 0.00003 seconds to copy into buffers)

Triangular Cells: 421
Voronoi Cells: N/A

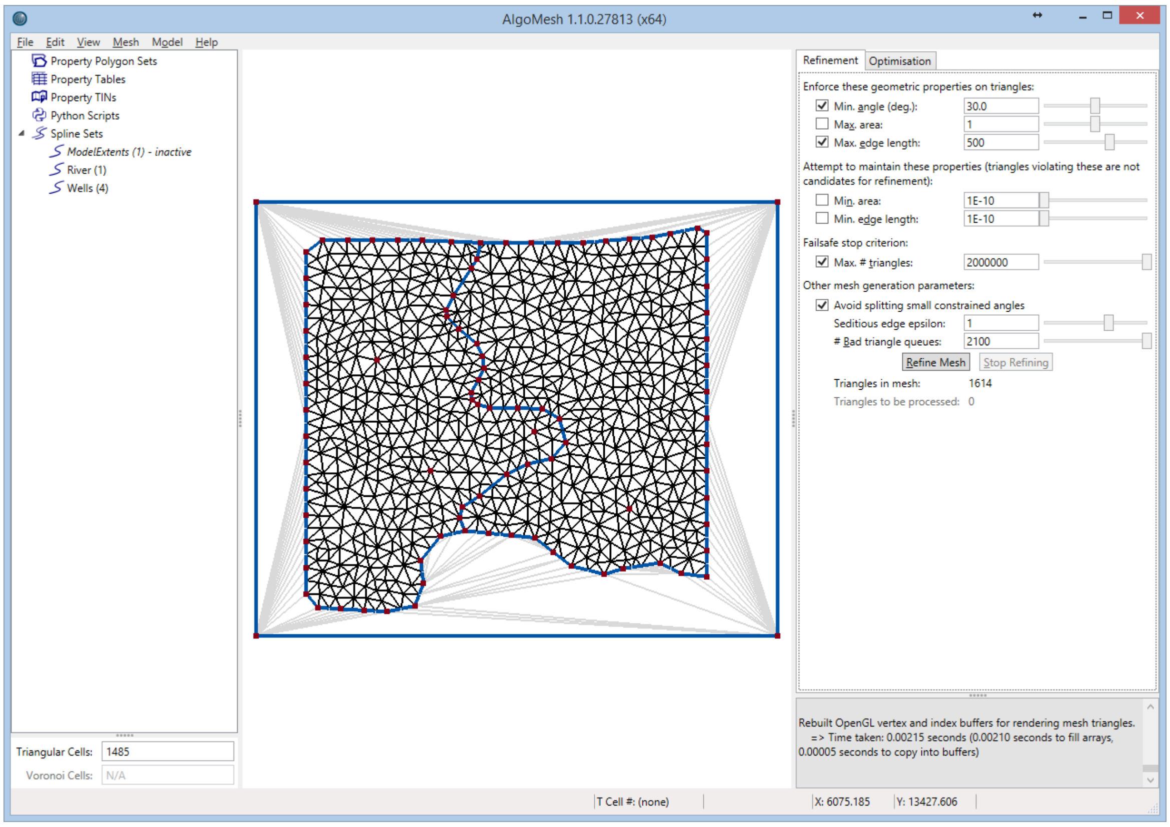
T Cell #: (none) X: 9107.6 Y: 13009.341

Figure 15: The triangular mesh after refinement with a 33 degree minimum angle.

You can also set a global maximum cell size (triangle edge length) to produce a more uniform mesh over the entire domain.

- ➢ In the Refinement tab settings, first change **Min. angle (deg.)** back to **30.0**. Ensure its checkbox remains checked.
- ➢ Check the **Max. edge length** checkbox, and set its value to **500**.
- ➢ Click **Refine Mesh**.

Figure 16 shows the resulting mesh, in which all triangles have a longest edge length of 500 or less.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [✓] Max. edge length: 500

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 1614
Triangles to be processed: 0

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.00215 seconds (0.00210 seconds to fill arrays, 0.00005 seconds to copy into buffers)

Triangular Cells: 1485
Voronoi Cells: N/A

T Cell #: (none) | X: 6075.185 | Y: 13427.606

Figure 16: The triangular mesh after refinement with a maximum edge length of 500 and a 30 degree minimum angle.

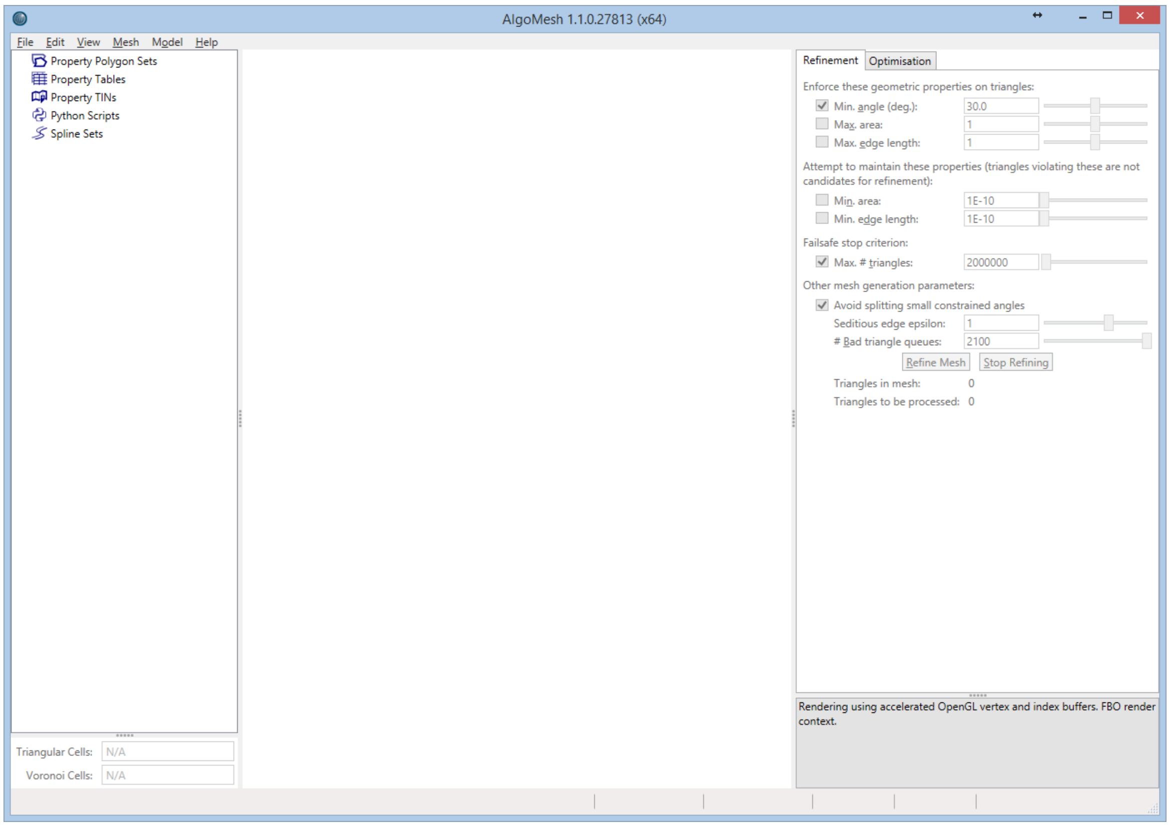
The maximum area constraint acts similarly to the maximum edge length constraint. The minimum area and edge length constraints may be used to stop AlgoMesh from splitting triangles when they get to a certain size, but these are not recommended for use in general, as they may result in a mesh with poor-quality cell geometry.

As it typically runs very quickly, the Delaunay refinement method is useful for getting an initial idea of whether your input geometry will produce an appropriate mesh. However, higher-quality meshes may be generated using the optimisation method. The Delaunay refinement method is still typically used in this case, but it is run instead as a post-processing step after the optimisation, to eliminate the small number of poor-quality triangles that may otherwise remain. More global controls on cell sizing and quality are also available using the optimisation method, and these are detailed in the Generate a High-Quality Triangular Mesh using Multi-Level Optimisation how-to guide.

3.2 Create a Simple Voronoi Mesh Using Delaunay Refinement

In this how-to, you will load an example AlgoMesh project file containing geometry for a simple model domain, and use the Delaunay refinement method to produce a Voronoi mesh from it. You will explore the effect of setting minimum angle and maximum edge length constraints on the refinement process. Note that there is an alternative version of this how-to section using triangular grids instead of Voronoi grids; follow whichever of these is relevant to the type of grid you want to create.

Start by launching the AlgoMesh user interface. You should be presented with a screen similar to that shown in Figure 17.



Enforce these geometric properties on triangles:
- ☑ Min. angle (deg.): 30.0
- ☐ Max. area: 1
- ☐ Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- ☐ Min. area: 1E-10
- ☐ Min. edge length: 1E-10

Failsafe stop criterion:
- ☑ Max. # triangles: 2000000

Other mesh generation parameters:
- ☑ Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 0
Triangles to be processed: 0

Rendering using accelerated OpenGL vertex and index buffers. FBO render context.

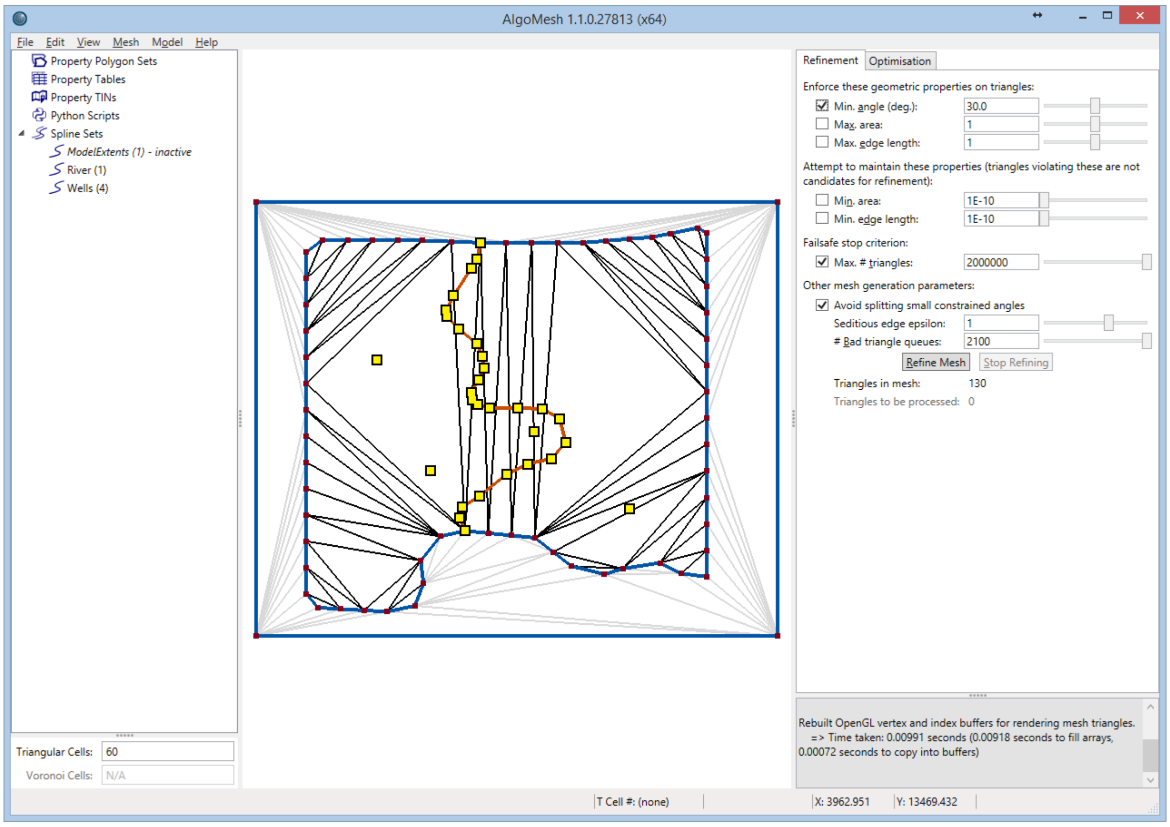
Triangular Cells: N/A
Voronoi Cells: N/A

Figure 17: The AlgoMesh user interface.

Now load the project file river-mesh-domain.amproj.

➢ Go to the **File** menu and choose **Open Project**.
➢ Browse to the **Samples** folder of your AlgoMesh installation.
➢ Select **river-mesh-domain.amproj** and click **Open**.

AlgoMesh will load the geometry for our mesh domain from the project file, including a bounding box, a model extents polygon, a river polyline and four points representing well locations, as in Figure 18.



Enforce these geometric properties on triangles:

- ✓ Min. angle (deg.): 30.0
- ☐ Max. area: 1
- ☐ Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- ☐ Min. area: 1E-10
- ☐ Min. edge length: 1E-10

Failsafe stop criterion:

- ✓ Max. # triangles: 2000000

Other mesh generation parameters:

- ✓ Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

Triangular Cells: 60
Voronoi Cells: N/A

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00991 seconds (0.00918 seconds to fill arrays,
0.00072 seconds to copy into buffers)
```

T Cell #: (none) X: 3962.951 Y: 13469.432

Figure 18: The initial CDT of the mesh domain.

There are a few things to note in the initial view of the mesh domain. Firstly, note that AlgoMesh shows triangles connecting all of the points of the model extents boundary. This is the Constrained Delaunay Triangulation (CDT) of the input geometry, which is automatically produced by AlgoMesh whenever new geometry is brought in. The CDT is merely a starting point for a mesh to be generated in AlgoMesh; it is not suitable as a simulation grid itself.

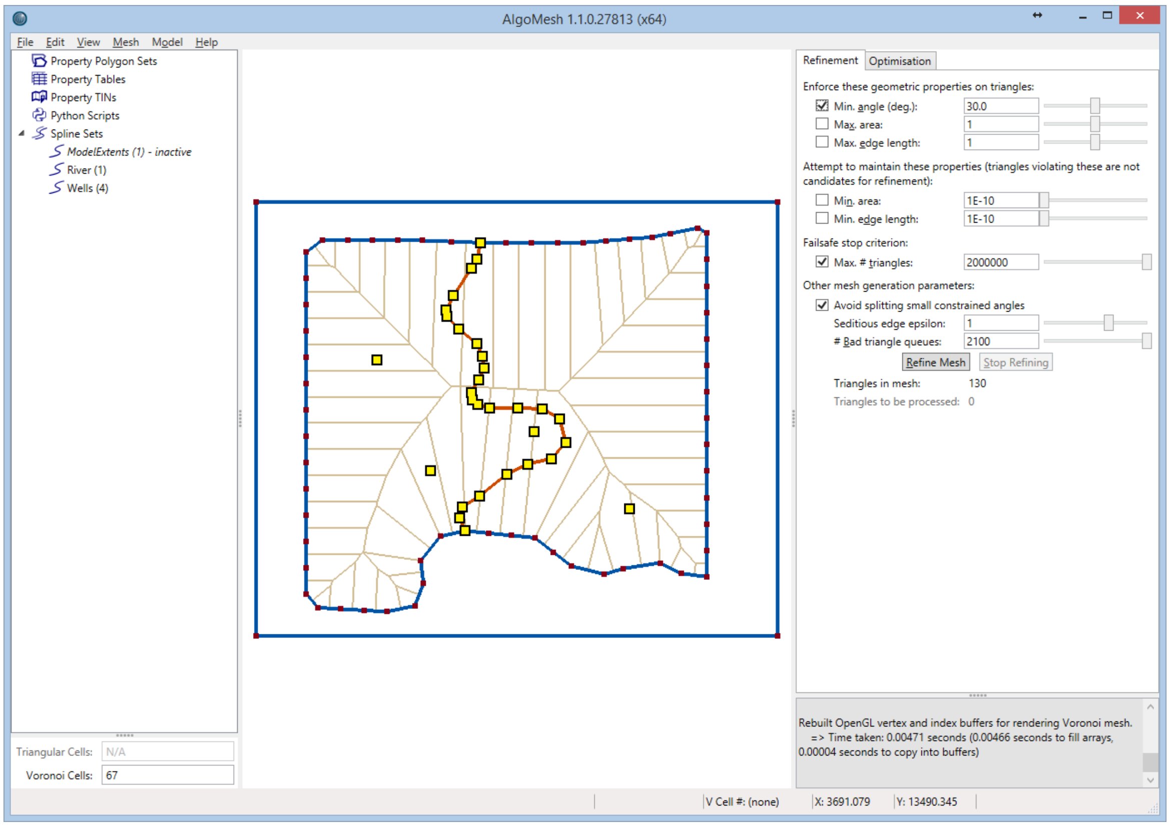
Secondly, note the triangles coloured grey around the edges of the model extents. These triangles are grey because they have been removed from the mesh domain, i.e. they will not be considered part of the geometry to be meshed, nor part of the final model.

Finally, note that the river and well features have not yet been added into the mesh geometry. Instead, they are displayed on top of the CDT in yellow and orange. These will be added automatically into the mesh by AlgoMesh as soon as the mesh generation process is executed. Keeping these features independent of the mesh is useful, as it allows us to move or re-sample them easily and regenerate the mesh to adjust the input geometry as needed, without having to re-import them each time.

We are going to be working with a Voronoi grid instead of a triangular grid, so we would like to display Voronoi cells instead of triangles.

➢ Go to the **View** menu and turn on **Voronoi Cells**.
➢ Go to the **View** menu and turn off **Triangles**.

Your screen should now look like Figure 19. Notice that the Voronoi grid is only shown to the edges of the model extents polygon. The area that was taken up by the grey triangles has been excluded from the Voronoi mesh. When generating a mesh, note that AlgoMesh will always operate on the underlying triangular grid, and many of the constraints on the mesh generation process refer to this triangular grid. The Voronoi grid is obtained as an automated second stage in the process, as a conversion from the generated triangular grid to its Voronoi dual representation.



Enforce these geometric properties on triangles:

- ✓ Min. angle (deg.): 30.0
- Max. area: 1
- Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- Min. area: 1E-10
- Min. edge length: 1E-10

Failsafe stop criterion:

- ✓ Max. # triangles: 2000000

Other mesh generation parameters:

- ✓ Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

Triangular Cells: N/A
Voronoi Cells: 67

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.00471 seconds (0.00466 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

V Cell #: (none) | X: 3691.079 | Y: 13490.345

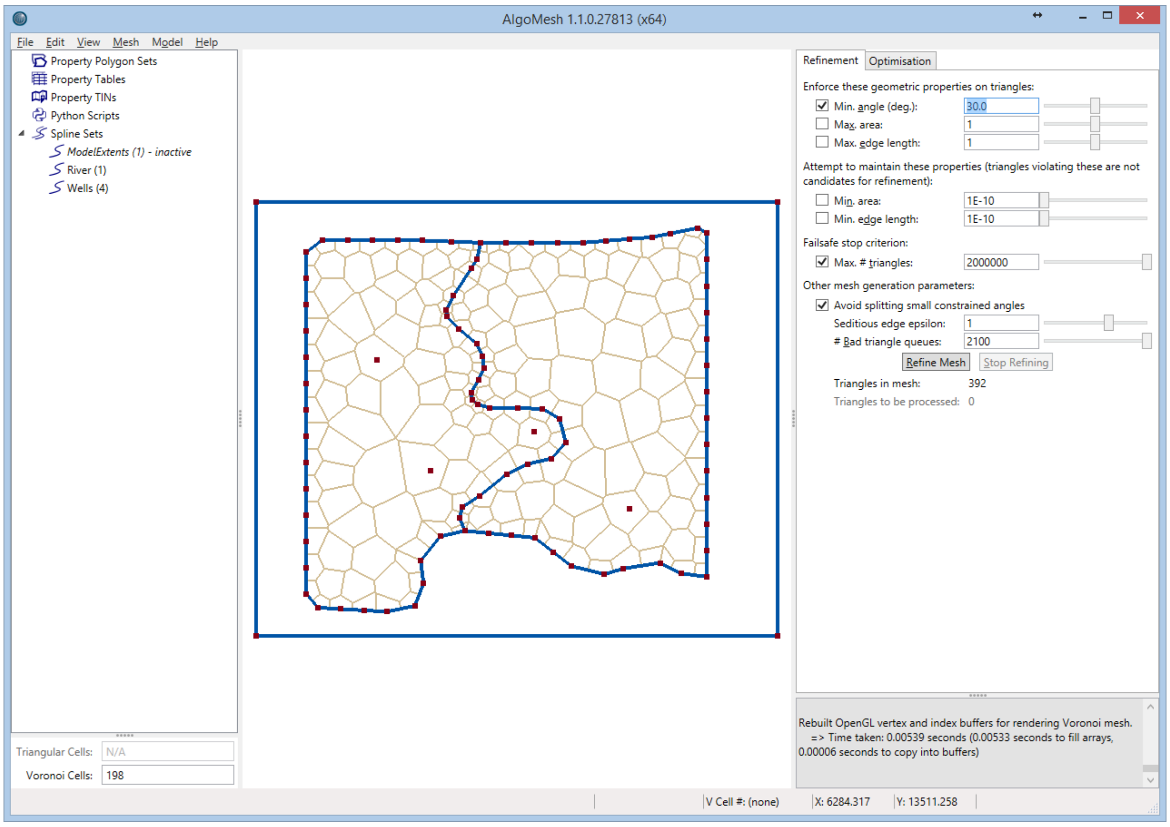
Figure 19: The Voronoi grid corresponding to the initial CDT of the mesh domain.

All of the geometry you see was initially brought into AlgoMesh from GIS shapefiles, and later saved into the AlgoMesh project (.amproj) file we just loaded. The Import Geometry from GIS how-to guide details this process.

We are now ready to generate the mesh using the Delaunay refinement method.

- On the right-hand side of the AlgoMesh window, ensure the **Refinement** tab is selected, and leave the settings at their defaults (30 degree minimum angle).
- Click the **Refine Mesh** button.
- Once the display updates to show the generated mesh, go to the **View** menu and turn off **Splines**. This hides the yellow and orange river and well features to make it easier to see the resulting mesh underneath.

Your AlgoMesh window should now look like Figure 20. Note that all of the points and line segments that were present in the river and well geometry are incorporated in the generated Voronoi grid. Points become Voronoi cell centres, and line segments form perpendicular flow paths between adjacent cells.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 392
Triangles to be processed: 0

Triangular Cells: N/A
Voronoi Cells: 198

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.00539 seconds (0.00533 seconds to fill arrays,
0.00006 seconds to copy into buffers)
```

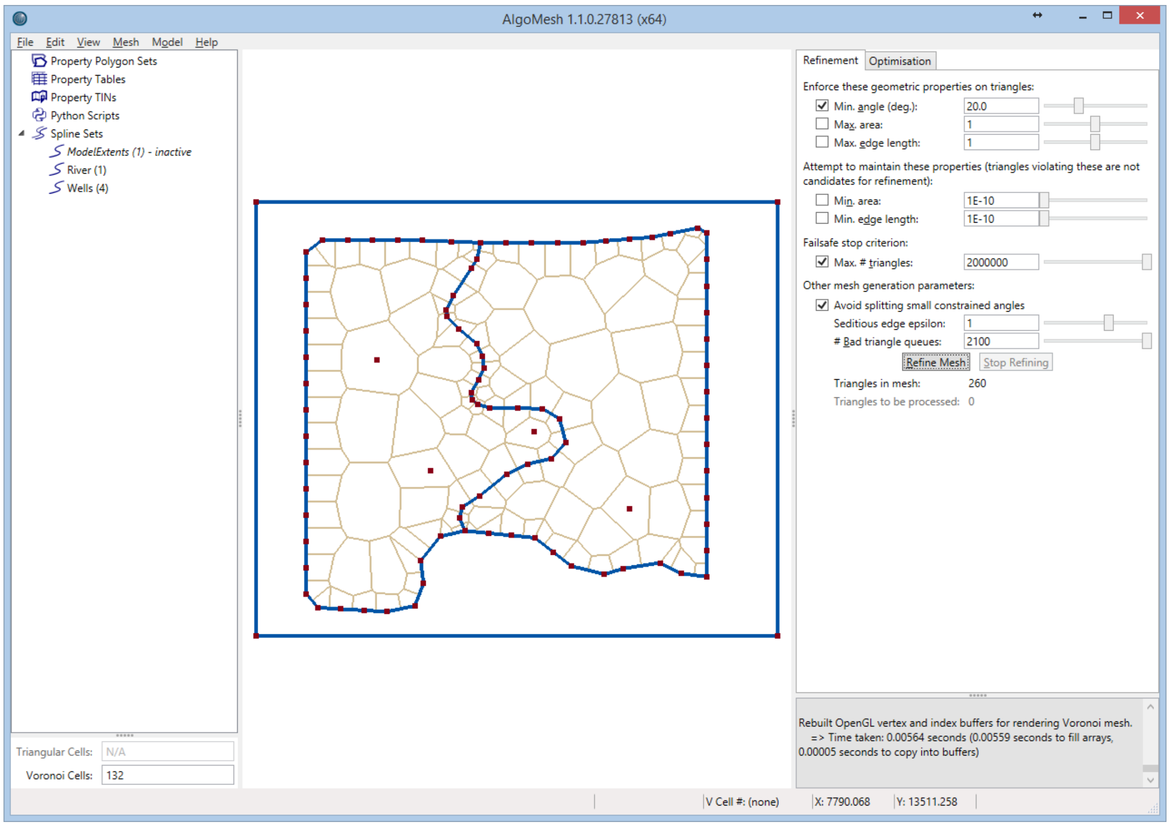
V Cell #: (none) X: 6284.317 Y: 13511.258

Figure 20: The Voronoi mesh after refinement with a 30 degree minimum angle.

The result is a complete Voronoi mesh, incorporating our input geometry and the 30 degree default minimum angle constraint on triangles. The minimum angle constraint can have a significant effect on the resulting mesh. To see this, try changing its value and regenerating the mesh.

➢ In the Refinement tab settings, change **Min. angle (deg.)** to **20.0**. Ensure its checkbox remains checked.
➢ Click **Refine Mesh**.

AlgoMesh regenerates the mesh from the initial geometry; it does not start from the existing generated mesh. The result should be similar to Figure 21. Note how the mesh now contains fewer cells, but some of them are quite elongated. These elongated cells correspond to narrow triangles (with one or more small vertex angles) in the underlying triangular grid generated by AlgoMesh.



Triangular Cells: N/A
Voronoi Cells: 132

Enforce these geometric properties on triangles:

- ✓ Min. angle (deg.): 20.0
- ☐ Max. area: 1
- ☐ Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- ☐ Min. area: 1E-10
- ☐ Min. edge length: 1E-10

Failsafe stop criterion:

- ✓ Max. # triangles: 2000000

Other mesh generation parameters:

- ✓ Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 260
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.00564 seconds (0.00559 seconds to fill arrays,
0.00005 seconds to copy into buffers)
```

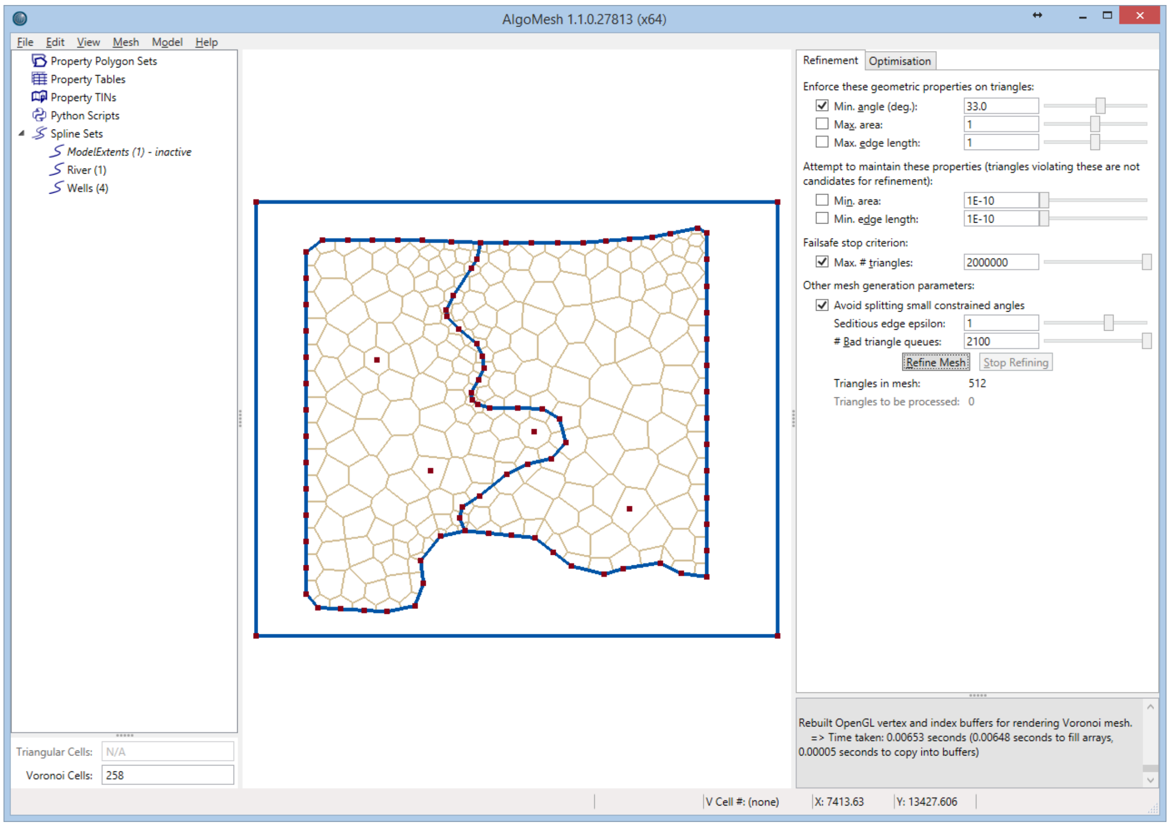
V Cell #: (none) X: 7790.068 Y: 13511.258

Figure 21: The Voronoi mesh after refinement with a 20 degree minimum angle.

Now try a higher minimum angle.

- ➢ In the Refinement tab settings, change **Min. angle (deg.)** to **33.0**. Ensure its checkbox remains checked.
- ➢ Click **Refine Mesh**.

The result should be similar to Figure 22. Note how the mesh now contains more Voronoi cells, but with many that are better shaped than the previous mesh (more uniform cell geometry). It is not recommended to increase the minimum angle too far beyond this point; with a high setting for minimum angle, the Delaunay refinement method may over-refine in some parts of the mesh, and may not even be able to complete, stopping only when the maximum number of triangles in the underlying triangular grid is reached.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 33.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 512
Triangles to be processed: 0

Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
=> Time taken: 0.00653 seconds (0.00648 seconds to fill arrays, 0.00005 seconds to copy into buffers)

Triangular Cells: N/A
Voronoi Cells: 258

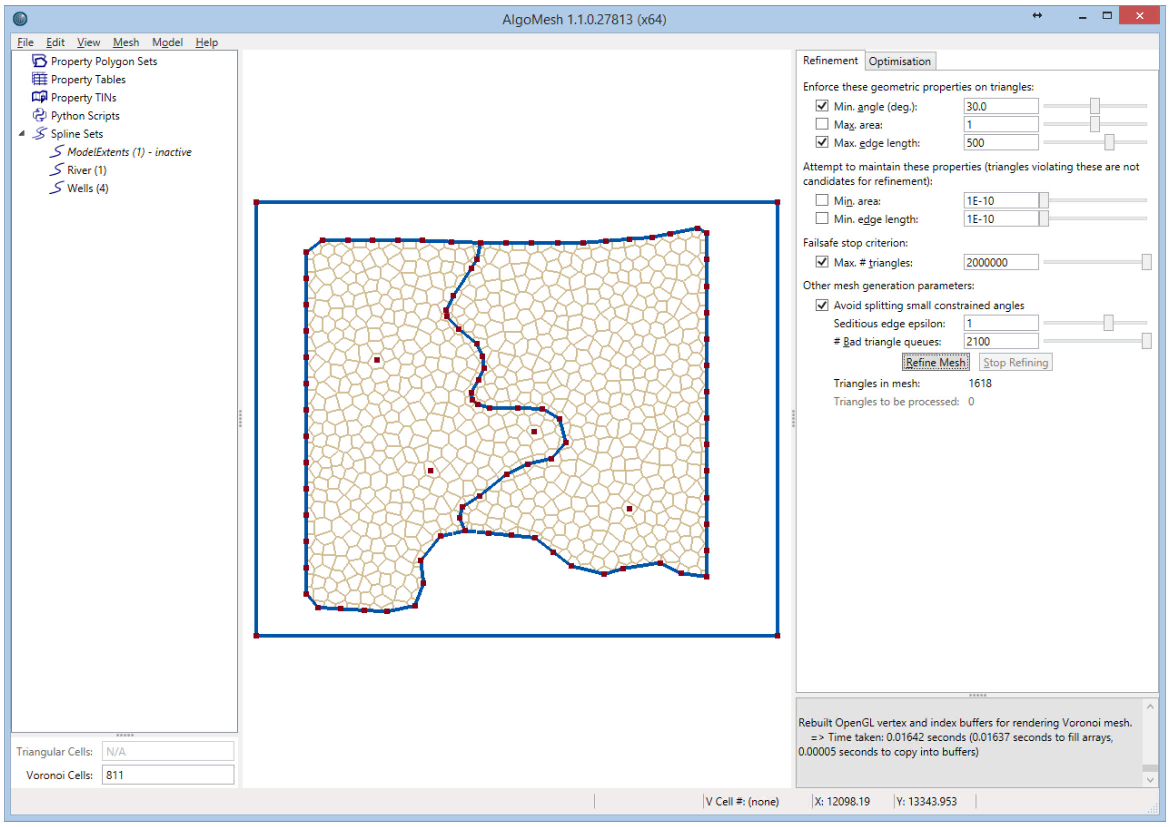
V Cell #: (none) | X: 7413.63 | Y: 13427.606

Figure 22: The Voronoi mesh after refinement with a 33 degree minimum angle.

You can also set a global maximum cell size (triangle edge length) to produce a more uniform mesh over the entire domain. A maximum edge length constraint applied to the underlying triangular grid equates to a maximum distance between cell centres of adjacent Voronoi cells, which is one way of measuring Voronoi cell size.

➢ In the Refinement tab settings, first change **Min. angle (deg.)** back to **30.0**. Ensure its checkbox remains checked.
➢ Check the **Max. edge length** checkbox, and set its value to **500**.
➢ Click **Refine Mesh**.

Figure 23 shows the resulting mesh, in which all triangles have a longest edge length of 500 or less.



Enforce these geometric properties on triangles:

☑ Min. angle (deg.): 30.0
☐ Max. area: 1
☑ Max. edge length: 500

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

☐ Min. area: 1E-10
☐ Min. edge length: 1E-10

Failsafe stop criterion:

☑ Max. # triangles: 2000000

Other mesh generation parameters:

☑ Avoid splitting small constrained angles
Seditious edge epsilon: 1
Bad triangle queues: 2100

Triangles in mesh: 1618
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.01642 seconds (0.01637 seconds to fill arrays,
0.00005 seconds to copy into buffers)
```

Triangular Cells: N/A
Voronoi Cells: 811

V Cell #: (none) | X: 12098.19 | Y: 13343.953

Figure 23: The Voronoi mesh after refinement with a maximum edge length of 500 and a 30 degree minimum angle.

The maximum area constraint acts similarly to the maximum edge length constraint - but note that the area refers to the area of each triangle in the underlying triangular grid, not the area of each Voronoi cell. The minimum area and edge length constraints may be used to stop AlgoMesh from splitting triangles in the underlying grid when they get to a certain size, but these are not recommended for use in general, as they may result in a mesh with poor-quality cell geometry.

As it typically runs very quickly, the Delaunay refinement method is useful for getting an initial idea of whether your input geometry will produce an appropriate mesh. However, higher-quality meshes may be generated using the optimisation method. The Delaunay refinement method is still typically used in this case, but it is run instead as a post-processing step after the optimisation, to eliminate the small number of poor-quality triangles that may otherwise remain. More global controls on cell sizing and quality are also available using the optimisation method, and these are detailed in the Generate a High-Quality Voronoi Mesh using Multi-Level Optimisation how-to guide.

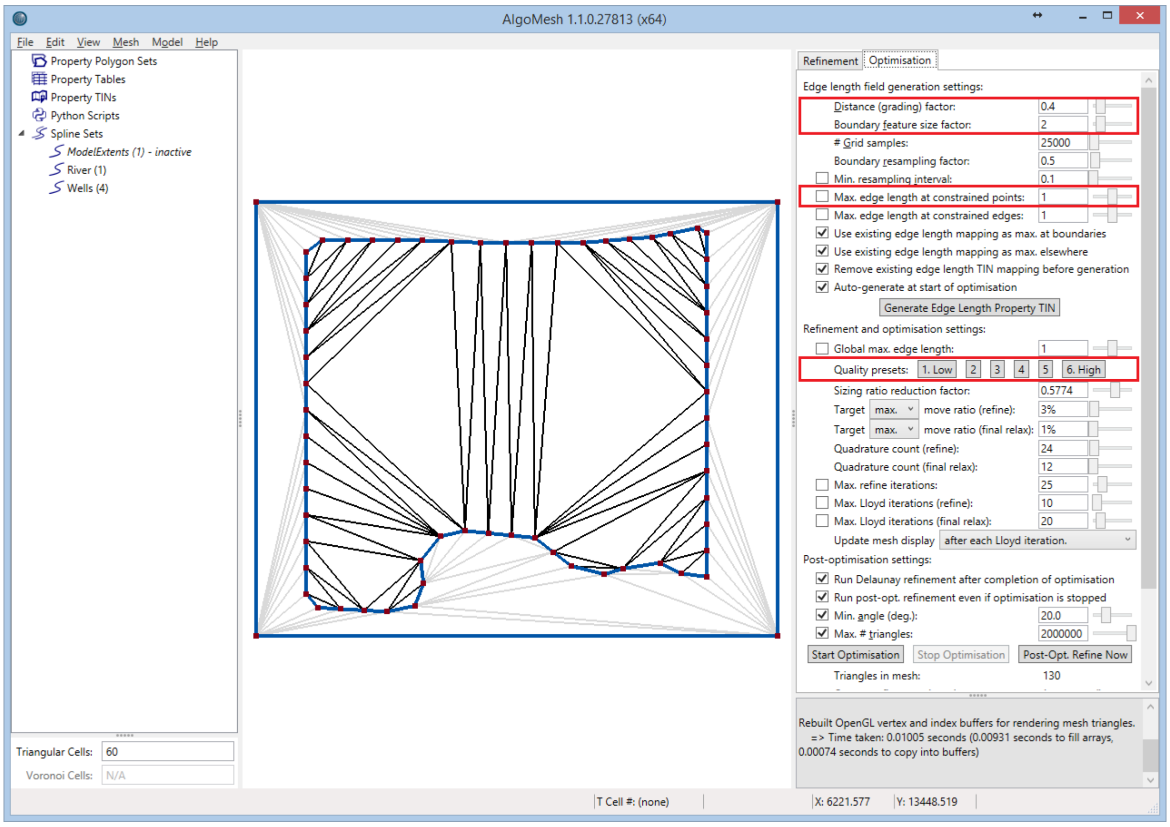
3.3 Generate a High-Quality Triangular Mesh using Multi-Level Optimisation

This section will show you how to use the multi-level optimisation mesh generation method to produce a triangular grid. You will explore the most important settings and their effect on the resulting mesh. The optimisation method normally produces much higher-quality meshes than the Delaunay refinement method alone, and is the recommended method for mesh generation in AlgoMesh. Note that there is an alternative version of this how-to section using Voronoi grids instead of triangular grids; follow whichever of these is relevant to the type of grid you want to create.

We will be starting from the same geometry that was used in the Delaunay refinement how-to guide. It is recommended that you first go through the Delaunay refinement tutorial if you have not already, as it introduces some of the basic knowledge that is assumed in this section.

Start by launching AlgoMesh and opening the project file river-mesh-domain.amproj.

➢ Go to the **File** menu and choose **Open Project**.
➢ Browse to the **Samples** folder of your AlgoMesh installation.
➢ Select **river-mesh-domain.amproj** and click **Open**.
➢ In the **View** menu, turn off **Splines** so that you have a good view of the mesh you are going to generate. You will notice the river and wells disappear from the view (Figure 24), but these will be visible when they are added into the generated mesh later.
➢ Select the **Optimisation** tab on the right-hand side of the AlgoMesh window.



Edge length field generation settings:
- Distance (grading) factor: 0.4
- Boundary feature size factor: 2
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ☐ Max. edge length at constrained points: 1
- ☐ Max. edge length at constrained edges: 1
- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☑ Auto-generate at start of optimisation

Refinement and optimisation settings:
- ☐ Global max. edge length: 1
- Quality presets: 1. Low / 2 / 3 / 4 / 5 / 6. High
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display: after each Lloyd iteration.

Post-optimisation settings:
- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped
- ☑ Min. angle (deg.): 20.0
- ☑ Max. # triangles: 2000000

Triangles in mesh: 130

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.01005 seconds (0.00931 seconds to fill arrays,
0.00074 seconds to copy into buffers)
```

Triangular Cells: 60
Voronoi Cells: N/A
T Cell #: (none)
X: 6221.577
Y: 13448.519

Figure 24: The initial geometry and default optimisation settings.

You should see a page full of settings and some buttons to control the optimisation process. Although there are many settings here, in most cases you will only need to worry about four of them - those highlighted in red in Figure 25: distance (grading) factor, boundary feature size factor, max. edge length at constrained points, and the quality preset for optimisation.

The first three of these important settings control the distribution of triangle sizes over the mesh domain. This distribution is generated as the first step of the mesh generation process, and stored in a property TIN with a variable mapped to the Mesh Cell Edge Length property.

The fourth important setting, the quality preset, sets the majority of the individual settings for the optimisation process to reasonable values, based on an overall "quality" setting from 1 (low) to 6 (high). These values in turn affect how the actual optimisation process operates, which is executed after an edge length distribution property TIN has been generated. Note that the term “quality” here is used loosely; in practice, the quality is determined mostly by the edge length distribution. A higher quality preset tends to match a certain edge length distribution better than a lower quality preset, using fewer cells, but at the cost of increased execution time. The default settings correspond to quality preset 3, which is usually a reasonable compromise between quality and speed. In a more complex mesh domain, you might start at quality preset 1 to experiment with different edge length property TIN generation settings, and then move to a higher value to produce the final mesh once you are happy with the cell size distribution. Higher quality preset values tend to produce meshes with fewer cells to match the same edge length field.

We will first run the optimisation method with all settings at their defaults, to get an idea of how it operates.

➢ Near the bottom of the **Optimisation** tab (you may need to scroll down to see this), click the **Start Optimisation** button.

The view animates as the optimisation proceeds by adding cells, then smoothing the mesh, and repeating this process until a complete mesh is generated. It may take a minute or so to complete, depending on your computer hardware, but as it proceeds you should see something similar to the sequence shown in Figure 26. Once the process is complete, the Stop Optimisation button and the informational labels at the bottom of the Optimisation tab will turn grey (disabled).

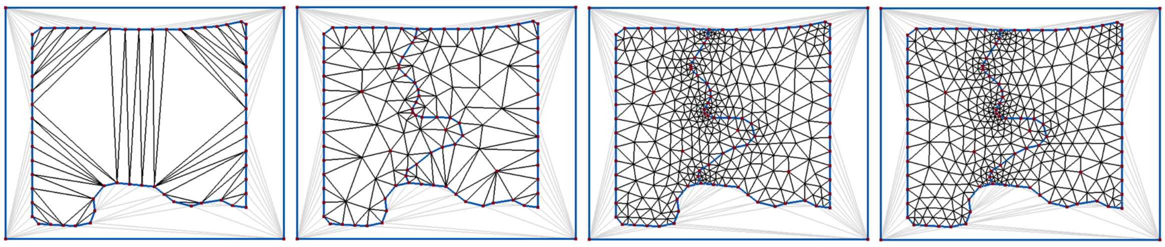


Figure 26: The mesh is progressively augmented with new cells and smoothed until a complete mesh is produced which satisfies the generated Mesh Cell Edge Length field.

Refining the Mesh around the Wells

With the default settings, the resulting mesh looks reasonable, but does not have much refinement around the wells or river. To firstly address the wells, we will set maximum edge length limits on constrained points in the mesh domain.

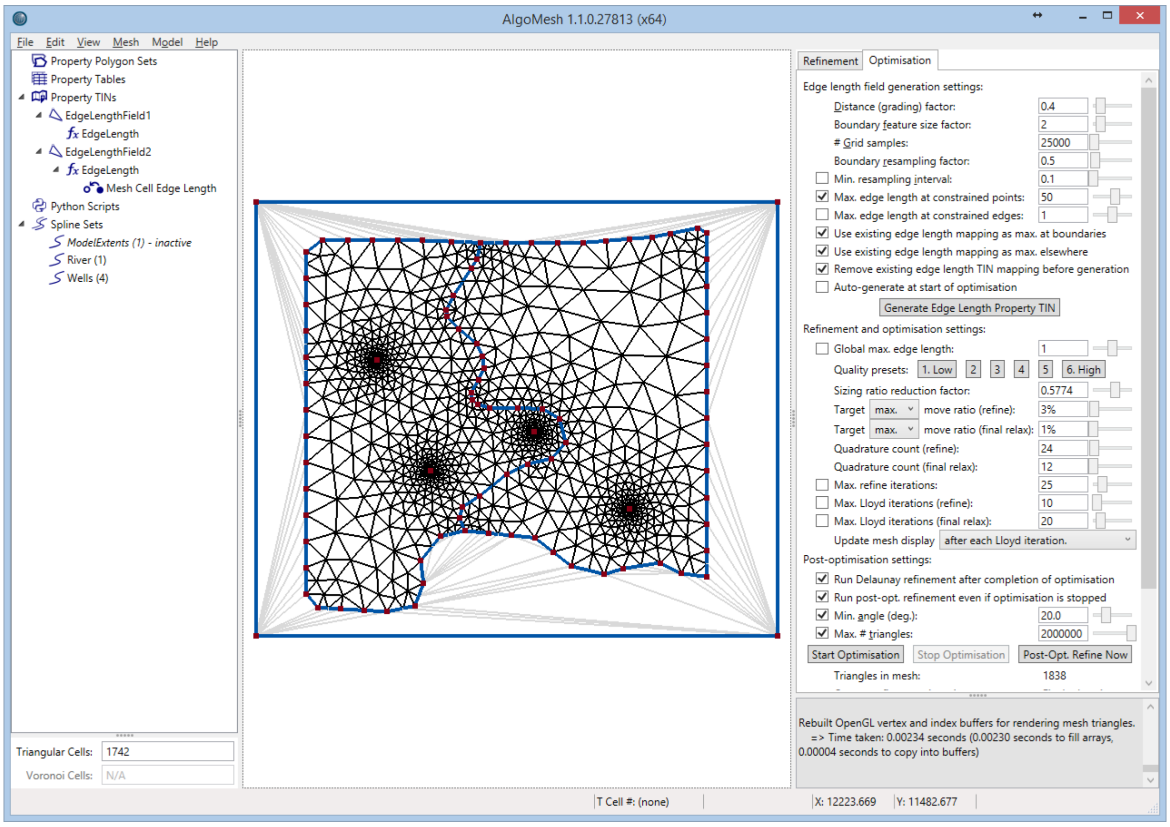
➢ Tick the box next to **Max. edge length at constrained points**.
➢ Set **Max. edge length at constrained points** to **50**. This will decrease the size of cells in the vicinity of the wells (constrained points in the mesh domain that are not part of any constrained edges).

When you modify the settings, note that "Auto-generate at start of optimisation" is automatically switched back on. This ensures that a new edge length property TIN will be generated and applied the next time optimisation is run. Once you run the optimisation, this flag will be turned off, allowing you to change only the optimisation settings and re-run if desired, without having to regenerate the edge length property TIN.

We can now run the optimisation again to produce a new mesh with the updated edge length setting.

➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

You should now have a mesh with heavy refinement around the four wells in the model, as shown in Figure 27.



Triangular Cells: 1742
Voronoi Cells: N/A

Edge length field generation settings:
- Distance (grading) factor: 0.4
- Boundary feature size factor: 2
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ☑ Max. edge length at constrained points: 50
- ☐ Max. edge length at constrained edges: 1
- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☐ Auto-generate at start of optimisation

Generate Edge Length Property TIN

Refinement and optimisation settings:
- ☐ Global max. edge length: 1
- Quality presets: 1. Low 2 3 4 5 6. High
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display: after each Lloyd iteration.

Post-optimisation settings:
- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped
- ☑ Min. angle (deg.): 20.0
- ☑ Max. # triangles: 2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 1838

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.00234 seconds (0.00230 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

T Cell #: (none) | X: 12223.669 | Y: 11482.677

Figure 27: Setting a maximum edge length at constrained points allows us to refine to much smaller cell sizes around all wells in the model.

Note that this is a global setting which affects all constrained points in the mesh domain. It is also possible with AlgoMesh to selectively refine only around some of these wells, or to choose different cell sizes for refinement at each, by using property polygons mapped to Mesh Cell Edge Length. The Use Property Polygons to Control Mesh Cell Sizing how-to guide explores this in more detail.

Refining the Mesh around the River

We would also like to increase the level of refinement around the river feature running through the centre of the model. We could use the **Max. edge length at constrained edges** setting to do this, but as this is a global setting, it would also affect cell sizing along the boundary of the model, which we would normally prefer to avoid, as it would add many cells to our model where they are probably not needed.

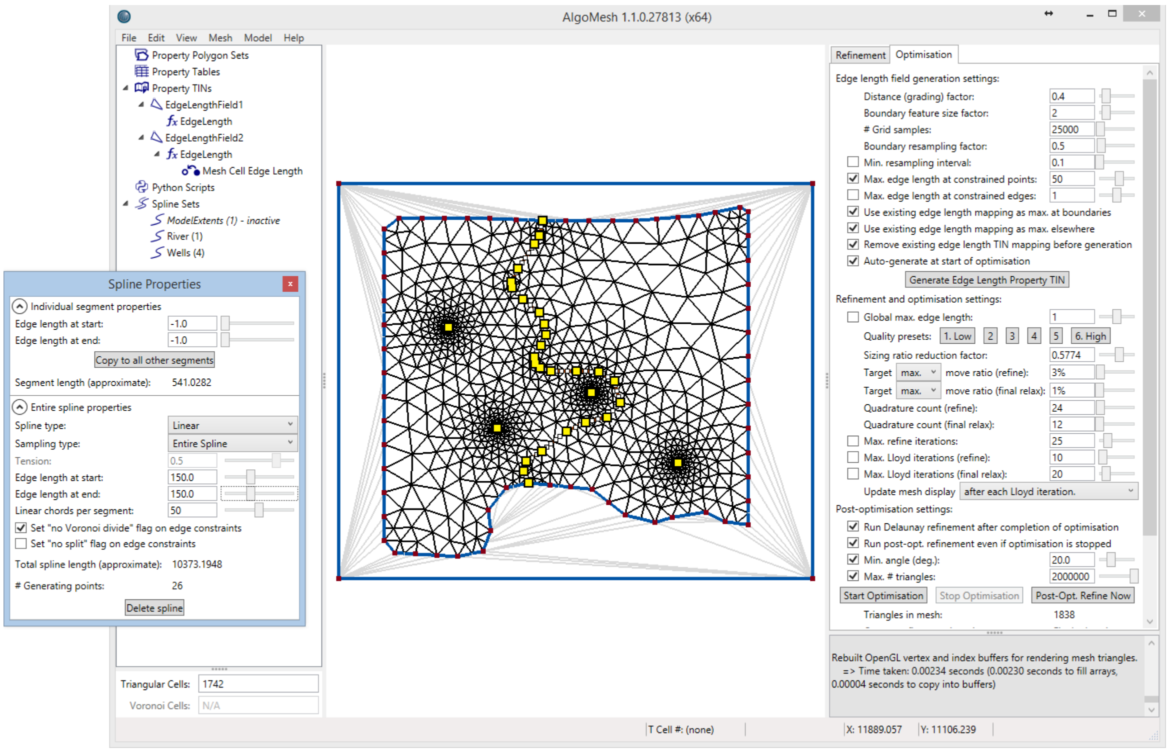
Instead, we will tell AlgoMesh to resample the polyline representing the river by inserting points at regular intervals along its length. As each of these points becomes a vertex in the final mesh, this will naturally increase the number of cells in the vicinity of the river.

➢ In the **View** menu, turn **Splines** back on so that you can see the orange river spline (polyline).
➢ In the **Edit** menu, select **Edit Splines** mode.
➢ **Move your mouse cursor over any part of the river spline** - it should turn a lighter orange

colour - and **click the right button** of the mouse. This will open the **Spline Properties** window.

- ➢ In the **Spline Properties** window, under **Entire spline properties**, change **Sampling type** to **Entire Spline**.
- ➢ Also under **Entire spline properties**, change **Edge length at start** to **150** and **Edge length at end** to **150**.

Your Spline Properties window and the river spline in the view should now look like those shown in Figure 28. Note that a series of small white squares appears along the river spline in the mesh view. These are the locations at which AlgoMesh will insert points immediately prior to the mesh generation process starting.



Spline Properties

Individual segment properties

Edge length at start:	-1.0
Edge length at end:	-1.0
Segment length (approximate):	541.0282

Entire spline properties

Spline type:	Linear
Sampling type:	Entire Spline
Tension:	0.5
Edge length at start:	150.0
Edge length at end:	150.0
Linear chords per segment:	50
☑ Set "no Voronoi divide" flag on edge constraints	
☐ Set "no split" flag on edge constraints	
Total spline length (approximate):	10373.1948
# Generating points:	26

Triangular Cells:	1742
Voronoi Cells:	N/A

Edge length field generation settings:

Distance (grading) factor:	0.4
Boundary feature size factor:	2
# Grid samples:	25000
Boundary resampling factor:	0.5
☐ Min. resampling interval:	0.1
☑ Max. edge length at constrained points:	50
☐ Max. edge length at constrained edges:	1
☑ Use existing edge length mapping as max. at boundaries	
☑ Use existing edge length mapping as max. elsewhere	
☑ Remove existing edge length TIN mapping before generation	
☑ Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐ Global max. edge length:	1
Quality presets:	1. Low, 2, 3, 4, 5, 6. High
Sizing ratio reduction factor:	0.5774
Target max. move ratio (refine):	3%
Target max. move ratio (final relax):	1%
Quadrature count (refine):	24
Quadrature count (final relax):	12
☐ Max. refine iterations:	25
☐ Max. Lloyd iterations (refine):	10
☐ Max. Lloyd iterations (final relax):	20
Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

☑ Run Delaunay refinement after completion of optimisation	
☑ Run post-opt. refinement even if optimisation is stopped	
☑ Min. angle (deg.):	20.0
☑ Max. # triangles:	2000000
Triangles in mesh:	1838

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.00234 seconds (0.00230 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

T Cell #: (none) X: 11889.057 Y: 11106.239

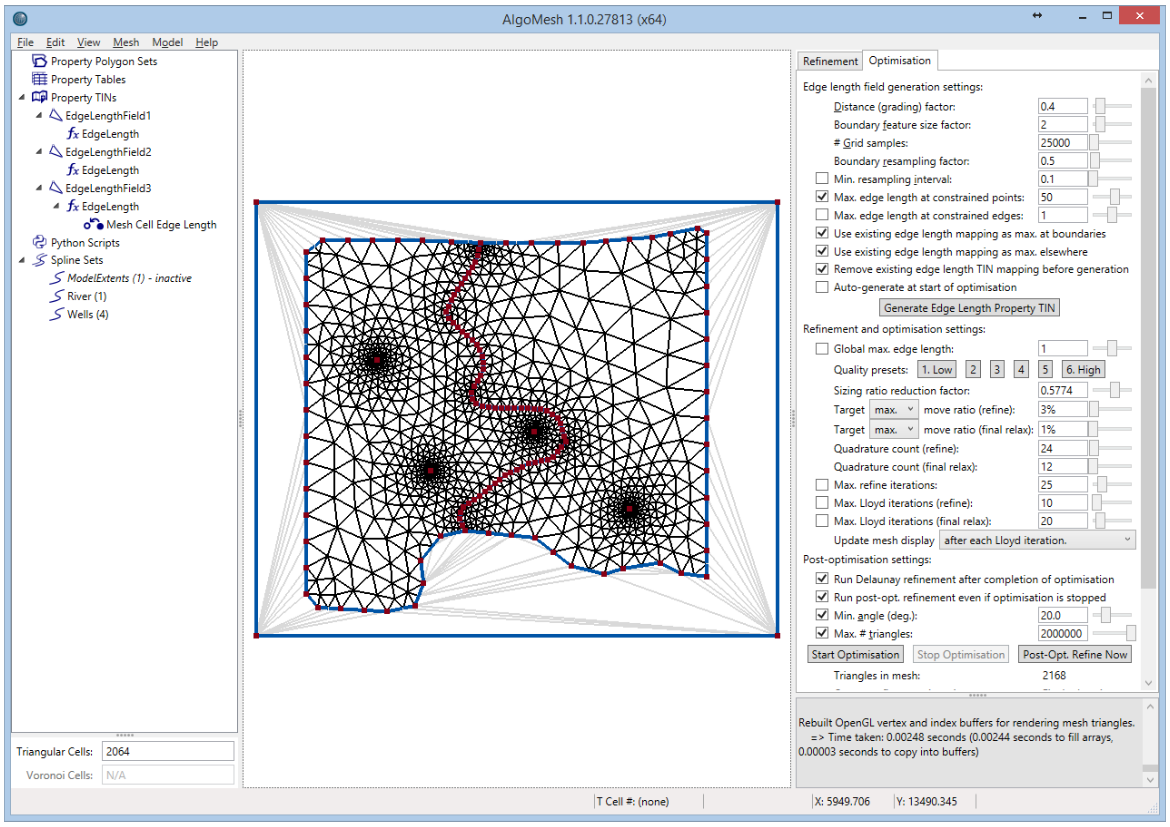
Figure 28: Setting the properties of the river spline allows resampling at a finer point spacing to control cell sizing along a linear feature.

There are a number of other spline properties you can use to get more control over linear features which we do not detail here, including the ability to represent them as smooth spline curves. The Manually Digitise and Edit Splines how-to guide explores these properties further.

We are now ready to regenerate the mesh to incorporate the resampled river polyline.

- ➢ **Click the red X** in the top-right corner of the **Spline Properties** window to close it.
- ➢ Ensure that **Auto-generate at start of optimisation** under **Edge length field generation settings** is turned on.
- ➢ Click the **Start Optimisation** button and wait for mesh generation to complete.
- ➢ In the **View** menu, turn **Splines** off once more to get a better look at the generated mesh.

You should end up with something like the mesh shown in Figure 29. Notice that the cells along the river are now somewhat smaller and more uniform in size, due to the uniform point spacing along the river polyline.



Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	2
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
✓	Max. edge length at constrained points:	50
☐	Max. edge length at constrained edges:	1
✓	Use existing edge length mapping as max. at boundaries	
✓	Use existing edge length mapping as max. elsewhere	
✓	Remove existing edge length TIN mapping before generation	
☐	Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

✓	Run Delaunay refinement after completion of optimisation	
✓	Run post-opt. refinement even if optimisation is stopped	
✓	Min. angle (deg.):	20.0
✓	Max. # triangles:	2000000
	Triangles in mesh:	2168

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.00248 seconds (0.00244 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

Triangular Cells: 2064
Voronoi Cells: N/A

T Cell #: (none) X: 5949.706 Y: 13490.345

Figure 29: Resampling the river polyline at a point spacing of 150 produces a finer, more uniform array of cells along the river in the generated mesh.

For now, we will settle for this level of refinement. Note that we could extend the refinement further afield of the river by using an appropriate property polygon mapped to Mesh Cell Edge Length; this process is described in the Use Property Polygons to Control Mesh Cell Sizing how-to guide.

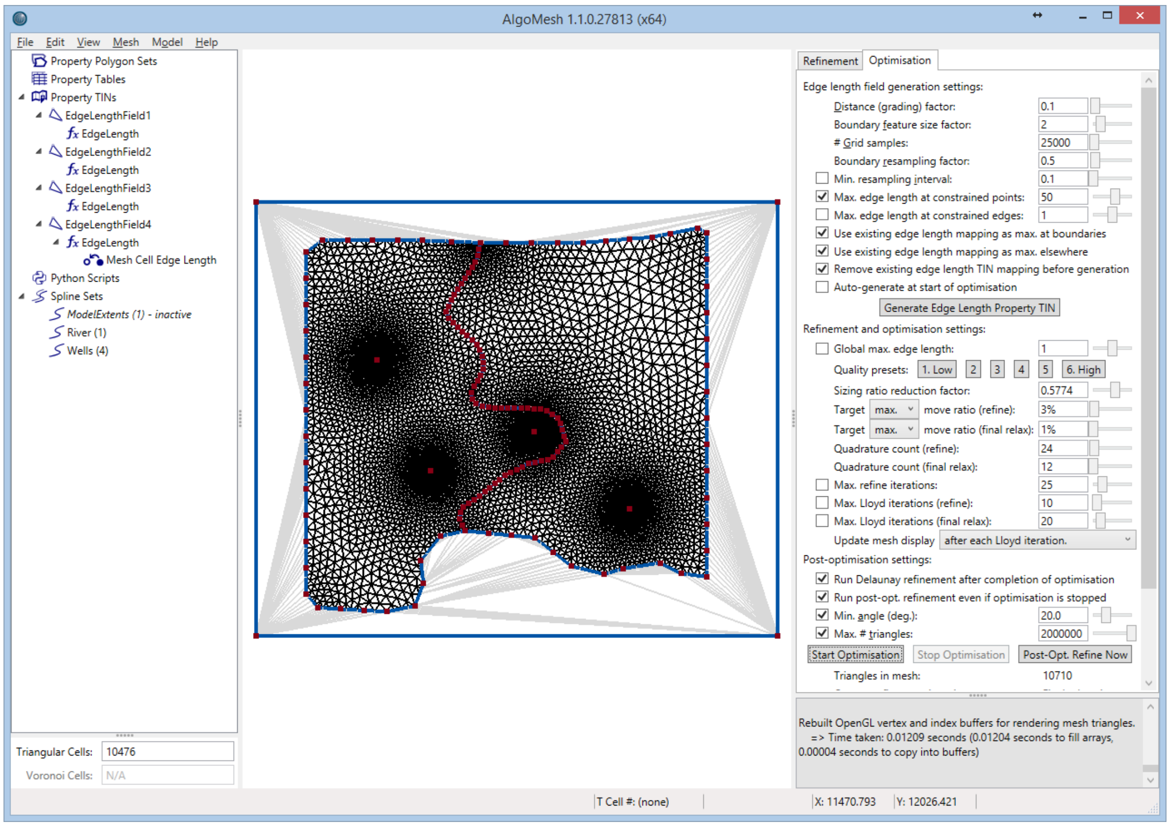
We will now look at the effect of varying some of the global edge length field generation and optimisation settings.

Altering the Distance (Grading) Factor

The distance (grading) factor controls how quickly cells grow in size with respect to their distance from the boundaries, where the smallest cells are. A low distance factor will result in a more uniform cell size over the mesh domain, and more cells overall.

- ➢ Change **Distance (grading) factor** to **0.1**.
- ➢ Click the **Start Optimisation** button and wait for mesh generation to complete. The process will take a little longer this time, as more cells are being generated.

Figure 30 shows the resulting mesh, which has around five times as many triangles, and a much more gradual change in cell size moving away from the boundaries.



Triangular Cells: 10476
Voronoi Cells: N/A

Edge length field generation settings:
Distance (grading) factor: 0.1
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☑ Max. edge length at constrained points: 50
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation

Refinement and optimisation settings:
☐ Global max. edge length: 1
Quality presets:
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000

Triangles in mesh: 10710

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.01209 seconds (0.01204 seconds to fill arrays, 0.00004 seconds to copy into buffers)

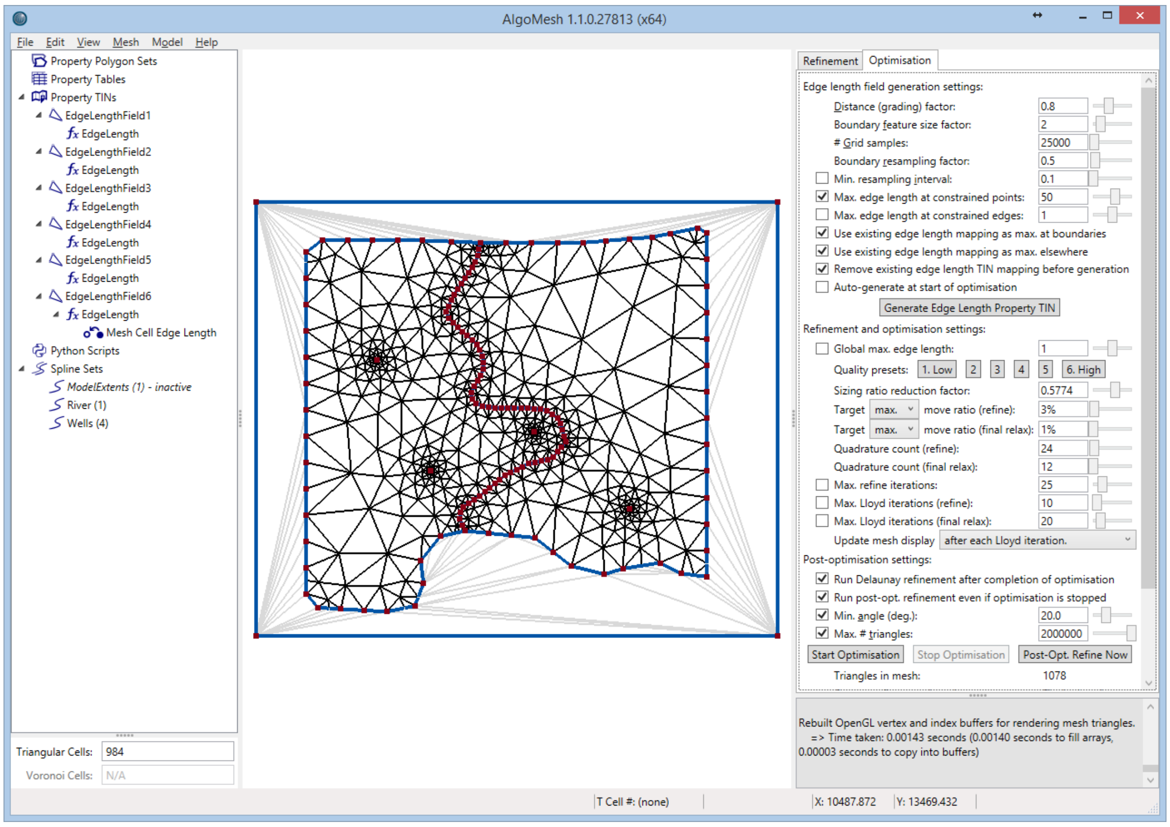
T Cell #: (none) X: 11470.793 Y: 12026.421

Figure 30: A low distance (grading) factor results in more cells of a more uniform size.

In contrast, a high distance factor will create much larger cells in empty areas of the mesh domain, with a rapid grading in size from large to small for the cells approaching the boundaries.

➢ Change **Distance (grading) factor** to **0.8**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 31 shows the resulting mesh, which has fewer cells, but more abrupt changes in cell size. In general, increasing the distance (grading) factor can help to reduce the number of cells in a mesh, but care should be taken to avoid going too far; high values may lead to poor-quality cells close to the boundaries, as they are "stretched" to try to meet the rapid changes in desired cell size. Note in particular that the triangles along the river are much narrower than in the mesh produced with a low distance factor.



Triangular Cells: 984
Voronoi Cells: N/A
T Cell #: (none)
X: 10487.872
Y: 13469.432

Edge length field generation settings:

- Distance (grading) factor: 0.8
- Boundary feature size factor: 2
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ☑ Max. edge length at constrained points: 50
- ☐ Max. edge length at constrained edges: 1
- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☐ Auto-generate at start of optimisation

Refinement and optimisation settings:

- ☐ Global max. edge length: 1
- Quality presets: 1. Low 2 3 4 5 6. High
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:

- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped
- ☑ Min. angle (deg.): 20.0
- ☑ Max. # triangles: 2000000

Triangles in mesh: 1078

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
    => Time taken: 0.00143 seconds (0.00140 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

Figure 31: A high distance (grading) factor results in fewer cells with more sudden changes in size moving from boundaries to empty regions of the mesh domain.

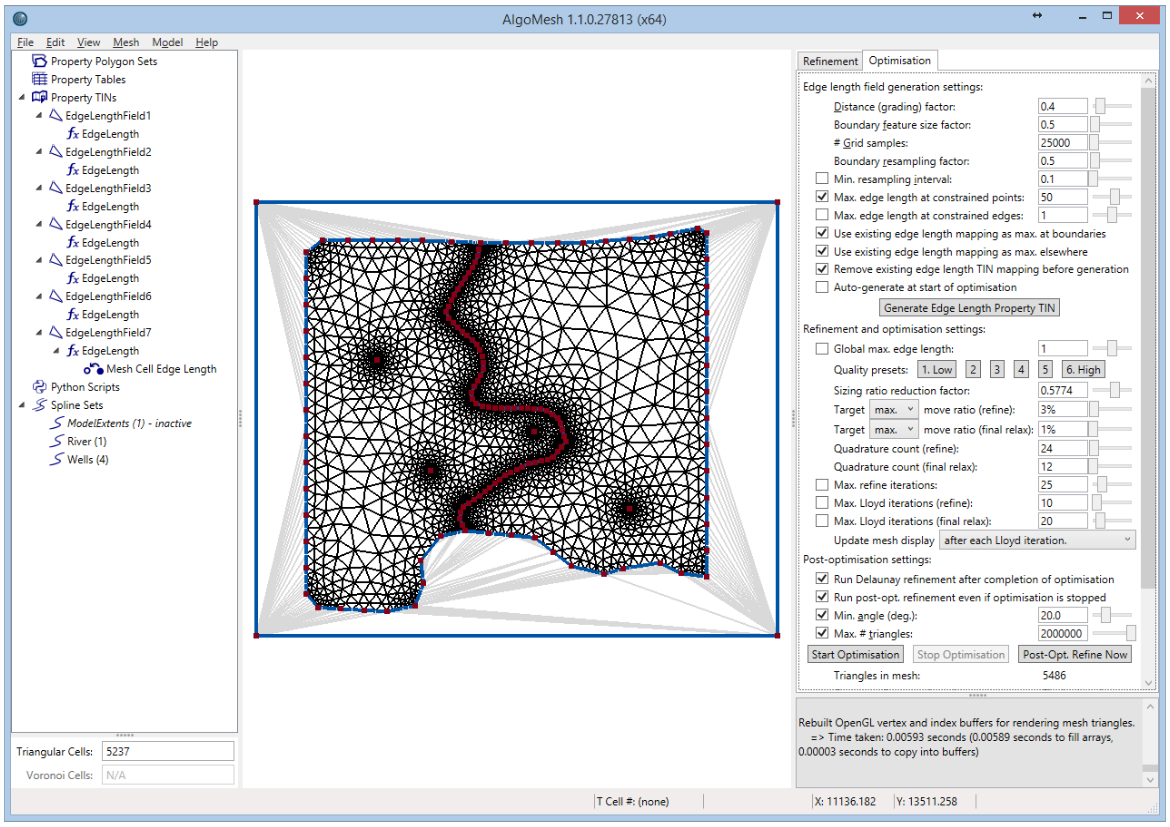
Altering the Boundary Feature Size Factor

The boundary feature size factor controls how many cells are generated along the boundaries of the mesh domain (including the outer extents of the model and any interior linear features, like the river in our model). The sizing of cells along the boundaries is always dependent on a combination of the spacing of input points and the distance from one boundary to other nearby boundaries, but the boundary feature size factor acts as a multiplier on this.

A low boundary feature size factor will have a similar effect to specifying a global maximum edge length at constrained edges, but will allow larger cells along parts of a boundary that are a long way from other boundaries.

➢ Change **Distance (grading) factor** back to the default of **0.4**.
➢ Change **Boundary feature size factor** to **0.5**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 32 shows the resulting mesh. Note the heavier refinement along the boundaries in general, which is exaggerated further along the river where points are spaced more closely together, and to a lesser degree at corners of the model extents, where the boundary lines are somewhat closer to one another. Areas near tight angles are refined much more heavily.



Triangular Cells: 5237
Voronoi Cells: N/A

Edge length field generation settings:
- Distance (grading) factor: 0.4
- Boundary feature size factor: 0.5
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- [ ] Min. resampling interval: 0.1
- [✓] Max. edge length at constrained points: 50
- [ ] Max. edge length at constrained edges: 1
- [✓] Use existing edge length mapping as max. at boundaries
- [✓] Use existing edge length mapping as max. elsewhere
- [✓] Remove existing edge length TIN mapping before generation
- [ ] Auto-generate at start of optimisation

Refinement and optimisation settings:
- [ ] Global max. edge length: 1
- Quality presets: 1. Low 2 3 4 5 6. High
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- [ ] Max. refine iterations: 25
- [ ] Max. Lloyd iterations (refine): 10
- [ ] Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:
- [✓] Run Delaunay refinement after completion of optimisation
- [✓] Run post-opt. refinement even if optimisation is stopped
- [✓] Min. angle (deg.): 20.0
- [✓] Max. # triangles: 2000000

Triangles in mesh: 5486

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00593 seconds (0.00589 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

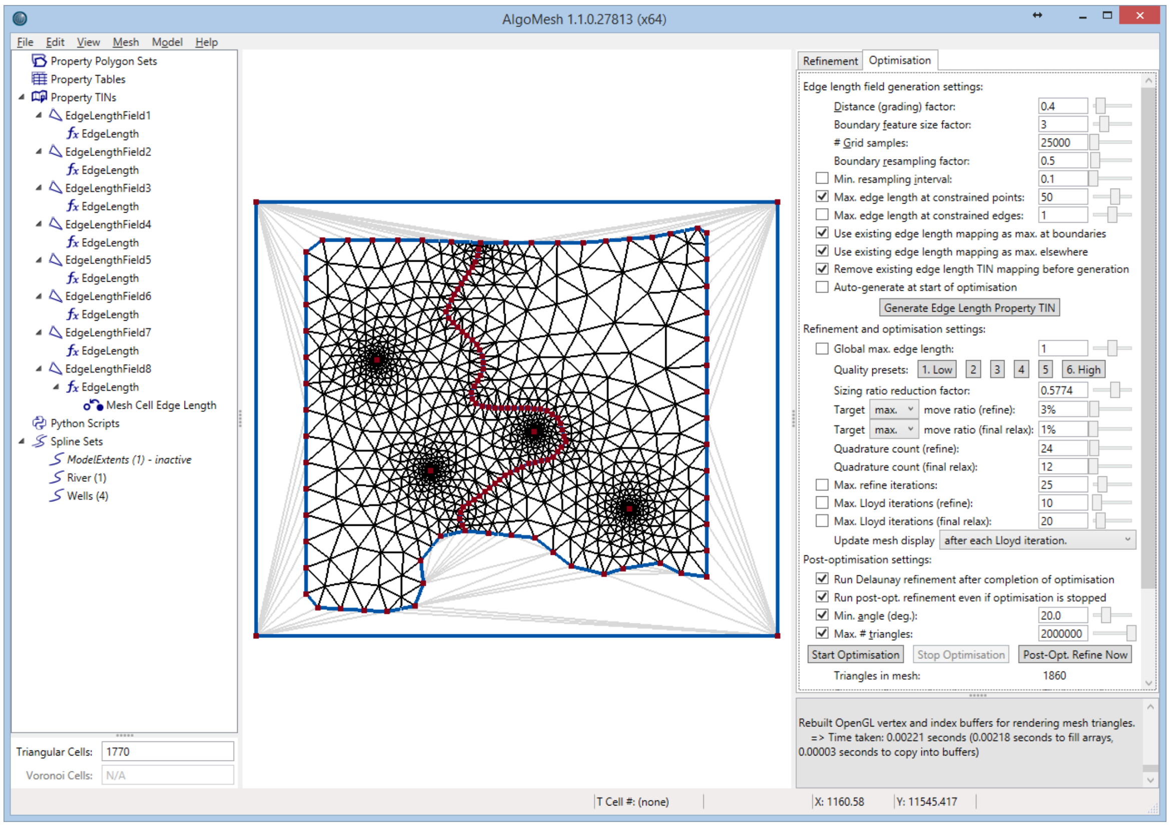
T Cell #: (none) | X: 11136.182 | Y: 13511.258

Figure 32: A low boundary feature size factor results in more cells being generated along the boundaries, without specifying a fixed upper limit on cell size.

In contrast, a high boundary feature size factor will increase the relative size of cells along boundaries, producing fewer, larger cells.

➢ Change **Boundary feature size factor** to **3**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 33 shows the result of using a higher boundary feature size factor. Triangles along the river are further elongated and there is less refinement at tight angles.



Triangular Cells:	1770
Voronoi Cells:	N/A

Edge length field generation settings:

Distance (grading) factor:	0.4
Boundary feature size factor:	3
# Grid samples:	25000
Boundary resampling factor:	0.5
☐ Min. resampling interval:	0.1
☑ Max. edge length at constrained points:	50
☐ Max. edge length at constrained edges:	1

- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☐ Auto-generate at start of optimisation

Refinement and optimisation settings:

☐ Global max. edge length:	1
Sizing ratio reduction factor:	0.5774
Target max. move ratio (refine):	3%
Target max. move ratio (final relax):	1%
Quadrature count (refine):	24
Quadrature count (final relax):	12
☐ Max. refine iterations:	25
☐ Max. Lloyd iterations (refine):	10
☐ Max. Lloyd iterations (final relax):	20
Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped

☑ Min. angle (deg.):	20.0
☑ Max. # triangles:	2000000
Triangles in mesh:	1860

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00221 seconds (0.00218 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

X: 1160.58 Y: 11545.417

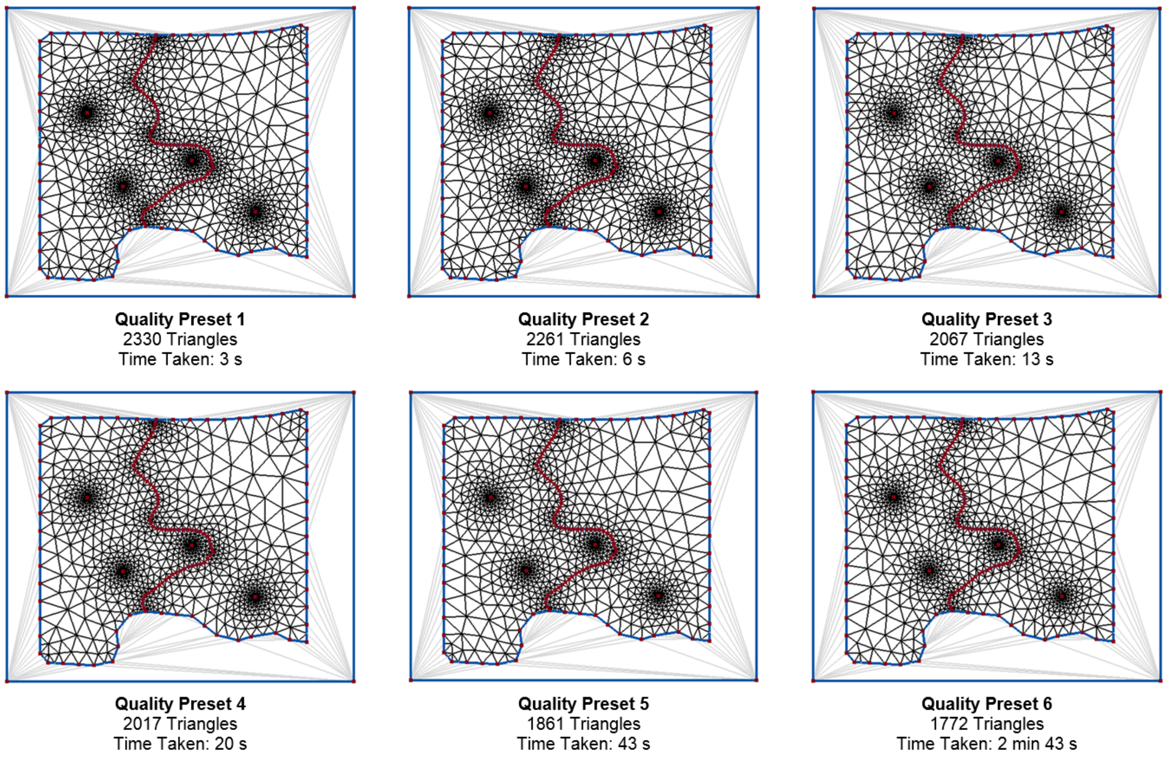
Figure 33: A high boundary feature size factor produces fewer cells of larger size along boundaries.

Altering the Optimisation Quality Preset

Once you are happy with the edge length field generation settings, you may wish to experiment with the quality preset levels to control the optimisation process itself. So far, we have been using the default quality preset 3 to generate all of our meshes. Clicking any of the quality preset buttons changes the five individual settings below the quality preset line. Note that you can also change these five settings independently, but if you click one of the preset levels again, your changes will be overwritten with the preset values.

More information on each of the individual settings and their effect can be found in the Meshing Control File (.mcf) file format reference. In general, however, choosing a preset between 1 and 6 is usually sufficient.

Figure 34 shows meshes generated for the same model using each of the quality preset levels. The differences between levels are somewhat subtle in this example, but can become significant when the mesh domain becomes more complex.



Quality Preset 1
2330 Triangles
Time Taken: 3 s

Quality Preset 2
2261 Triangles
Time Taken: 6 s

Quality Preset 3
2067 Triangles
Time Taken: 13 s

Quality Preset 4
2017 Triangles
Time Taken: 20 s

Quality Preset 5
1861 Triangles
Time Taken: 43 s

Quality Preset 6
1772 Triangles
Time Taken: 2 min 43 s

Figure 34: Triangular meshes generated using each of the optimisation quality preset levels.

Finishing Up

You may have noticed a number of "EdgeLengthField" property TINs building up with each mesh generation in the project explorer on the left-hand side of the AlgoMesh window. These are the edge length fields generated by the optimisation process. They may be removed once mesh generation is complete to save a bit of memory.

- **Right-click** on the **Property TINs** header.
- Select **Remove All Property TINs** from the context menu that appears. This will remove all of the edge length field TINs. Alternatively, you could remove only some of the TINs by right-clicking on each TIN you want to remove and choosing Remove from the menu.

Once you are happy with your generated mesh, you may want to save it so you can retrieve it later.

- Go to the **File** menu and choose **Save Project**.
- **Browse to a location** to which you have write access, such as your Documents folder.
- **Type a name** for the project file to be saved, e.g. **FinalOptMesh.amproj** and click **Save**.

AlgoMesh will save the mesh and all of the settings from the user interface to the project file. You can bring it all back at a later date by going to the **File** menu, choosing **Open Project** and opening your saved file.

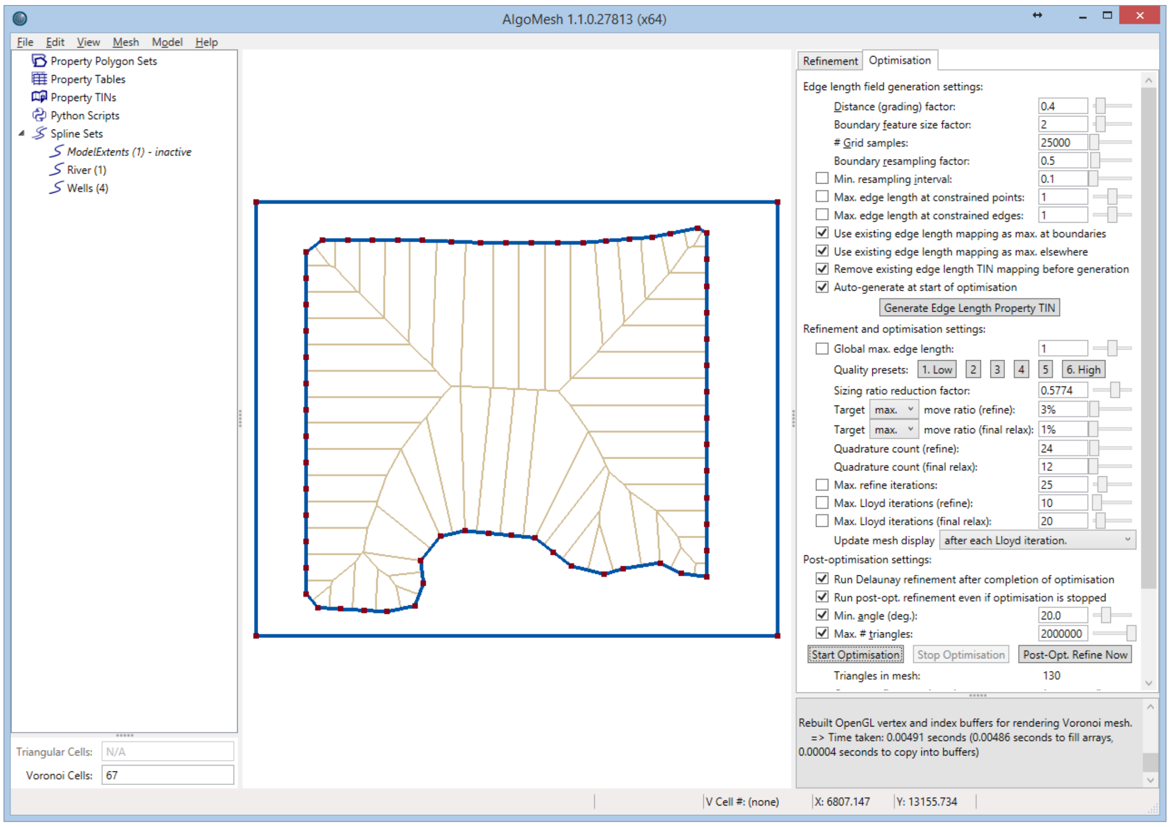
3.4 Generate a High-Quality Voronoi Mesh using Multi-Level Optimisation

This section will show you how to use the multi-level optimisation mesh generation method to produce a Voronoi grid. You will explore the most important settings and their effect on the resulting mesh. The optimisation method normally produces much higher-quality meshes than the Delaunay refinement method alone, and is the recommended method for mesh generation in AlgoMesh. Note that there is an alternative version of this how-to section using triangular grids instead of Voronoi grids; follow whichever of these is relevant to the type of grid you want to create.

We will be starting from the same geometry that was used in the Delaunay refinement how-to guide. It is recommended that you first go through the Delaunay refinement tutorial if you have not already, as it introduces some of the basic knowledge that is assumed in this section.

Start by launching AlgoMesh and opening the project file river-mesh-domain.amproj.

- ➢ Go to the **File** menu and choose **Open Project**.
- ➢ Browse to the **Samples** folder of your AlgoMesh installation.
- ➢ Select **river-mesh-domain.amproj** and click **Open**.
- ➢ In the **View** menu, turn off **Triangles**.
- ➢ In the **View** menu, turn on **Voronoi Cells**.
- ➢ In the **View** menu, turn off **Splines** so that you have a good view of the mesh you are going to generate. You will notice the river and wells disappear from the view (Figure 35), but these will be visible when they are added into the generated mesh later.
- ➢ Select the **Optimisation** tab on the right-hand side of the AlgoMesh window.



Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	2
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
☐	Max. edge length at constrained points:	1
☐	Max. edge length at constrained edges:	1
✓	Use existing edge length mapping as max. at boundaries	
✓	Use existing edge length mapping as max. elsewhere	
✓	Remove existing edge length TIN mapping before generation	
✓	Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

✓	Run Delaunay refinement after completion of optimisation	
✓	Run post-opt. refinement even if optimisation is stopped	
✓	Min. angle (deg.):	20.0
✓	Max. # triangles:	2000000
	Triangles in mesh:	130

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.00491 seconds (0.00486 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

Triangular Cells: N/A
Voronoi Cells: 67

V Cell #: (none) | X: 6807.147 | Y: 13155.734

Figure 35: The initial geometry and default optimisation settings.

You should see a page full of settings and some buttons to control the optimisation process. Although there are many settings here, in most cases you will only need to worry about four of them - those highlighted in red in Figure 36: distance (grading) factor, boundary feature size factor, max. edge length at constrained points, and the quality preset for optimisation.

The first three of these important settings control the distribution of cell sizes over the mesh domain. This distribution is generated as the first step of the mesh generation process, and stored in a property TIN with a variable mapped to the Mesh Cell Edge Length property.

The fourth important setting, the quality preset, sets the majority of the individual settings for the optimisation process to reasonable values, based on an overall "quality" setting from 1 (low) to 6 (high). These values in turn affect how the actual optimisation process operates, which is executed after an edge length distribution property TIN has been generated. Note that the term “quality” here is used loosely; in practice, the quality is determined mostly by the edge length distribution. A higher quality preset tends to match a certain edge length distribution better than a lower quality preset, using fewer cells, but at the cost of increased execution time. The default settings correspond to quality preset 3, which is usually a reasonable compromise between quality and speed. In a more complex mesh domain, you might start at quality preset 1 to experiment with different edge length property TIN generation settings, and then move to a higher value to produce the final mesh once you are happy with the cell size distribution. Higher quality preset values tend to produce meshes with fewer cells to match the same edge length field.

We will first run the optimisation method with all settings at their defaults, to get an idea of how it operates.

➢ Near the bottom of the **Optimisation** tab (you may need to scroll down to see this), click the **Start Optimisation** button.

The view animates as the optimisation proceeds by adding cells, then smoothing the mesh, and repeating this process until a complete mesh is generated. It may take a minute or so to complete, depending on your computer hardware, but as it proceeds you should see something similar to the sequence shown in Figure 37. Once the process is complete, the Stop Optimisation button and the informational labels at the bottom of the Optimisation tab will turn grey (disabled).

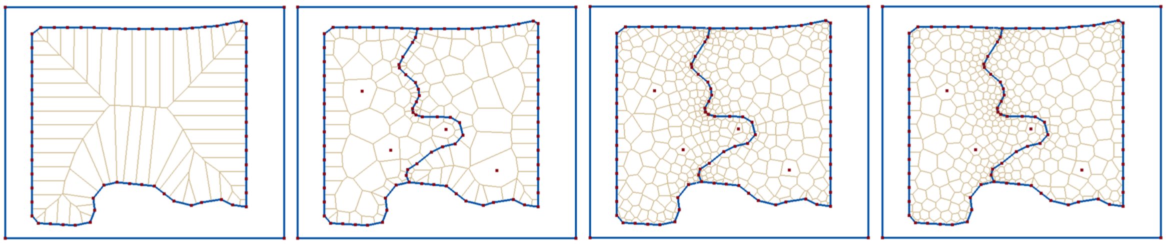


Figure 37: The mesh is progressively augmented with new cells and smoothed until a complete mesh is produced which satisfies the generated Mesh Cell Edge Length field.

Refining the Mesh around the Wells

With the default settings, the resulting mesh looks reasonable, but does not have much refinement around the wells or river. To firstly address the wells, we will set maximum edge length limits on constrained points in the mesh domain.

➢ Tick the box next to **Max. edge length at constrained points**.

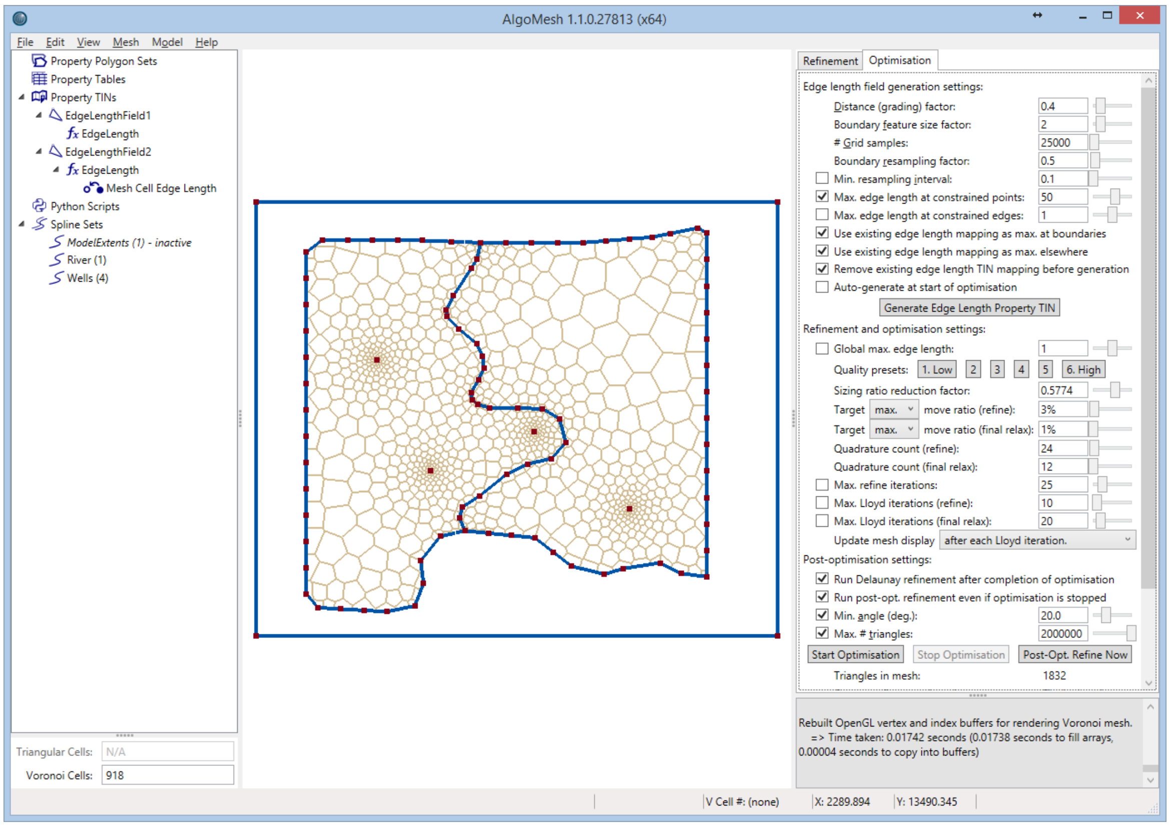
➢ Set **Max. edge length at constrained points** to **50**. This will decrease the size of cells in the vicinity of the wells (constrained points in the mesh domain that are not part of any constrained edges).

When you modify the settings, note that "Auto-generate at start of optimisation" is automatically switched back on. This ensures that a new edge length property TIN will be generated and applied the next time optimisation is run. Once you run the optimisation, this flag will be turned off, allowing you to change only the optimisation settings and re-run if desired, without having to regenerate the edge length property TIN.

We can now run the optimisation again to produce a new mesh with the updated edge length setting.

➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

You should now have a mesh with heavy refinement around the four wells in the model, as shown in Figure 38.



Triangular Cells: N/A
Voronoi Cells: 918

Edge length field generation settings:

- Distance (grading) factor: 0.4
- Boundary feature size factor: 2
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ☑ Max. edge length at constrained points: 50
- ☐ Max. edge length at constrained edges: 1
- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☐ Auto-generate at start of optimisation

Refinement and optimisation settings:

- ☐ Global max. edge length: 1
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:

- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped
- ☑ Min. angle (deg.): 20.0
- ☑ Max. # triangles: 2000000

Triangles in mesh: 1832

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.01742 seconds (0.01738 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

V Cell #: (none) | X: 2289.894 | Y: 13490.345

Figure 38: Setting a maximum edge length at constrained points allows us to refine to much smaller cell sizes around all wells in the model.

Note that this is a global setting which affects all constrained points in the mesh domain. It is also possible with AlgoMesh to selectively refine only around some of these wells, or to choose different cell sizes for refinement at each, by using property polygons mapped to Mesh Cell Edge Length. The Use Property Polygons to Control Mesh Cell Sizing how-to guide explores this in more detail.

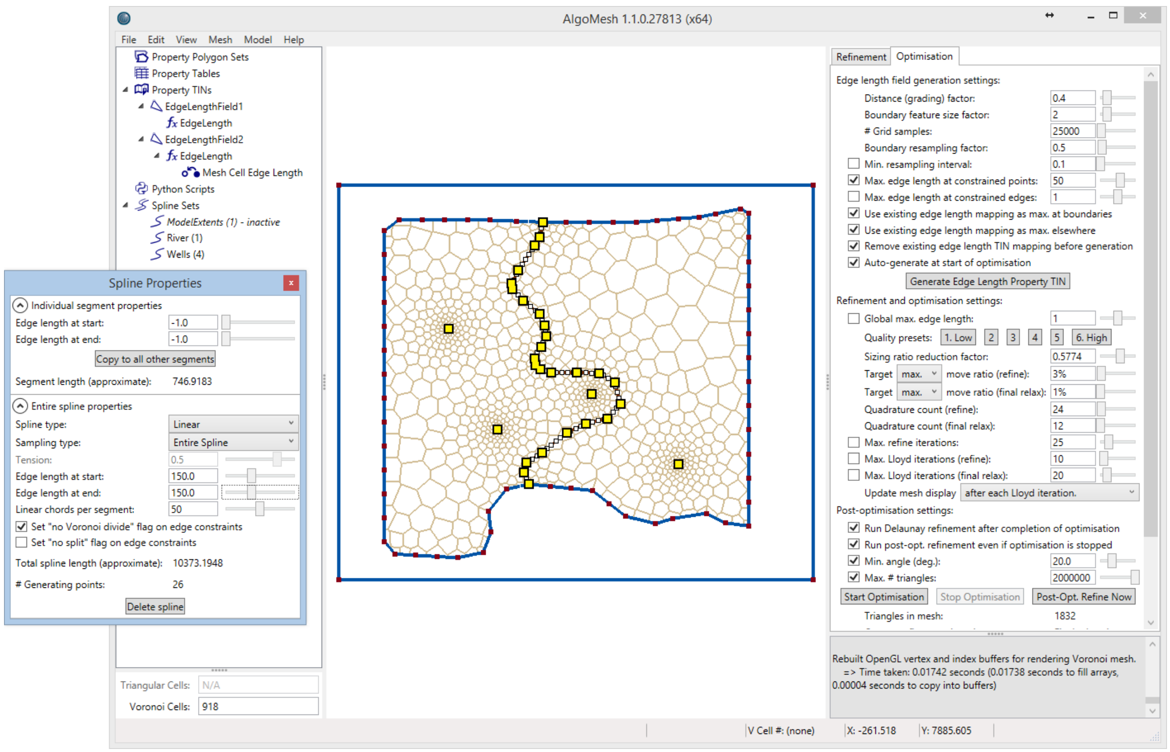
Refining the Mesh around the River

We would also like to increase the level of refinement around the river feature running through the centre of the model. We could use the **Max. edge length at constrained edges** setting to do this, but as this is a global setting, it would also affect cell sizing along the boundary of the model, which we would normally prefer to avoid, as it would add many cells to our model where they are probably not needed.

Instead, we will tell AlgoMesh to resample the polyline representing the river by inserting points at regular intervals along its length. As each of these points becomes a vertex in the final mesh, this will naturally increase the number of cells in the vicinity of the river.

- In the **View** menu, turn **Splines** back on so that you can see the orange river spline (polyline).
- In the **Edit** menu, select **Edit Splines** mode.
- **Move your mouse cursor over any part of the river spline** - it should turn a lighter orange colour - and **click the right button** of the mouse. This will open the **Spline Properties** window.
- In the **Spline Properties** window, under **Entire spline properties**, change **Sampling type** to **Entire Spline**.
- Also under **Entire spline properties**, change **Edge length at start** to **150** and **Edge length at end** to **150**.

Your Spline Properties window and the river spline in the view should now look like those shown in Figure 39. Note that a series of small white squares appears along the river spline in the mesh view. These are the locations at which AlgoMesh will insert points immediately prior to the mesh generation process starting.



Spline Properties

Individual segment properties
Edge length at start: -1.0
Edge length at end: -1.0
Copy to all other segments
Segment length (approximate): 746.9183

Entire spline properties
Spline type: Linear
Sampling type: Entire Spline
Tension: 0.5
Edge length at start: 150.0
Edge length at end: 150.0
Linear chords per segment: 50
☑ Set "no Voronoi divide" flag on edge constraints
☐ Set "no split" flag on edge constraints
Total spline length (approximate): 10373.1948
Generating points: 26
Delete spline

Triangular Cells: N/A
Voronoi Cells: 918

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☑ Max. edge length at constrained points: 50
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☑ Auto-generate at start of optimisation
Generate Edge Length Property TIN

Refinement and optimisation settings:
☐ Global max. edge length: 1
Quality presets: 1. Low 2 3 4 5 6. High
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000
Start Optimisation Stop Optimisation Post-Opt. Refine Now
Triangles in mesh: 1832

Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
=> Time taken: 0.01742 seconds (0.01738 seconds to fill arrays, 0.00004 seconds to copy into buffers)

X: -261.518 Y: 7885.605

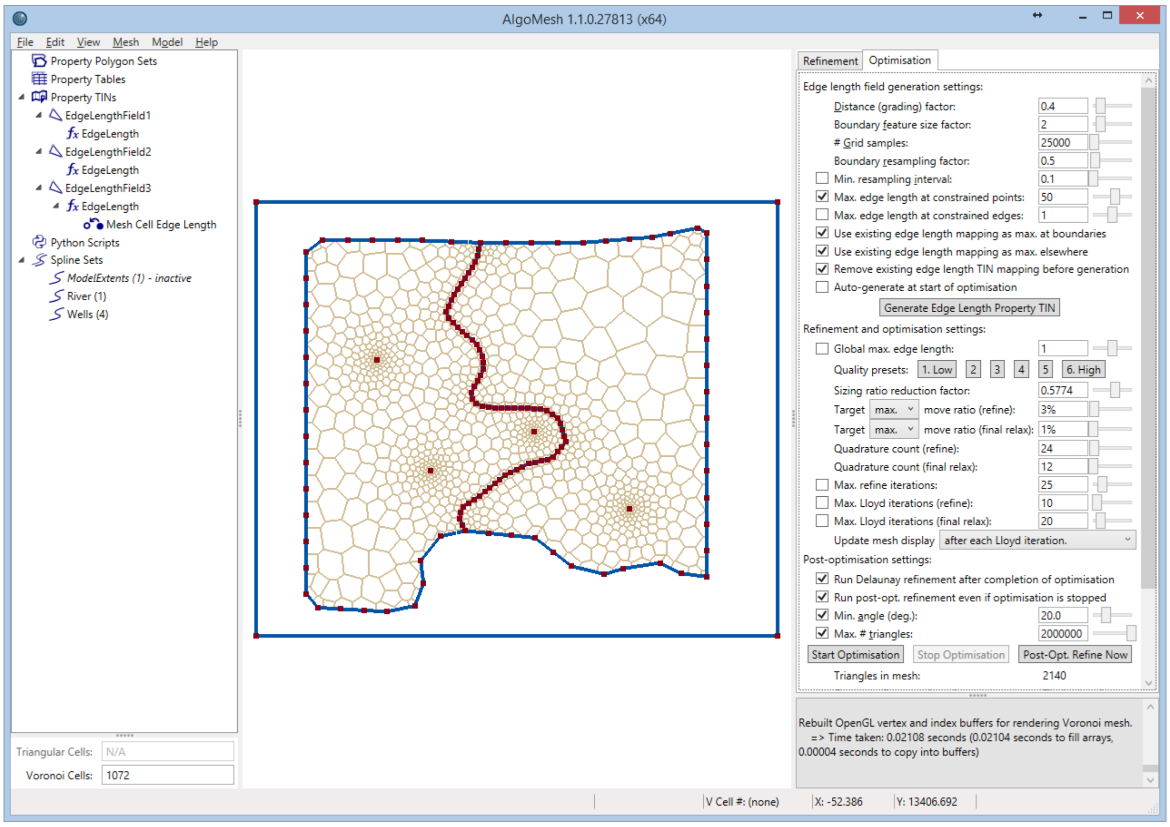
Figure 39: Setting the properties of the river spline allows resampling at a finer point spacing to control cell sizing along a linear feature.

There are a number of other spline properties you can use to get more control over linear features which we do not detail here, including the ability to represent them as smooth spline curves. The Manually Digitise and Edit Splines how-to guide explores these properties further.

We are now ready to regenerate the mesh to incorporate the resampled river polyline.

➢ **Click the red X** in the top-right corner of the **Spline Properties** window to close it.
➢ Ensure that **Auto-generate at start of optimisation** under **Edge length field generation settings** is turned on.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.
➢ In the **View** menu, turn **Splines** off once more to get a better look at the generated mesh.

You should end up with something like the mesh shown in Figure 40. Notice that the cells along the river are now somewhat smaller and more uniform in size, due to the uniform point spacing along the river polyline.



Triangular Cells: N/A
Voronoi Cells: 1072

Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	2
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
✓	Max. edge length at constrained points:	50
☐	Max. edge length at constrained edges:	1
✓	Use existing edge length mapping as max. at boundaries	
✓	Use existing edge length mapping as max. elsewhere	
✓	Remove existing edge length TIN mapping before generation	
☐	Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

✓	Run Delaunay refinement after completion of optimisation	
✓	Run post-opt. refinement even if optimisation is stopped	
✓	Min. angle (deg.):	20.0
✓	Max. # triangles:	2000000
	Triangles in mesh:	2140

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.02108 seconds (0.02104 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

V Cell #: (none) | X: -52.386 | Y: 13406.692

Figure 40: Resampling the river polyline at a point spacing of 150 produces a finer, more uniform array of cells along the river in the generated mesh.

For now, we will settle for this level of refinement. Note that we could extend the refinement further afield of the river by using an appropriate property polygon mapped to Mesh Cell Edge Length; this process is described in the Use Property Polygons to Control Mesh Cell Sizing how-to guide.

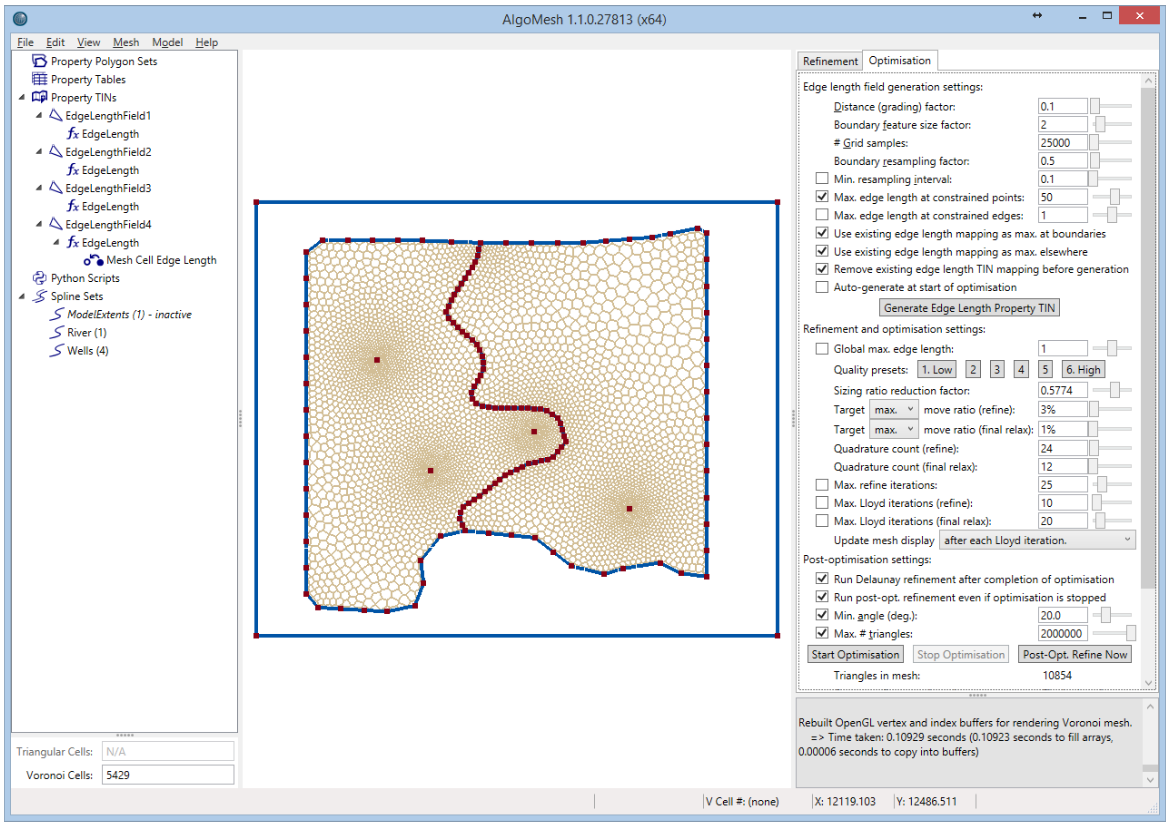
We will now look at the effect of varying some of the global edge length field generation and optimisation settings.

Altering the Distance (Grading) Factor

The distance (grading) factor controls how quickly cells grow in size with respect to their distance from the boundaries, where the smallest cells are. A low distance factor will result in a more uniform cell size over the mesh domain, and more cells overall.

➢ Change **Distance (grading) factor** to **0.1**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete. The process will take a little longer this time, as more cells are being generated.

Figure 41 shows the resulting mesh, which has around five times as many cells, and a much more gradual change in cell size moving away from the boundaries.



Edge length field generation settings:

Distance (grading) factor:	0.1
Boundary feature size factor:	2
# Grid samples:	25000
Boundary resampling factor:	0.5
☐ Min. resampling interval:	0.1
☑ Max. edge length at constrained points:	50
☐ Max. edge length at constrained edges:	1
☑ Use existing edge length mapping as max. at boundaries	
☑ Use existing edge length mapping as max. elsewhere	
☑ Remove existing edge length TIN mapping before generation	
☐ Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐ Global max. edge length:	1
Sizing ratio reduction factor:	0.5774
Target max. move ratio (refine):	3%
Target max. move ratio (final relax):	1%
Quadrature count (refine):	24
Quadrature count (final relax):	12
☐ Max. refine iterations:	25
☐ Max. Lloyd iterations (refine):	10
☐ Max. Lloyd iterations (final relax):	20
Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

☑ Run Delaunay refinement after completion of optimisation	
☑ Run post-opt. refinement even if optimisation is stopped	
☑ Min. angle (deg.):	20.0
☑ Max. # triangles:	2000000
Triangles in mesh:	10854

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.10929 seconds (0.10923 seconds to fill arrays,
0.00006 seconds to copy into buffers)
```

Triangular Cells:	N/A
Voronoi Cells:	5429

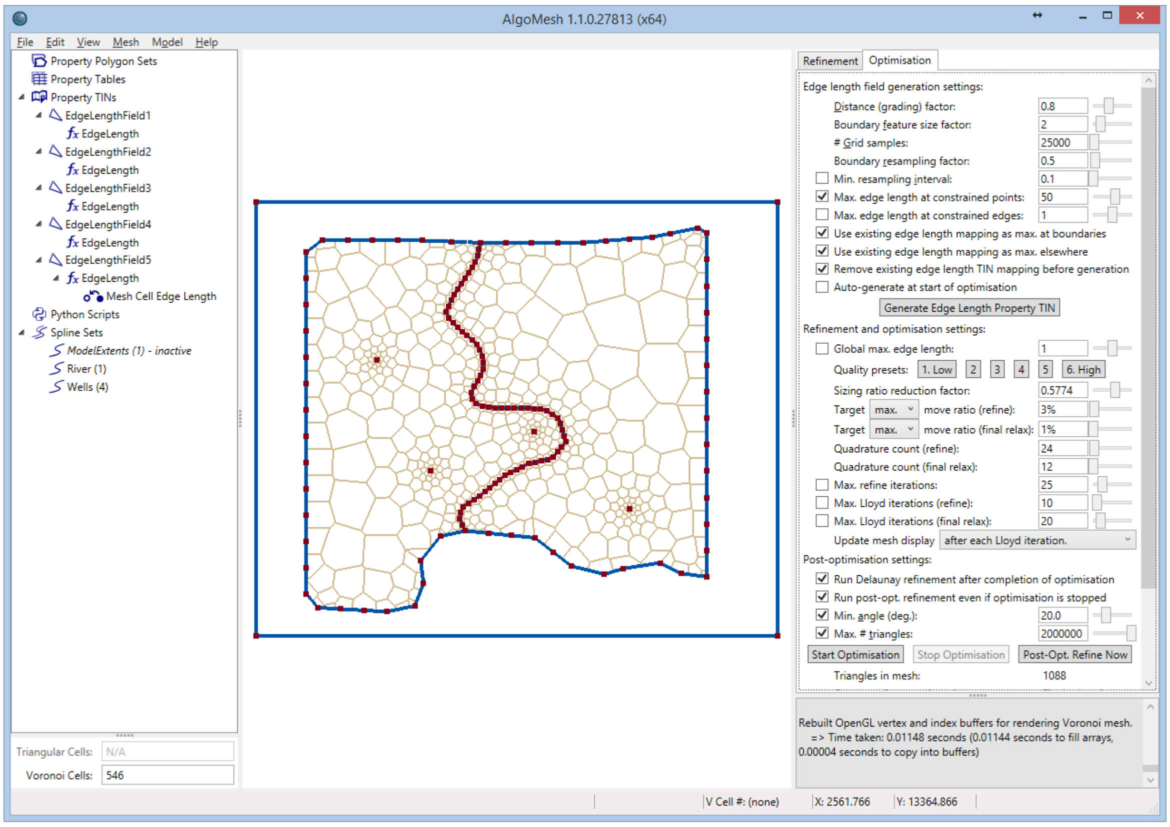
V Cell #: (none) | X: 12119.103 | Y: 12486.511

Figure 41: A low distance (grading) factor results in more cells of a more uniform size.

In contrast, a high distance factor will create much larger cells in empty areas of the mesh domain, with a rapid grading in size from large to small for the cells approaching the boundaries.

➢ Change **Distance (grading) factor** to **0.8**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 42 shows the resulting mesh, which has fewer cells, but more abrupt changes in cell size. In general, increasing the distance (grading) factor can help to reduce the number of cells in a mesh, but care should be taken to avoid going too far; high values may lead to poor-quality cells close to the boundaries, as they are "stretched" to try to meet the rapid changes in desired cell size. Note in particular that the cells along the river are much more elongated than in the mesh produced with a low distance factor.



Triangular Cells: N/A
Voronoi Cells: 546

Edge length field generation settings:
- Distance (grading) factor: 0.8
- Boundary feature size factor: 2
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- [ ] Min. resampling interval: 0.1
- [✓] Max. edge length at constrained points: 50
- [ ] Max. edge length at constrained edges: 1
- [✓] Use existing edge length mapping as max. at boundaries
- [✓] Use existing edge length mapping as max. elsewhere
- [✓] Remove existing edge length TIN mapping before generation
- [ ] Auto-generate at start of optimisation

Refinement and optimisation settings:
- [ ] Global max. edge length: 1
- Quality presets: 1. Low 2 3 4 5 6. High
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- [ ] Max. refine iterations: 25
- [ ] Max. Lloyd iterations (refine): 10
- [ ] Max. Lloyd iterations (final relax): 20
- Update mesh display: after each Lloyd iteration.

Post-optimisation settings:
- [✓] Run Delaunay refinement after completion of optimisation
- [✓] Run post-opt. refinement even if optimisation is stopped
- [✓] Min. angle (deg.): 20.0
- [✓] Max. # triangles: 2000000

Triangles in mesh: 1088

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.01148 seconds (0.01144 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

V Cell #: (none) | X: 2561.766 | Y: 13364.866

Figure 42: A high distance (grading) factor results in fewer cells with more sudden changes in size moving from boundaries to empty regions of the mesh domain.

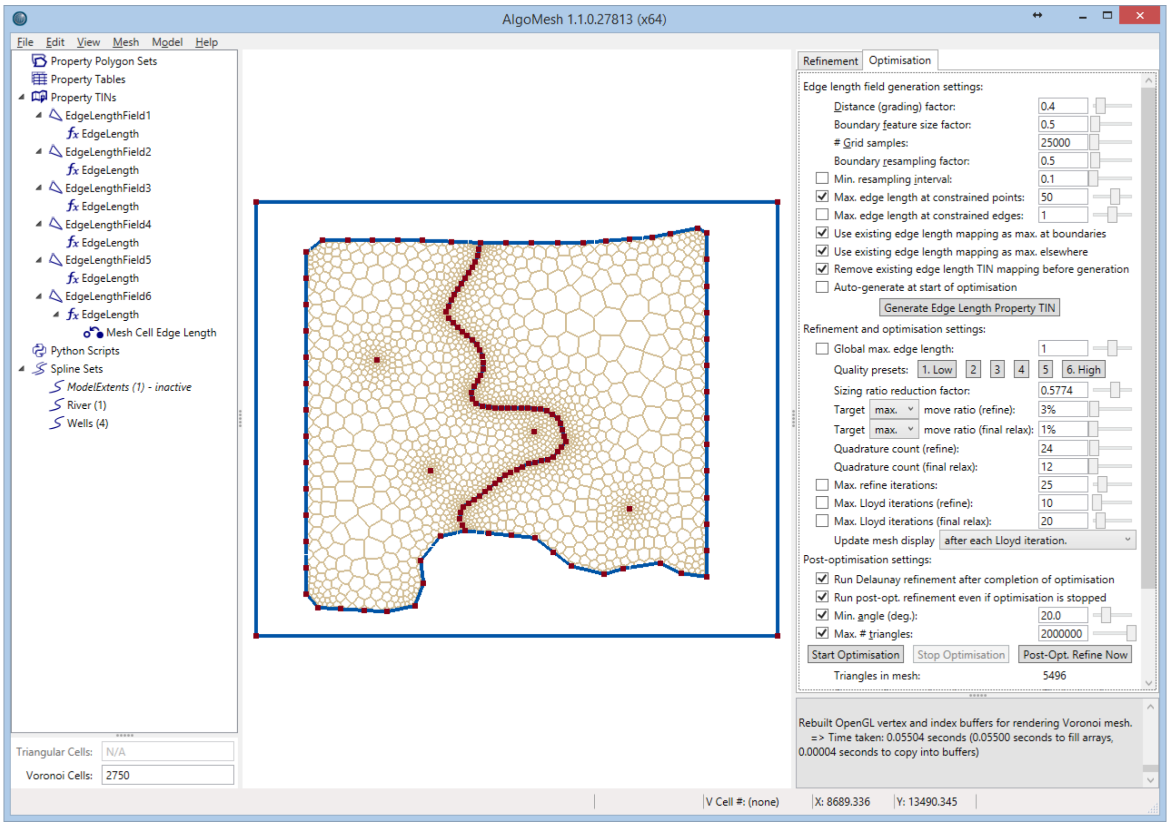
Altering the Boundary Feature Size Factor

The boundary feature size factor controls how many cells are generated along the boundaries of the mesh domain (including the outer extents of the model and any interior linear features, like the river in our model). The sizing of cells along the boundaries is always dependent on a combination of the spacing of input points and the distance from one boundary to other nearby boundaries, but the boundary feature size factor acts as a multiplier on this.

A low boundary feature size factor will have a similar effect to specifying a global maximum edge length at constrained edges, but will allow larger cells along parts of a boundary that are a long way from other boundaries.

- ➢ Change **Distance (grading) factor** back to the default of **0.4**.
- ➢ Change **Boundary feature size factor** to **0.5**.
- ➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 43 shows the resulting mesh. Note the heavier refinement along the boundaries in general, which is exaggerated further along the river where points are spaced more closely together, and to a lesser degree at corners of the model extents, where the boundary lines are somewhat closer to one another. Areas near tight angles are refined much more heavily.



Triangular Cells: N/A
Voronoi Cells: 2750

Edge length field generation settings:

- Distance (grading) factor: 0.4
- Boundary feature size factor: 0.5
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ☑ Max. edge length at constrained points: 50
- ☐ Max. edge length at constrained edges: 1
- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☐ Auto-generate at start of optimisation

Refinement and optimisation settings:

- ☐ Global max. edge length: 1
- Quality presets: 1. Low 2 3 4 5 6. High
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:

- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped
- ☑ Min. angle (deg.): 20.0
- ☑ Max. # triangles: 2000000

Triangles in mesh: 5496

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.05504 seconds (0.05500 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

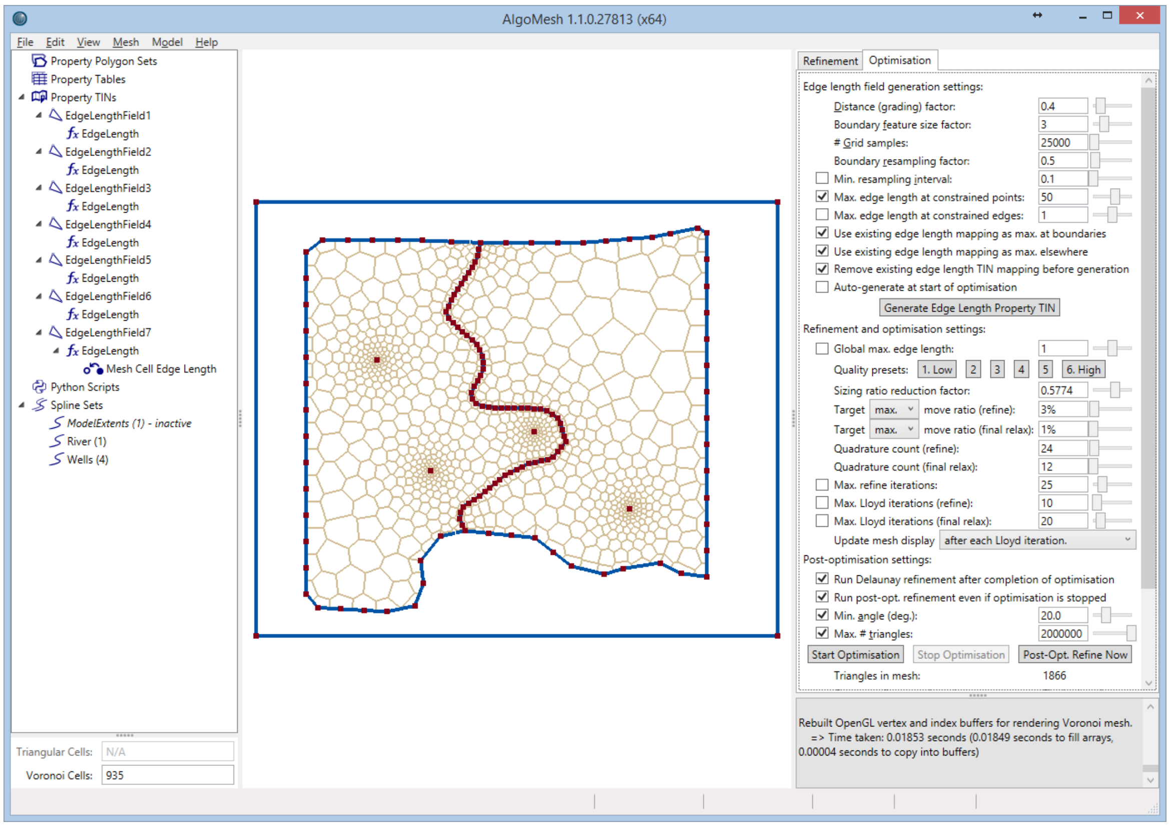
V Cell #: (none) X: 8689.336 Y: 13490.345

Figure 43: A low boundary feature size factor results in more cells being generated along the boundaries, without specifying a fixed upper limit on cell size.

In contrast, a high boundary feature size factor will increase the relative size of cells along boundaries, producing fewer, larger cells.

➢ Change **Boundary feature size factor** to **3**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 44 shows the result of using a higher boundary feature size factor. Cells along the river are further elongated and there is less refinement at tight angles.



Triangular Cells: N/A
Voronoi Cells: 935

Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	3
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
✓	Max. edge length at constrained points:	50
☐	Max. edge length at constrained edges:	1
✓	Use existing edge length mapping as max. at boundaries	
✓	Use existing edge length mapping as max. elsewhere	
✓	Remove existing edge length TIN mapping before generation	
☐	Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

✓	Run Delaunay refinement after completion of optimisation	
✓	Run post-opt. refinement even if optimisation is stopped	
✓	Min. angle (deg.):	20.0
✓	Max. # triangles:	2000000
	Triangles in mesh:	1866

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.01853 seconds (0.01849 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

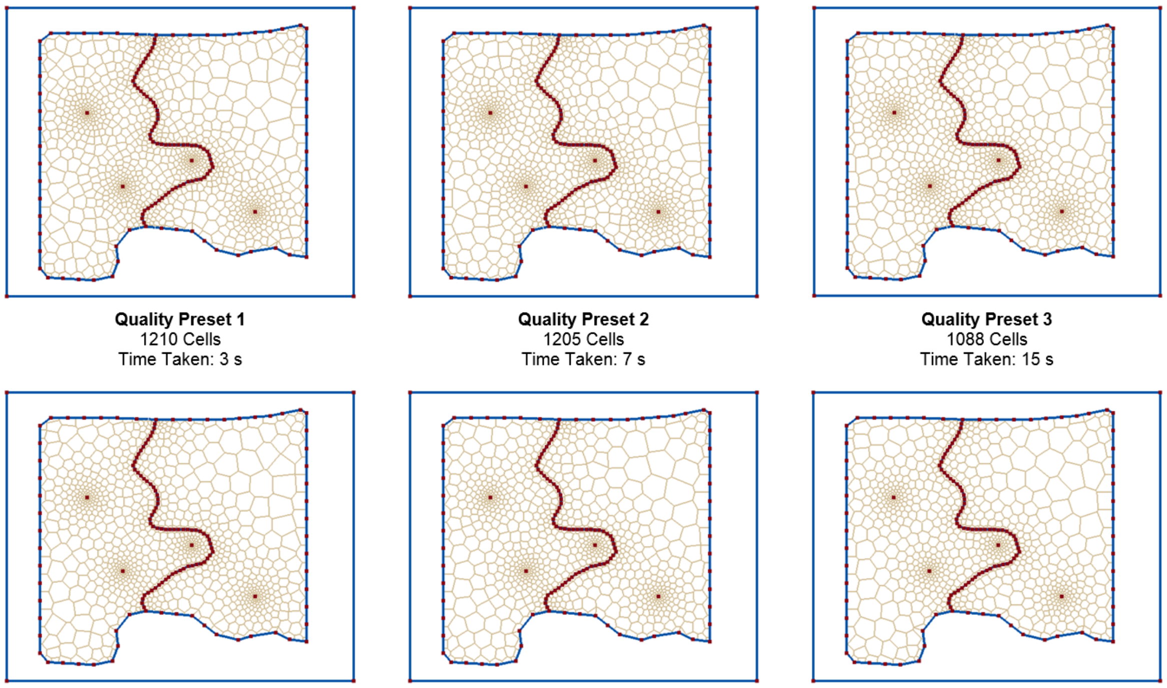
Figure 44: A high boundary feature size factor produces fewer cells of larger size along boundaries.

Altering the Optimisation Quality Preset

Once you are happy with the edge length field generation settings, you may wish to experiment with the quality preset levels to control the optimisation process itself. So far, we have been using the default quality preset 3 to generate all of our meshes. Clicking any of the quality preset buttons changes the five individual settings below the quality preset line. Note that you can also change these five settings independently, but if you click one of the preset levels again, your changes will be overwritten with the preset values.

More information on each of the individual settings and their effect can be found in the Meshing Control File (.mcf) file format reference. In general, however, choosing a preset between 1 and 6 is usually sufficient.

Figure 45 shows meshes generated for the same model using each of the quality preset levels. The differences between levels are somewhat subtle in this example, but can become significant when the mesh domain becomes more complex.



Quality Preset 1
1210 Cells
Time Taken: 3 s

Quality Preset 2
1205 Cells
Time Taken: 7 s

Quality Preset 3
1088 Cells
Time Taken: 15 s

Quality Preset 4
1053 Cells
Time Taken: 16 s

Quality Preset 5
989 Cells
Time Taken: 41 s

Quality Preset 6
919 Cells
Time Taken: 2 min 31 s

Figure 45: Voronoi meshes generated using each of the optimisation quality preset levels.

Finishing Up

You may have noticed a number of "EdgeLengthField" property TINs building up with each mesh generation in the project explorer on the left-hand side of the AlgoMesh window. These are the edge length fields generated by the optimisation process. They may be removed once mesh generation is complete to save a bit of memory.

- **Right-click** on the **Property TINs** header.
- Select **Remove All Property TINs** from the context menu that appears. This will remove all of the edge length field TINs. Alternatively, you could remove only some of the TINs by right-clicking on each TIN you want to remove and choosing Remove from the menu.

Once you are happy with your generated mesh, you may want to save it so you can retrieve it later.

- Go to the **File** menu and choose **Save Project**.
- **Browse to a location** to which you have write access, such as your Documents folder.
- **Type a name** for the project file to be saved, e.g. **FinalOptMesh.amproj** and click **Save**.

AlgoMesh will save the mesh and all of the settings from the user interface to the project file. You can bring it all back at a later date by going to the **File** menu, choosing **Open Project** and opening your saved file.

3.5 Import Geometry from GIS

Before any mesh generation can take place in AlgoMesh, an appropriate outer boundary for the model must be defined, and any interior point or linear features brought in. The usual method for doing this is to import geometry for the boundary and interior features from GIS shapefiles. This how-to will take you through starting a new project in AlgoMesh from scratch, bringing in the model boundary and a number of interior model features until you have a model domain ready for mesh generation. This procedure is the same regardless of whether you will be generating triangular or Voronoi grids.

It is recommended that you first go through one of the Delaunay refinement tutorials (for triangular or Voronoi grids) if you have not already, as these introduce some of the basic knowledge that is assumed in this section.

Defining the Model Extents

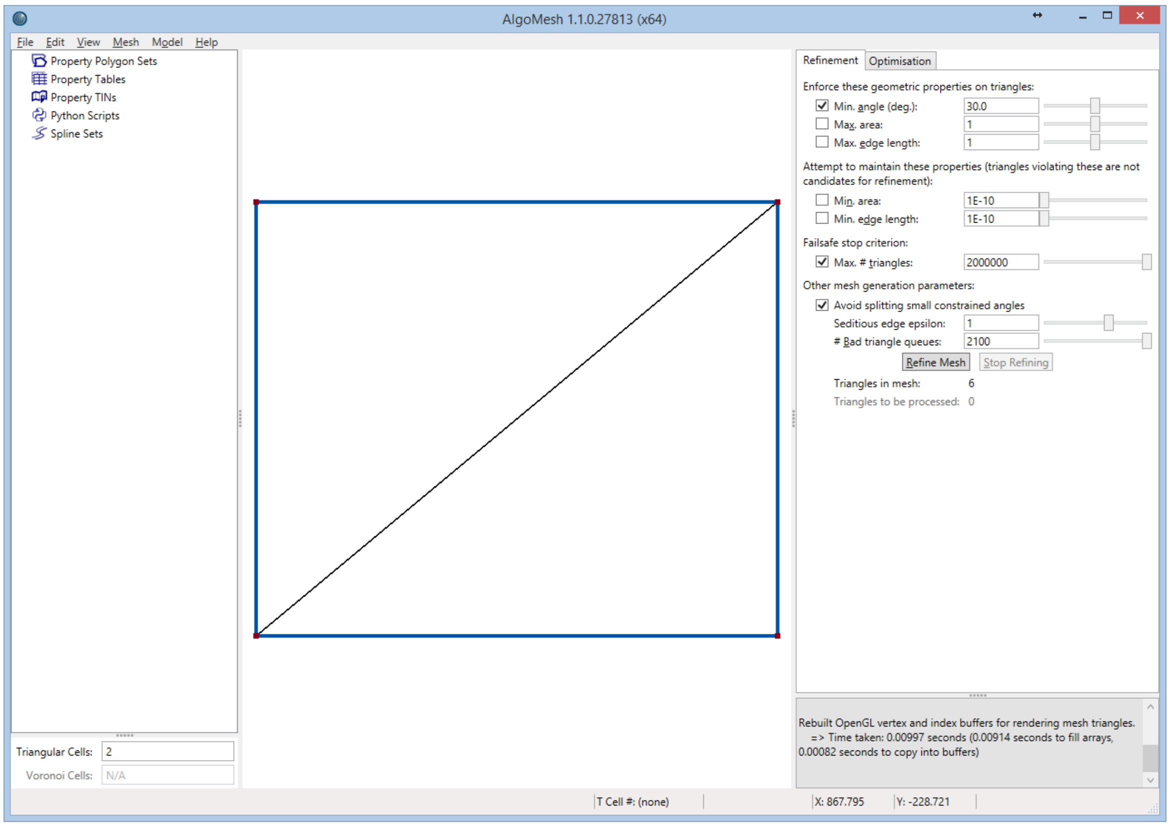
To bring the outer model boundary (the model extents) into AlgoMesh, we will use the following process:

1. Bring in a rectangular bounding box encompassing the entire model domain.
2. Import a polygon shapefile containing the actual model extents, which may be any shape. This polygon should fit well inside the rectangular bounding box, with some space around its edges.
3. (Optionally) resample the model extents polygon to obtain uniform point spacing along its perimeter.
4. "Bake" the resampled model extents polygon into the mesh such that the points and polyline edges become a permanent part of the mesh domain.
5. Deactivate the area outside the model extents polygon (but inside the rectangular bounding box) so that it will not be included in the mesh generation process.

Start by launching AlgoMesh and importing the geometry for the bounding box from the shapefile BoundingRectangle.shp.

➢ Go to the **File** menu and select **Import Mesh Geometry**.
➢ Browse to the **Samples** folder of your AlgoMesh installation.
➢ Select **BoundingRectangle.shp** and click **Open**.

AlgoMesh will load the bounding rectangle, and you should now see a basic mesh domain dividing this rectangle into two triangles, as in Figure 46.



Enforce these geometric properties on triangles:
- ☑ Min. angle (deg.): 30.0
- ☐ Max. area: 1
- ☐ Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- ☐ Min. area: 1E-10
- ☐ Min. edge length: 1E-10

Failsafe stop criterion:
- ☑ Max. # triangles: 2000000

Other mesh generation parameters:
- ☑ Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 6
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.00997 seconds (0.00914 seconds to fill arrays,
0.00082 seconds to copy into buffers)
```

Triangular Cells: 2
Voronoi Cells: N/A

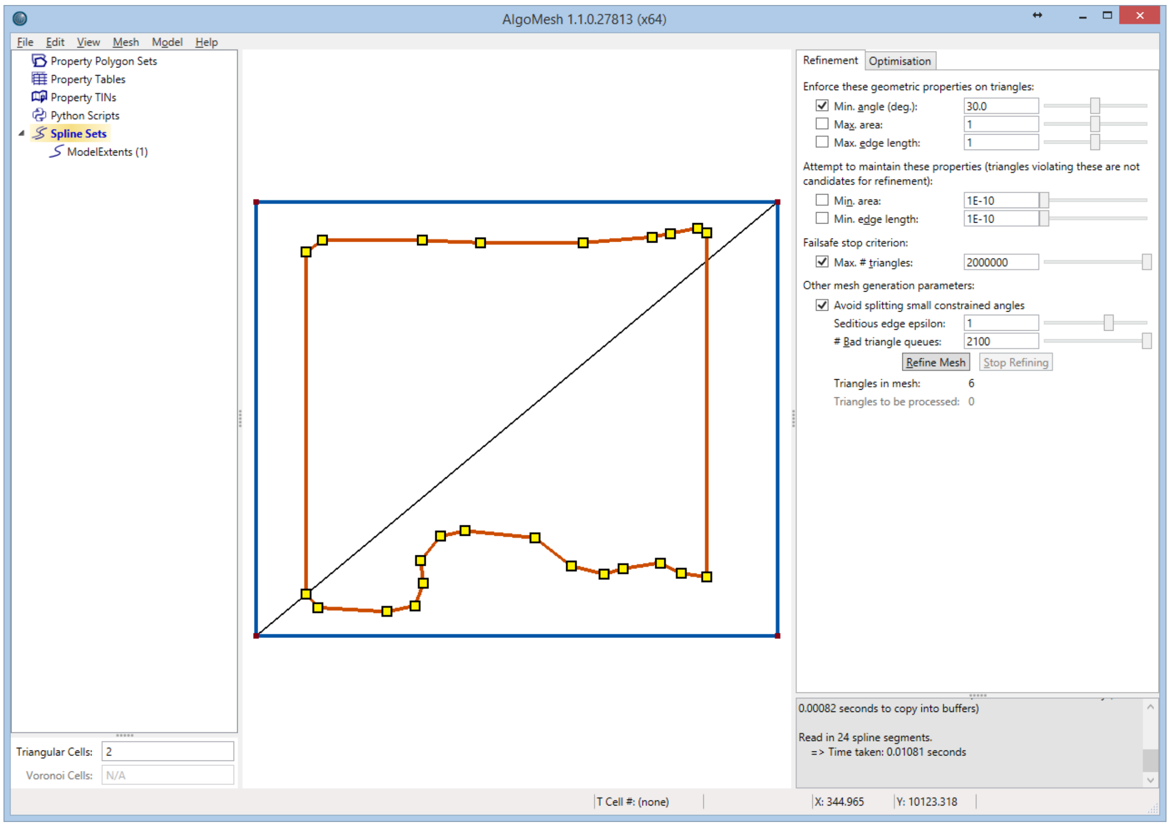
T Cell #: (none) X: 867.795 Y: -228.721

Figure 46: Importing a shapefile containing a bounding rectangle of the model gives us an initial mesh domain to work with.

We will now import the real model boundary polygon from shapefile as a Spline Set in AlgoMesh, which will allow us to resample the boundary before adding it into the mesh.

- ➢ **Right-click** on the **Spline Sets** header on the left-hand side of the AlgoMesh window.
- ➢ Choose **Import GIS Polylines as Splines** from the context menu that appears.
- ➢ Select **ModelExtents.shp** and click **Open**.

The model extents polygon layer is added underneath the Spline Sets header, and the polygon itself is now shown in an orange colour in the mesh view (Figure 47).



Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 6
Triangles to be processed: 0

```
0.00082 seconds to copy into buffers)

Read in 24 spline segments.
  => Time taken: 0.01081 seconds
```

Triangular Cells: 2
Voronoi Cells: N/A

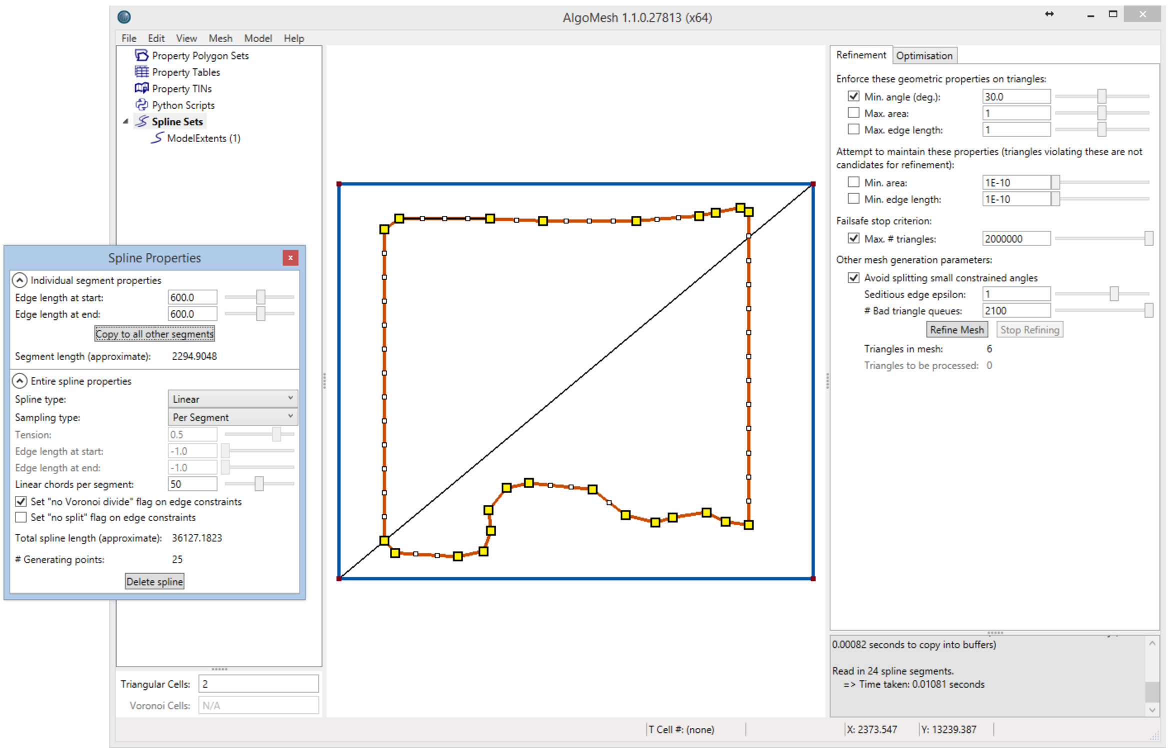
T Cell #: (none) X: 344.965 Y: 10123.318

Figure 47: The initial mesh domain with a spline delineating the actual model extents overlying it.

We could add the model extents polygon into the mesh as it is, but it is often preferable to resample it first, to promote uniform sizing of cells, and limit their maximum size, along the boundaries of the model.

- Go to the **Edit** menu and activate **Edit Splines** mode.
- **Move your mouse cursor over any part of the river spline** - it should turn a lighter orange colour - and **click the right button** of the mouse. This will open the **Spline Properties** window.
- In the **Spline Properties** window under **Individual segment properties**, set **Edge length at start** to **600** and **Edge length at end** to **600**.
- Again in the **Spline Properties** window under **Individual segment properties**, click the **Copy to all other segments** button.

The mesh view should update to show white squares at each of the locations where a point will be inserted into the mesh along the boundary, as shown in Figure 48. Note that in this case, we have left "Sample type" as "Per Segment" instead of "Entire Spline". This will force AlgoMesh to include in the mesh all of the vertices that are part of the original polygon, and create additional points at an approximate spacing of 600 between each pair of vertices. If we had instead chosen "Entire Spline", AlgoMesh would have created points at a spacing of 600 along the entire perimeter of the polygon, irrespective of whether they lined up with the original vertices or not.



Spline Properties

Individual segment properties
Edge length at start: 600.0
Edge length at end: 600.0
Segment length (approximate): 2294.9048

Entire spline properties
Spline type: Linear
Sampling type: Per Segment
Tension: 0.5
Edge length at start: -1.0
Edge length at end: -1.0
Linear chords per segment: 50
[✓] Set "no Voronoi divide" flag on edge constraints
[ ] Set "no split" flag on edge constraints
Total spline length (approximate): 36127.1823
Generating points: 25

Enforce these geometric properties on triangles:
[✓] Min. angle (deg.): 30.0
[ ] Max. area: 1
[ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
[ ] Min. area: 1E-10
[ ] Min. edge length: 1E-10

Failsafe stop criterion:
[✓] Max. # triangles: 2000000

Other mesh generation parameters:
[✓] Avoid splitting small constrained angles
Seditious edge epsilon: 1
Bad triangle queues: 2100

Triangles in mesh: 6
Triangles to be processed: 0

```
0.00082 seconds to copy into buffers)

Read in 24 spline segments.
  => Time taken: 0.01081 seconds
```

Triangular Cells: 2
Voronoi Cells: N/A

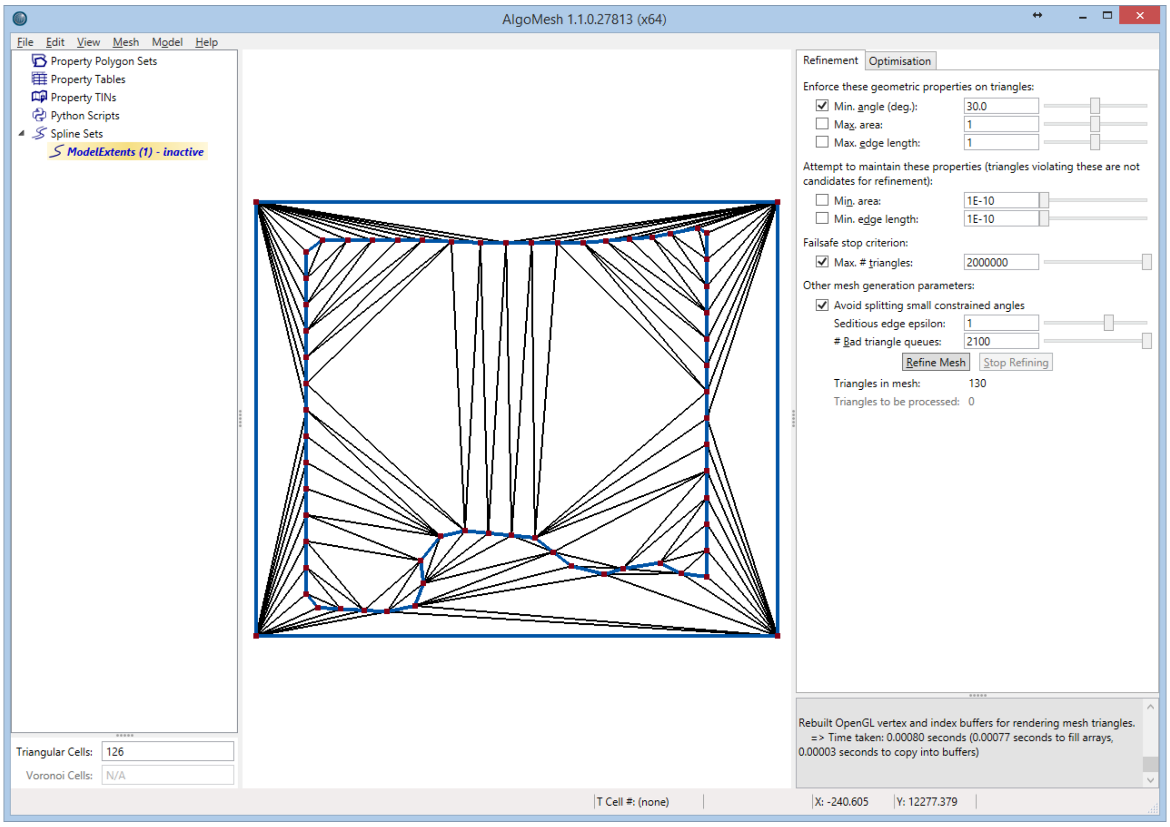
T Cell #: (none) X: 2373.547 Y: 13239.387

Figure 48: A uniform edge length of 600 is set for all segments of the model extents spline.

We are now ready to "bake" our resampled model boundary into the mesh.

- **Click the red X** in the top-right corner of the **Spline Properties** window to close it.
- **Right-click** the ModelExtents layer in the project explorer on the left-hand side of the AlgoMesh window.
- In the context menu that appears, choose **Add to Mesh Now and Deactivate Set**, followed by **Original Mesh and Generated Mesh**.

The model extents are now added directly into the mesh itself (Figure 49). Note also that the ModelExtents entry in the project explorer has been made "inactive". This means that the polygon will no longer be added to the mesh when mesh generation is performed - this is important here, as we have already manually included the geometry in the mesh.



Enforce these geometric properties on triangles:

✓	Min. angle (deg.):	30.0
☐	Max. area:	1
☐	Max. edge length:	1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

☐	Min. area:	1E-10
☐	Min. edge length:	1E-10

Failsafe stop criterion:

✓	Max. # triangles:	2000000

Other mesh generation parameters:

✓ Avoid splitting small constrained angles

Seditious edge epsilon:	1
# Bad triangle queues:	2100
Triangles in mesh:	130
Triangles to be processed:	0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00080 seconds (0.00077 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

Triangular Cells:	126
Voronoi Cells:	N/A

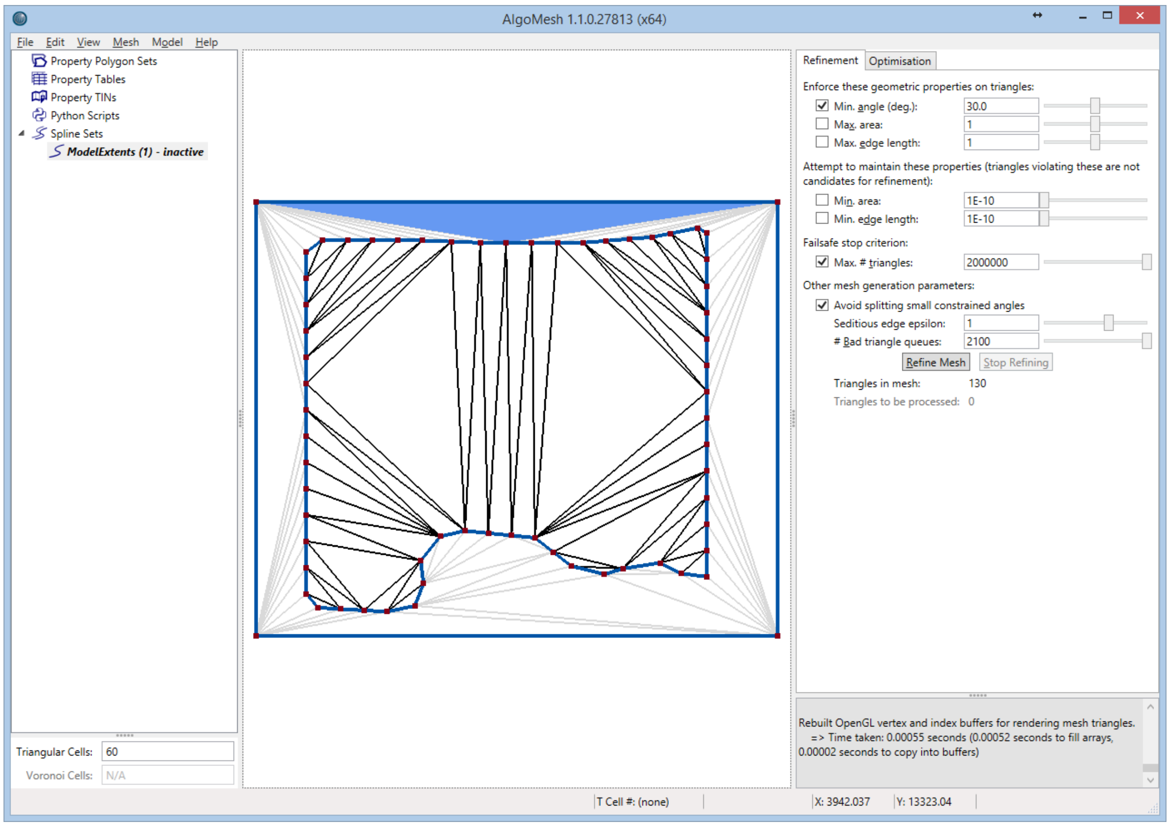
T Cell #: (none) | X: -240.605 | Y: 12277.379

Figure 49: The triangulated mesh domain containing the resampled model extents boundary.

There is one task remaining to complete the definition of the model extents: to deactivate the parts of the domain that should not be included in the final mesh. If we do not do this, the entire area inside the outer bounding rectangle will be considered part of the model, and the boundary polygon we just added would act as a linear feature, instead of the outside boundary of the mesh.

- Go to the **Edit** menu and activate **Edit Mesh** mode.
- **Click** in the mesh view on any one of the triangles that is **outside the model extents polygon but inside the outer bounding rectangle**. The triangle will turn blue to indicate that it has been selected. If you accidentally selected a triangle in the wrong area, just click again on a different triangle to change this selection.
- **Press the minus ('-') key** on your keyboard. All of the triangles outside the model extents polygon should turn grey (inactive), as in Figure 50.
- **Press the ESC key** on your keyboard to deselect the triangle. The blue colour should disappear, as the triangle selection has been cleared.

You should now have a complete model boundary, with the grey area outside this boundary excluded from the mesh domain. You can also use this technique to create "holes" in the interior of your mesh domain, should your model require it.



Enforce these geometric properties on triangles:

- ☑ Min. angle (deg.): 30.0
- ☐ Max. area: 1
- ☐ Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- ☐ Min. area: 1E-10
- ☐ Min. edge length: 1E-10

Failsafe stop criterion:

- ☑ Max. # triangles: 2000000

Other mesh generation parameters:

- ☑ Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00055 seconds (0.00052 seconds to fill arrays,
0.00002 seconds to copy into buffers)
```

Triangular Cells: 60
Voronoi Cells: N/A

T Cell #: (none) | X: 3942.037 | Y: 13323.04

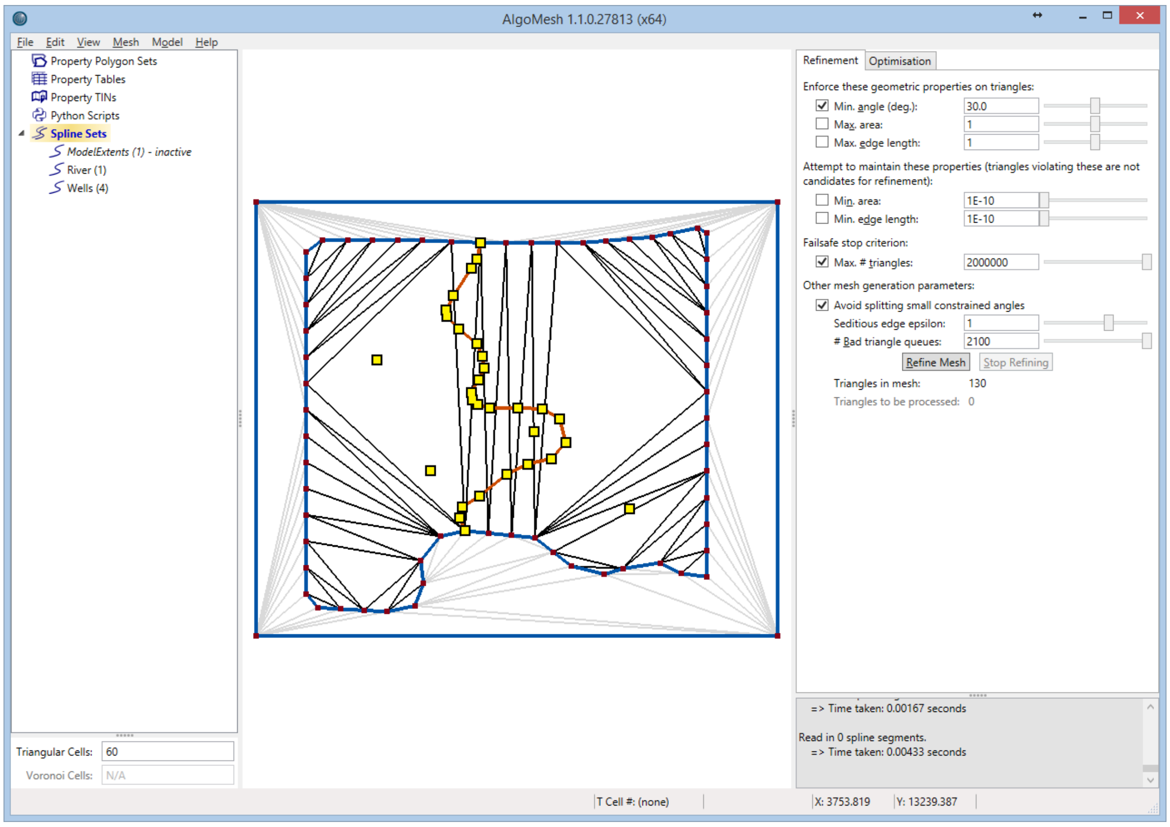
Figure 50: Pressing the minus ('-') key on the keyboard in Edit Mesh mode with a triangle selected turns all triangles in that enclosed area grey (inactive).

Adding Interior Features

Once you have an appropriate model extents polygon defined in the mesh, it is time to add any interior point, polyline or polygon features that should be included in the mesh for your model. For this example, we are going to add a polyline representing a river through the centre of the model domain, and four points representing groundwater wells.

- ➢ **Right-click** on the **Spline Sets** header.
- ➢ Choose **Import GIS Polylines as Splines** from the context menu that appears.
- ➢ Select **River.shp** and click **Open**. A River layer is added to the list of Spline Sets and the river polyline appears in the view.
- ➢ **Right-click** again on the **Spline Sets** header, and choose **Import GIS Polylines as Splines** from the context menu that appears.
- ➢ Select **Wells.shp** and click **Open**. A Wells layer is added to the list of Spline Sets and the well locations are added as yellow squares in the view.

That is all there is to importing your geometry into AlgoMesh. Figure 51 shows the completed mesh domain.



Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

```
   => Time taken: 0.00167 seconds

Read in 0 spline segments.
   => Time taken: 0.00433 seconds
```

Triangular Cells: 60
Voronoi Cells: N/A

T Cell #: (none) | X: 3753.819 | Y: 13239.387

Figure 51: The complete mesh domain, ready for mesh generation.

Note that although we didn't do it here, polygons may also be added using this mechanism. Spline Properties for the interior elements may also be manipulated if desired to resample linear features appropriately prior to mesh generation. See the Manually Digitise and Edit Splines how-to guide for a more thorough explanation of these properties.

Note that we do not need to bake these interior features into the mesh like we did with the model extents. That was only necessary so that we could instruct AlgoMesh which parts of the domain should be considered "inside" and "outside" the mesh. When mesh generation is performed, the features from all active spline sets will be added into the mesh automatically. This allows easy manipulation of spline properties and subsequent re-meshing, without the need to re-import the geometry every time.

Now that the mesh domain is complete, you may want to save it to an AlgoMesh project file.

➢ Go to the **File** menu and choose **Save Project**.
➢ **Browse to a location** to which you have write access, such as your Documents folder.
➢ **Type a name** for the project file to be saved, e.g. **MeshDomain.amproj** and click **Save**.

3.6 Use Property Polygons to Control Mesh Cell Sizing

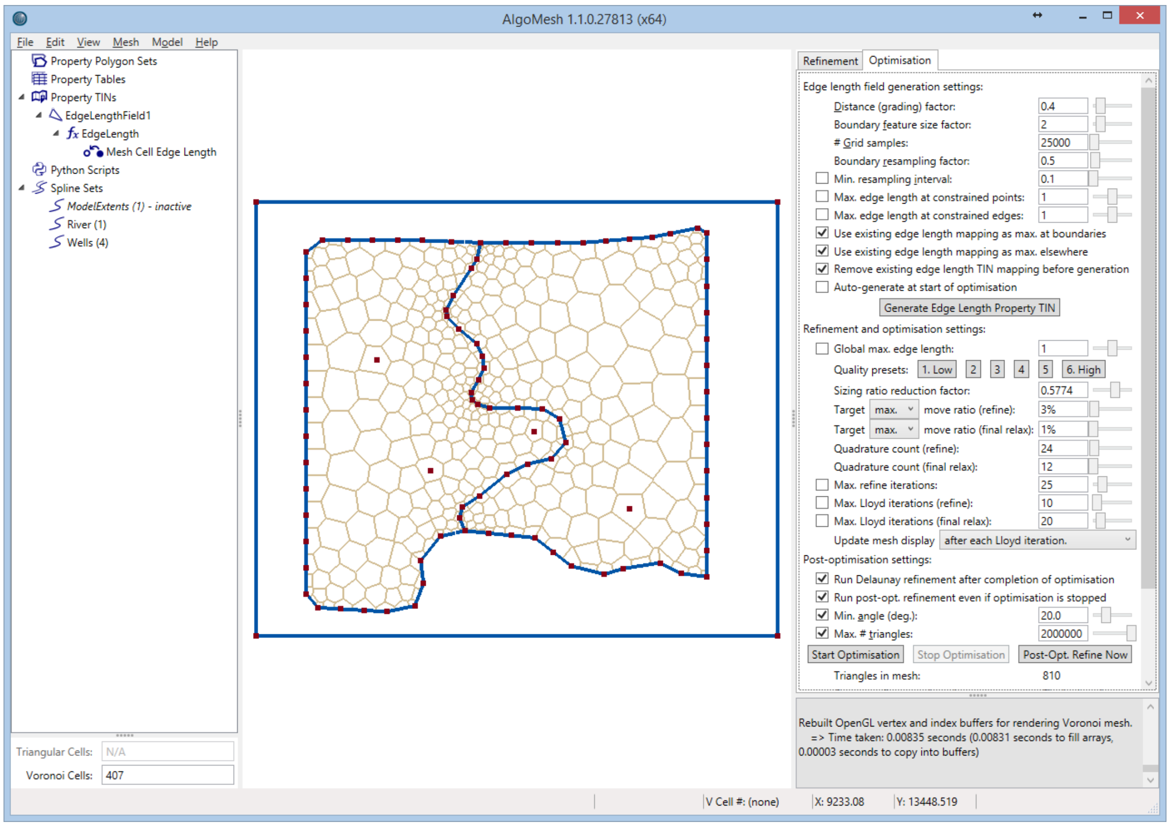
In this how-to, you will learn how to use polygons to create additional refinement in specified areas of a mesh. We will be using Voronoi grids to illustrate, but the procedure is the same regardless of whether you will be generating triangular or Voronoi grids.

We will be starting from the same geometry that was used in the Delaunay refinement and optimisation how-to sections. It is recommended that you first go through the optimisation tutorial if you have not already, as it introduces some of the basic knowledge that is assumed in this section.

Start by launching AlgoMesh and opening the project file river-mesh-domain.amproj.

- Go to the **File** menu and choose **Open Project**.
- Browse to the **Samples** folder of your AlgoMesh installation.
- Select **river-mesh-domain.amproj** and click **Open**.
- In the **View** menu, turn off **Triangles**.
- In the **View** menu, turn on **Voronoi Cells**.
- In the **View** menu, turn off **Splines** so that you have a good view of the mesh you are going to generate.
- Select the **Optimisation** tab on the right-hand side of the AlgoMesh window.
- Click the **Start Optimisation** button near the bottom of the Optimisation tab, and wait for mesh generation to complete.

Your screen should look like that shown in Figure 52.



Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	2
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
☐	Max. edge length at constrained points:	1
☐	Max. edge length at constrained edges:	1
✓	Use existing edge length mapping as max. at boundaries	
✓	Use existing edge length mapping as max. elsewhere	
✓	Remove existing edge length TIN mapping before generation	
☐	Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

✓	Run Delaunay refinement after completion of optimisation	
✓	Run post-opt. refinement even if optimisation is stopped	
✓	Min. angle (deg.):	20.0
✓	Max. # triangles:	2000000

Triangles in mesh: 810

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.00835 seconds (0.00831 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

Triangular Cells: N/A
Voronoi Cells: 407

V Cell #: (none) | X: 9233.08 | Y: 13448.519

Figure 52: The initial mesh generated by the optimisation method.

In the optimisation tutorial, we controlled cell sizing to some extent by resampling to a point spacing of 150 along the river polyline, and by specifying a maximum edge length of 50 at constrained points (wells). Both of these techniques can be useful, but sometimes we need more control over the location and spatial extent of cell size constraints.

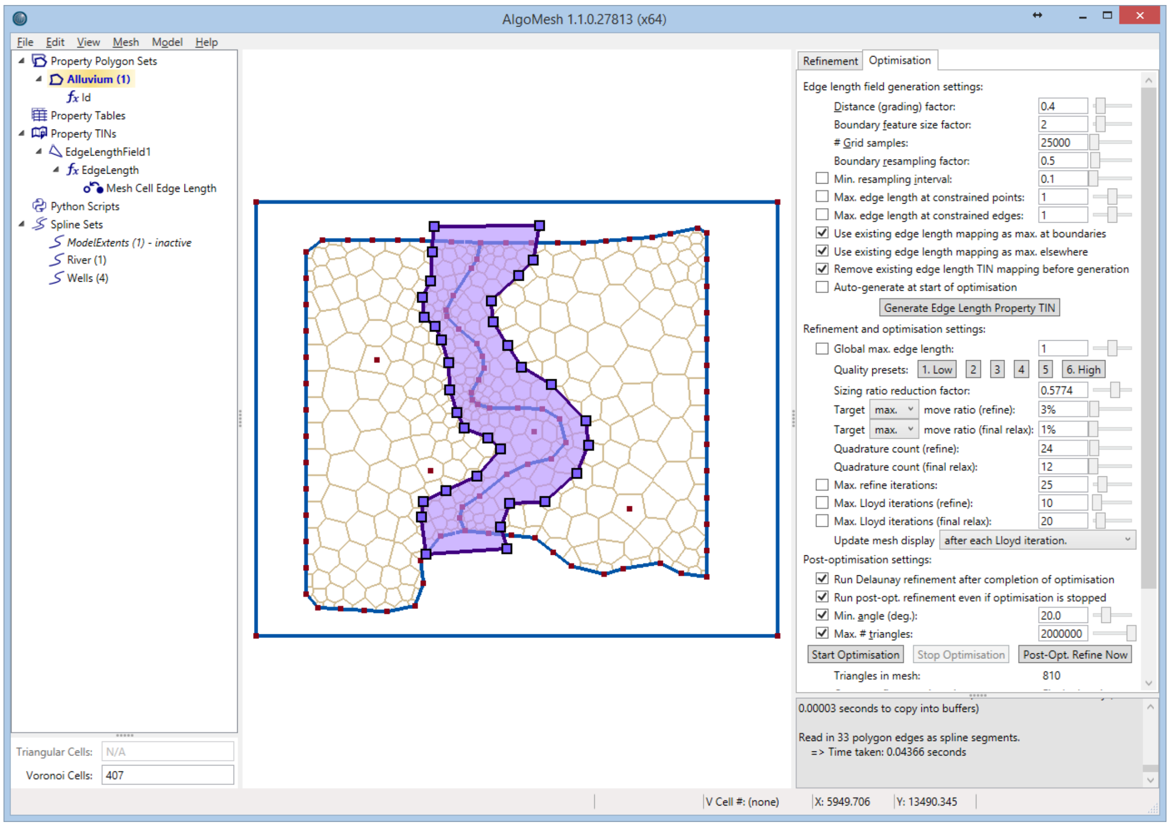
Refining Cells in the Alluvium around the River

First, we are going to bring in a polygon representing an alluvial region around the river in our model. Suppose that we would like to restrict the size of cells within this region to at most 150 metres in diameter to promote accurate simulation of water flows in the alluvium.

Note: although we refer here to the diameter of cells, AlgoMesh actually restricts cell sizing based on the edge length of triangles in the underlying triangular mesh. In a Voronoi mesh, this equates to the distance between centres of adjacent mesh cells, which is not truly a diameter. In a perfectly uniform hexagonal Voronoi mesh, this distance would be the perpendicular distance between two opposite sides of each hexagon. Also note that although we refer to measurements in metres in this model, the actual measurement unit depends entirely on the projection and coordinate system of your data. AlgoMesh will function in the same way regardless of the units used, as long as they are consistent in your imported data.

- **Right-click** on the **Property Polygon Sets** header in the project explorer on the left-hand side of the AlgoMesh window.
- Choose **Import GIS Polygons** from the context menu that appears.
- Select **Alluvium.shp** and click **Open**.

An entry for the Alluvium polygon set (GIS layer) appears under Property Polygon Sets, and the alluvium polygon it contains is shown in the main view (Figure 53).



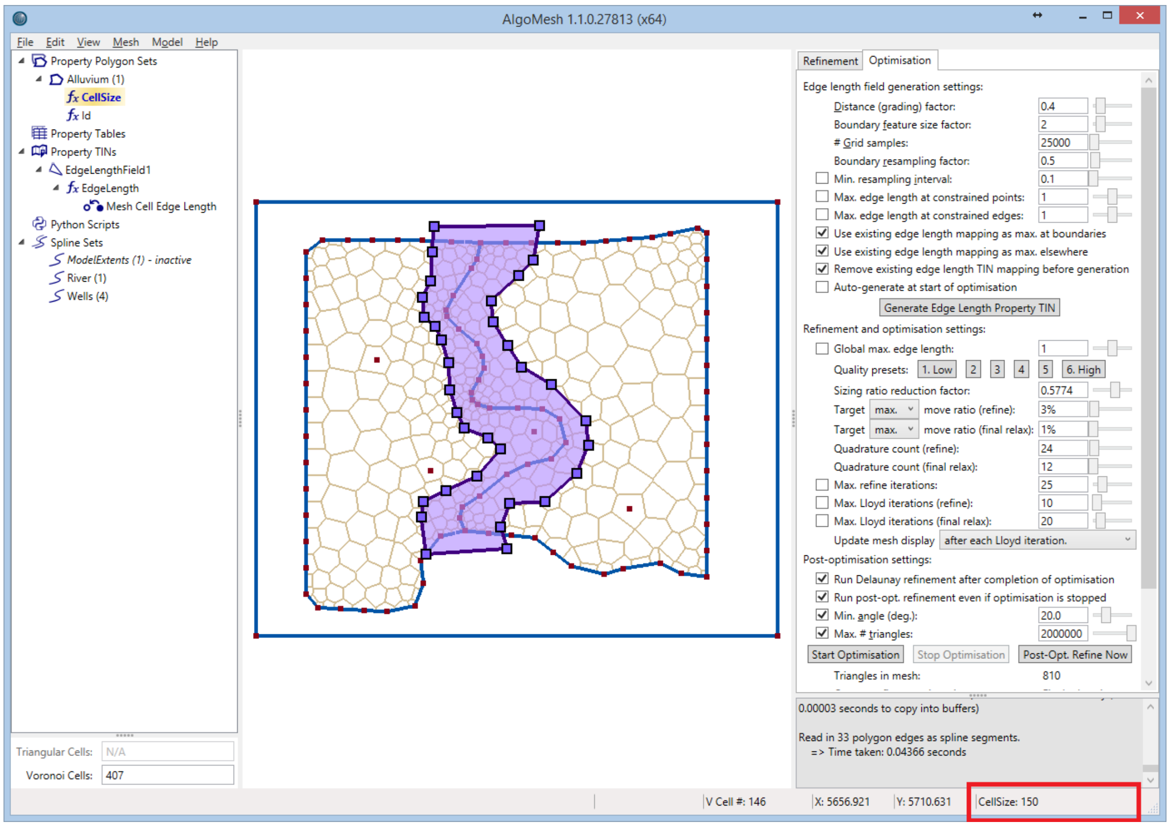
```
Edge length field generation settings:
    Distance (grading) factor:                          0.4
    Boundary feature size factor:                       2
    # Grid samples:                                     25000
    Boundary resampling factor:                         0.5
 ☐ Min. resampling interval:                            0.1
 ☐ Max. edge length at constrained points:              1
 ☐ Max. edge length at constrained edges:               1
 ☑ Use existing edge length mapping as max. at boundaries
 ☑ Use existing edge length mapping as max. elsewhere
 ☑ Remove existing edge length TIN mapping before generation
 ☐ Auto-generate at start of optimisation
Refinement and optimisation settings:
 ☐ Global max. edge length:                             1
    Sizing ratio reduction factor:                      0.5774
    Target max. move ratio (refine):                    3%
    Target max. move ratio (final relax):               1%
    Quadrature count (refine):                          24
    Quadrature count (final relax):                     12
 ☐ Max. refine iterations:                              25
 ☐ Max. Lloyd iterations (refine):                      10
 ☐ Max. Lloyd iterations (final relax):                 20
    Update mesh display  after each Lloyd iteration.
Post-optimisation settings:
 ☑ Run Delaunay refinement after completion of optimisation
 ☑ Run post-opt. refinement even if optimisation is stopped
 ☑ Min. angle (deg.):                                   20.0
 ☑ Max. # triangles:                                    2000000
    Triangles in mesh:                                  810

0.00003 seconds to copy into buffers)

Read in 33 polygon edges as spline segments.
  => Time taken: 0.04366 seconds

Triangular Cells: N/A
Voronoi Cells: 407
V Cell #: (none)   X: 5949.706   Y: 13490.345
```

Figure 53: The alluvium polygon overlays the mesh in the main view after importing it from the shapefile.

Now that we have brought the shape of the alluvial region into AlgoMesh, we need to create a property (GIS field) attached to it to specify the maximum cell size we would like inside the region.

- **Right-click** the **Alluvium** polygon set.
- Choose **New Property** from the context menu that appears.

- ➢ In the New Property window, **give the property the name CellSize**, and a default value of **150**, and **click OK**. An entry for the new CellSize property appears under the Alluvium polygon set. Note that we could have done this earlier in GIS software by adding a field called CellSize to our polygon layer and giving an appropriate value for the alluvium polygon, instead of adding it in AlgoMesh. In general, either method is fine.
- ➢ To verify what we have done, **click** on the **CellSize** property to select it.
- ➢ Now move the mouse cursor over the top of the alluvium polygon in the mesh view. You should see the value 150 displayed in the status bar in the bottom-right corner of the AlgoMesh window when the cursor is on top of the polygon. When the cursor is outside the polygon, you will see "Outside (0)" displayed instead, indicating that this property has no value outside the polygon. Figure 54 shows the property value displayed in the status bar, outlined in red on the figure.



Triangular Cells: N/A
Voronoi Cells: 407

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☐ Max. edge length at constrained points: 1
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation

Refinement and optimisation settings:
☐ Global max. edge length: 1
Quality presets:
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000

Triangles in mesh: 810

```
0.00003 seconds to copy into buffers)

Read in 33 polygon edges as spline segments.
  => Time taken: 0.04366 seconds
```

V Cell #: 146 | X: 5656.921 | Y: 5710.631 | CellSize: 150

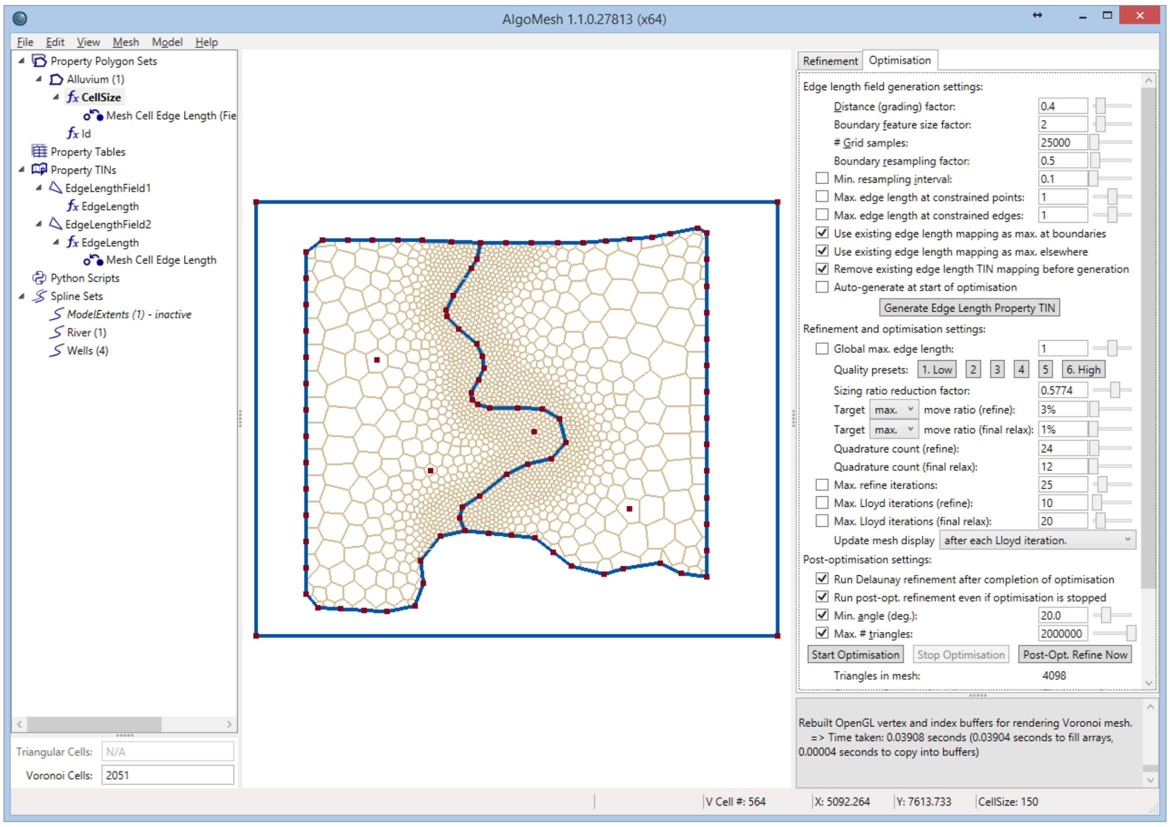
Figure 54: The value of the currently selected property at the mouse cursor position is shown on the right-hand side of the status bar.

Having verified that we are seeing the correct property value within the alluvium, we can now tell AlgoMesh that this property's value should be used as a maximum edge length value for mesh cells.

- ➢ **Right-click** the **CellSize** property under the Alluvium polygon set.
- ➢ From the context menu that appears, choose **Map to Mesh Variable as Overlay**, and then **Mesh Cell Edge Length (Field Generation Only)**. An entry appears under CellSize showing the new property mapping.
- ➢ In the Optimisation tab, under Edge length field generation settings, **turn on Auto-generate at start of optimisation**.
- ➢ Click the **Start Optimisation** button and wait for mesh generation to complete.
- ➢ In the **View** menu, turn off **Property Polygons** so you can better see the cells in the generated mesh.

You should now have a mesh in which the refinement to cells of less than 150 metres in size is clear in the alluvial region around the river, as in Figure 55. The polygon used for the alluvium did not need to be wholly contained within the mesh domain; the parts lying within the domain are used to restrict cell size, and the parts outside are ignored.

Note that when specifying cell sizes in this way, AlgoMesh treats any value given as a maximum edge length in the underlying triangular mesh, and cells will normally not take precisely this size (but the resulting edge lengths will always be under the given value). It is typical for cells to be generated on average about 20-30% smaller than the given value. If you want cells to more closely match a specific size, it is advisable to increase your maximum cell edge length value by 30% or so, measure the resulting cell sizes, and tweak the value from there as necessary. Alternatively, it is possible in open, boundary-free areas of the mesh domain to generate a structured sub-grid with a precise cell size. This is not detailed in this how-to; see the Structured Sub-Grid Generation advanced topic for more information.



Triangular Cells: N/A
Voronoi Cells: 2051

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☐ Max. edge length at constrained points: 1
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation

Refinement and optimisation settings:
☐ Global max. edge length: 1
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000
Triangles in mesh: 4098

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
    => Time taken: 0.03908 seconds (0.03904 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

V Cell #: 564 | X: 5092.264 | Y: 7613.733 | CellSize: 150

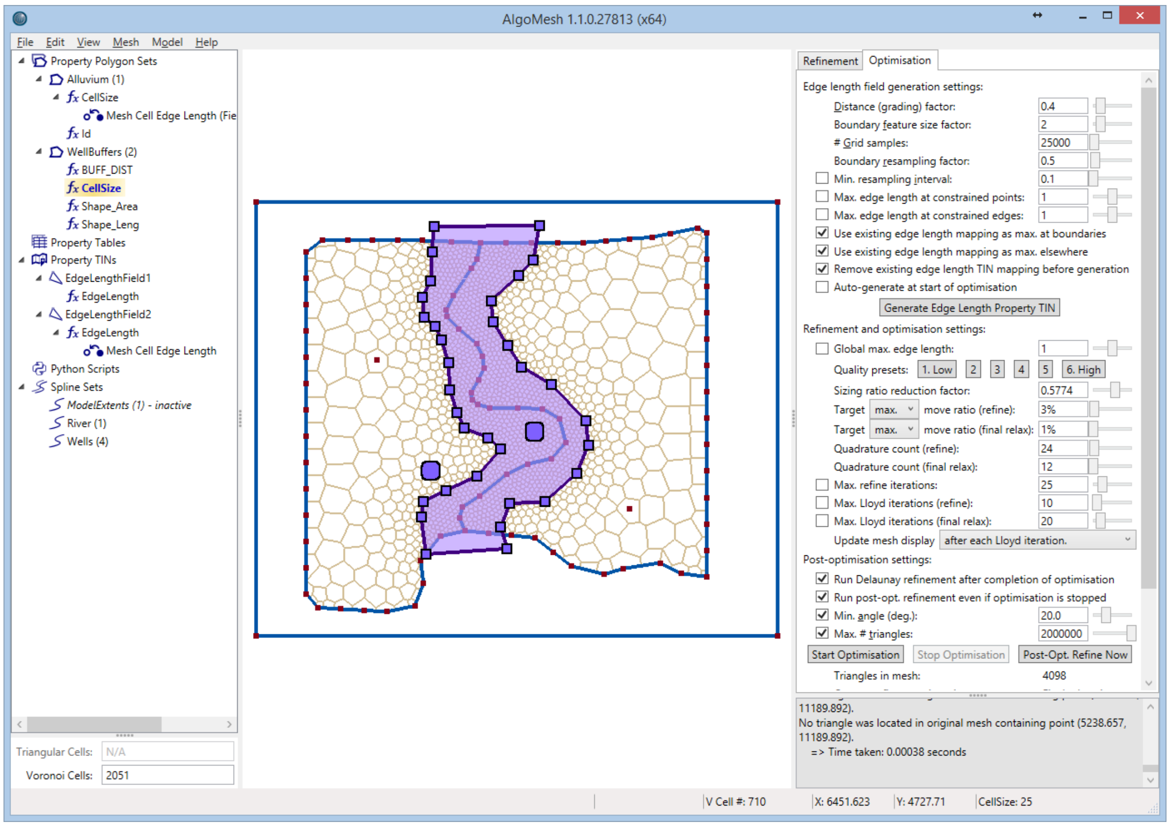
Figure 55: The generated mesh after restricting cell edge length to a maximum of 150 in the alluvium.

Selectively Refining Cells around Wells

Suppose that the two points closest to the river in our model input geometry are pumping wells, and the remaining two are purely observation locations. We may want to refine around the pumping wells, but not around the observation points. We can do this by introducing a buffered polygon a small distance around each of the pumping wells, and mapping an appropriate cell size for each.

➢ In the **View** menu, turn back on **Property Polygons**.
➢ **Right-click** on the **Property Polygon Sets** header.
➢ Choose **Import GIS Polygons** from the context menu that appears.

- Select **WellBuffers.shp** and click **Open**. The WellBuffers layer is added as another property polygon set, beneath the Alluvium set, and two rounded buffer polygons can be seen around the two pumping wells in the main view, as in Figure 56.
- **Right-click** the **WellBuffers** polygon set.
- Choose **New Property** from the context menu that appears.
- In the New Property window, **give the property the name CellSize**, and a default value of **25**, and **click OK**. An entry for the new CellSize property appears under the Alluvium polygon set.
- To verify what we have done, **click** on the **CellSize** property to select it and move the mouse over each of the buffer polygons; the CellSize property should show up with the value 25 in the status bar.



Triangular Cells: N/A
Voronoi Cells: 2051

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☐ Max. edge length at constrained points: 1
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation

Refinement and optimisation settings:
☐ Global max. edge length: 1
Quality presets: 1. Low 2 3 4 5 6. High
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000

Triangles in mesh: 4098

```
11189.892).
No triangle was located in original mesh containing point (5238.657,
11189.892).
  => Time taken: 0.00038 seconds
```

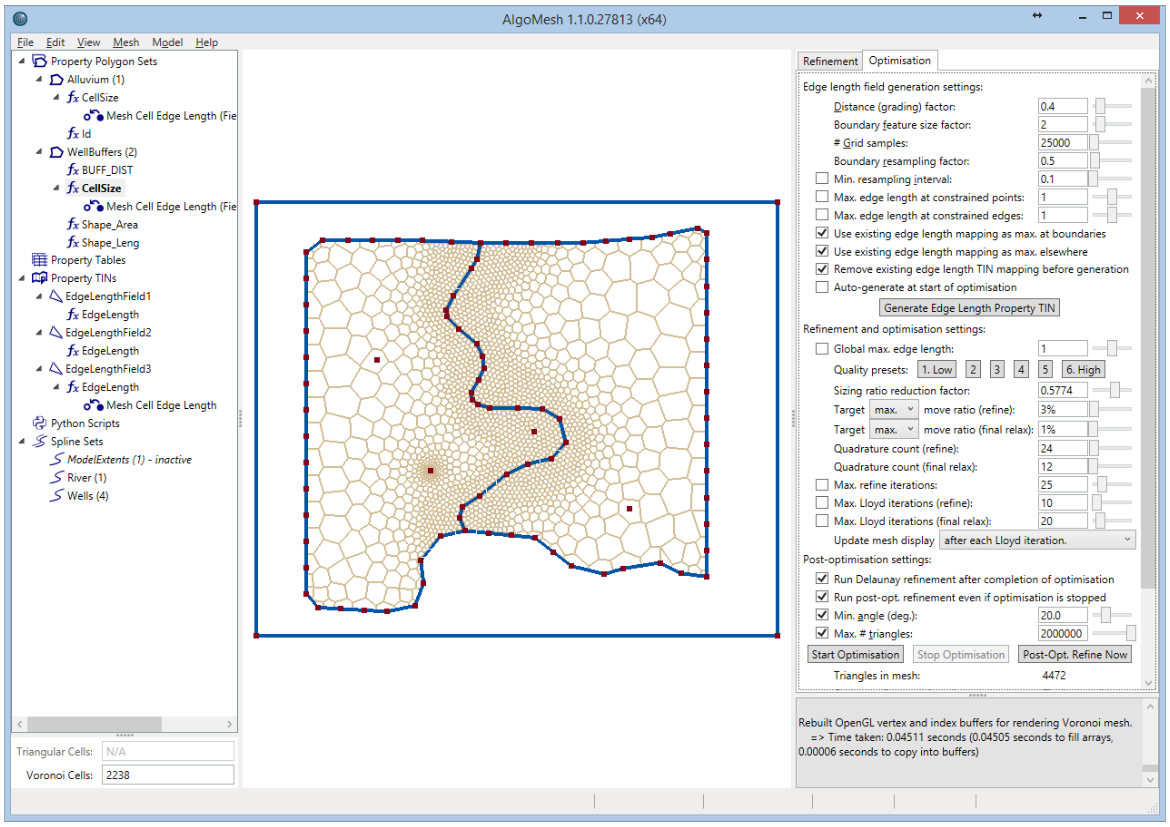
V Cell #: 710 | X: 6451.623 | Y: 4727.71 | CellSize: 25

Figure 56: The river mesh with the newly added well buffer polygons overlying it.

You can now map the cell size property you just created to Mesh Cell Edge Length as before.

- **Right-click** the **CellSize** property under the WellBuffers polygon set.
- From the context menu that appears, choose **Map to Mesh Variable as Overlay**, and then **Mesh Cell Edge Length (Field Generation Only)**. An entry appears under CellSize showing the new property mapping.
- In the **View** menu, turn off **Property Polygons** so you can better see the cells in the generated mesh.
- In the Optimisation tab, under Edge length field generation settings, **turn on Auto-generate at start of optimisation**.
- Click the **Start Optimisation** button and wait for mesh generation to complete.

You should now have a mesh looking like the one in Figure 57.



Triangular Cells: N/A
Voronoi Cells: 2238

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☐ Max. edge length at constrained points: 1
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation

Refinement and optimisation settings:
☐ Global max. edge length: 1
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000

Triangles in mesh: 4472

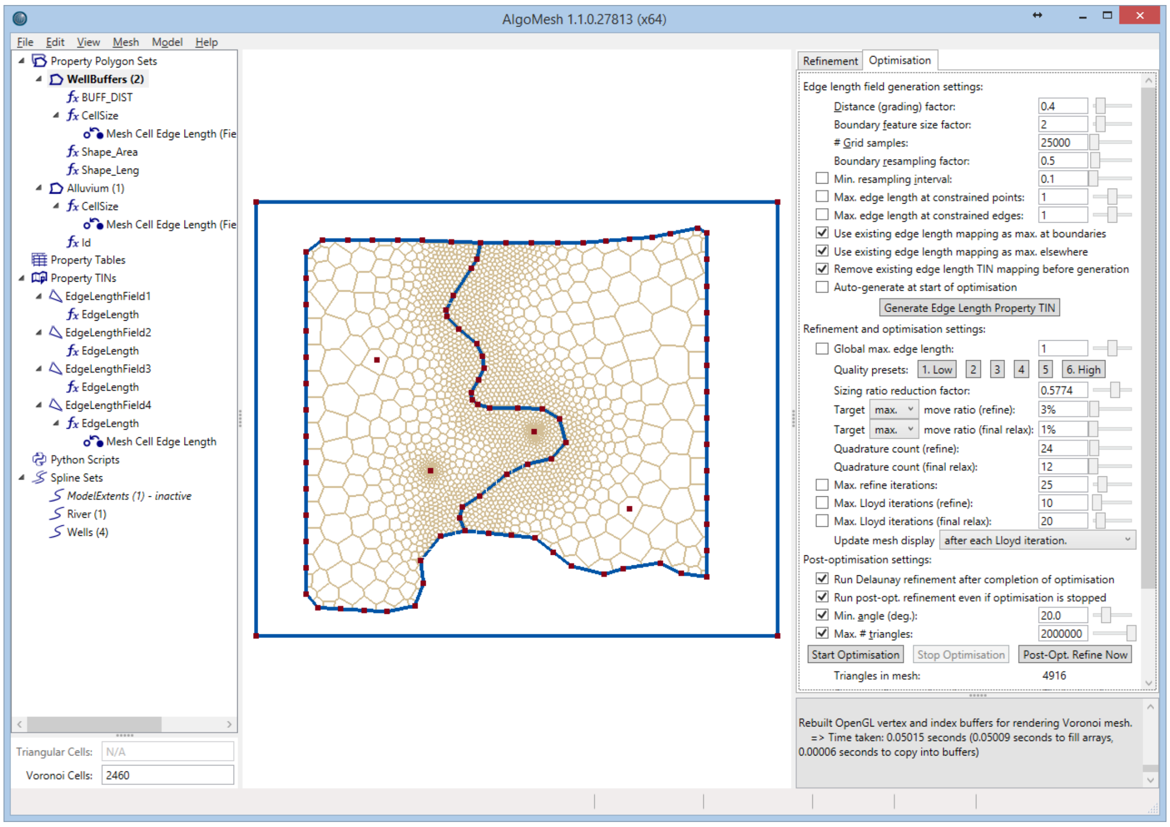
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
=> Time taken: 0.04511 seconds (0.04505 seconds to fill arrays, 0.00006 seconds to copy into buffers)

Figure 57: The first attempt at generating a mesh with refinement down to 25m around the two pumping wells.

Note in the resulting mesh that only one of the two wells - the one outside the alluvium - appears to have been refined as expected. This is because of the ordering of the polygon sets: the Alluvium set is listed above the WellBuffers set under Property Polygon Sets. Since the alluvium polygon and the well buffer polygon overlap, the higher one in the list takes precedence. In our case, we want the well buffer polygon to take precedence instead, as we would like a higher degree of refinement immediately around the well than what is currently being generated in the alluvium. To fix this, we simply need to switch the order of the polygon sets.

- **Right-click** the **WellBuffers** polygon set.
- Choose **Move Up** from the context menu that appears. The WellBuffers set moves to take a place at the top of the list, and the Alluvium set moves below it.
- In the Optimisation tab, under Edge length field generation settings, **turn on Auto-generate at start of optimisation**.
- Click the **Start Optimisation** button and wait for mesh generation to complete.

The mesh is now generated with the correct order of precedence for our cell sizing polygons, and both wells show refinement around them as expected (Figure 58).



Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	2
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
☐	Max. edge length at constrained points:	1
☐	Max. edge length at constrained edges:	1
✓	Use existing edge length mapping as max. at boundaries	
✓	Use existing edge length mapping as max. elsewhere	
✓	Remove existing edge length TIN mapping before generation	
☐	Auto-generate at start of optimisation	

Generate Edge Length Property TIN

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Quality presets: 1. Low 2 3 4 5 6. High	
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

✓	Run Delaunay refinement after completion of optimisation	
✓	Run post-opt. refinement even if optimisation is stopped	
✓	Min. angle (deg.):	20.0
✓	Max. # triangles:	2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 4916

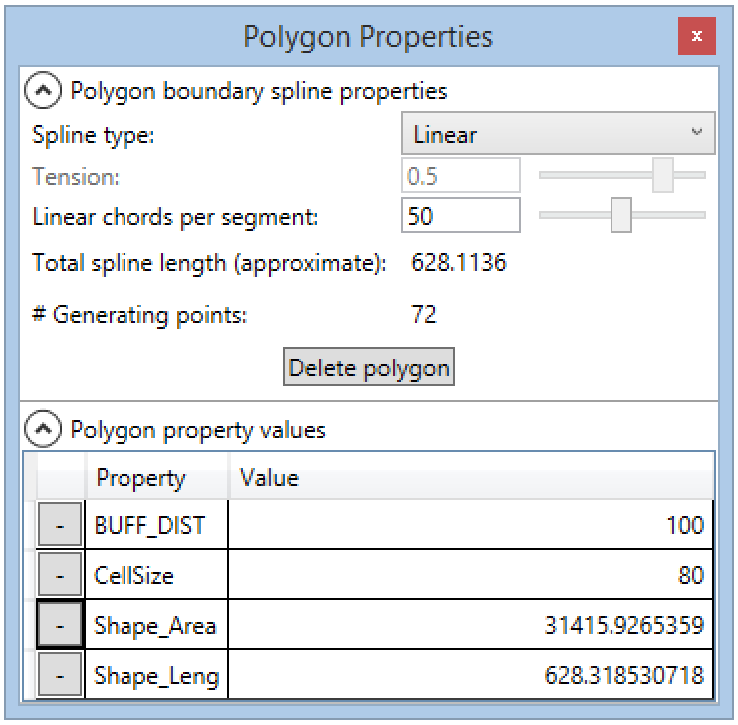
```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.05015 seconds (0.05009 seconds to fill arrays,
0.00006 seconds to copy into buffers)
```

Figure 58: Adjusting the property polygon set ordering causes the well buffer polygons to take precedence, resulting in refinement around both pumping wells as desired.

Note that you can use AlgoMesh's simple mouse controls to pan and zoom the main view if you need to see more detail. With the mouse positioned over a point of interest, move the mouse's scroll wheel upwards to zoom in, or downwards to zoom out. Click and drag with the left mouse button in an empty area of the view to pan around. To restore the default view showing the entire mesh, go to the View menu and select Zoom to Extents.

Finally, you may want to set different cell edge length values for different polygons in the same set. You can either do this by creating the cell size field beforehand in GIS software and setting a different field value for each polygon, or you can set a different value for each of the polygons directly in AlgoMesh. We will do the latter here to relax the cell sizing on the western pumping well to 80m.

- ➢ In the **View** menu, turn back on **Property Polygons**.
- ➢ In the **Edit** menu, switch to **Edit Property Polygons** mode.
- ➢ **Move your mouse cursor** over the **western** pumping well buffer polygon, and **right-click**. A Polygon Properties window should appear.
- ➢ Under Polygon property values, there is a default value entry for CellSize of 25. Click the **+ button** next to this to override the default with a specific value for this polygon.
- ➢ **Click on the 25** value listed for CellSize and **change it to 80**. Your Polygon Properties window should look like that shown in Figure 59.



Polygon boundary spline properties

Spline type:	Linear
Tension:	0.5
Linear chords per segment:	50
Total spline length (approximate):	628.1136
# Generating points:	72

Polygon property values

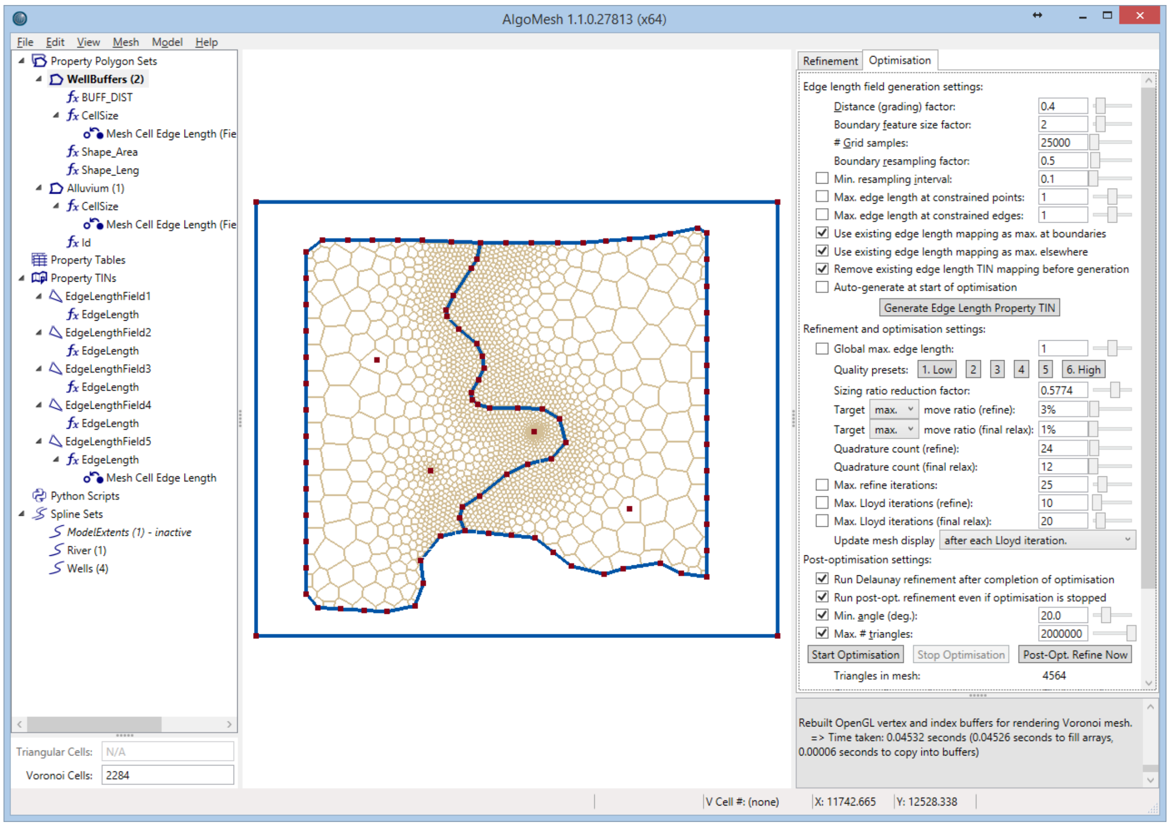
Property	Value
BUFF_DIST	100
CellSize	80
Shape_Area	31415.9265359
Shape_Leng	628.318530718

Figure 59: The polygon properties window of the western pumping well, with CellSize changed to 80.

We can once again regenerate the mesh and evaluate the effect of the cell size change.

➢ **Click the red X** in the top-right corner of the Polygon Properties window to close it.
➢ In the **View** menu, turn off **Property Polygons** so you can better see the cells in the generated mesh.
➢ In the Optimisation tab, under Edge length field generation settings, **turn on Auto-generate at start of optimisation**.
➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

The complete mesh is shown in Figure 60. You can see that the refinement around the western pumping well is no longer so heavy, as the cells are now only restricted to an 80m size, rather than 25m.



Triangular Cells: N/A
Voronoi Cells: 2284

Edge length field generation settings:

Distance (grading) factor:	0.4
Boundary feature size factor:	2
# Grid samples:	25000
Boundary resampling factor:	0.5
☐ Min. resampling interval:	0.1
☐ Max. edge length at constrained points:	1
☐ Max. edge length at constrained edges:	1
☑ Use existing edge length mapping as max. at boundaries	
☑ Use existing edge length mapping as max. elsewhere	
☑ Remove existing edge length TIN mapping before generation	
☐ Auto-generate at start of optimisation	

Refinement and optimisation settings:

☐ Global max. edge length:	1
Quality presets:	1. Low, 2, 3, 4, 5, 6. High
Sizing ratio reduction factor:	0.5774
Target max. move ratio (refine):	3%
Target max. move ratio (final relax):	1%
Quadrature count (refine):	24
Quadrature count (final relax):	12
☐ Max. refine iterations:	25
☐ Max. Lloyd iterations (refine):	10
☐ Max. Lloyd iterations (final relax):	20
Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

☑ Run Delaunay refinement after completion of optimisation	
☑ Run post-opt. refinement even if optimisation is stopped	
☑ Min. angle (deg.):	20.0
☑ Max. # triangles:	2000000
Triangles in mesh:	4564

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.04532 seconds (0.04526 seconds to fill arrays,
0.00006 seconds to copy into buffers)
```

V Cell #: (none) | X: 11742.665 | Y: 12528.338

Figure 60: The complete mesh, with each of the two pumping wells refined to different maximum cell sizes.

Although the polygons used in this tutorial were all imported from shapefiles, you can also manually digitise polygons in AlgoMesh for the same purpose. Furthermore, the boundaries of the polygon share some similar properties to the splines used for linear feature geometry, which may be adjusted manually. The Manually Digitise and Edit Splines how-to guide goes into these processes in depth.

3.7 Manually Digitise and Edit Splines

This how-to details the process of manually adding splines to a mesh domain, and explores some of the settings that affect how each spline is sampled. We will be using triangular grids to illustrate, but the procedure is the same regardless of whether you will be generating triangular or Voronoi grids.

It is recommended that you first go through the Import Geometry from GIS tutorial if you have not already, as it introduces some of the basic knowledge that is assumed in this section.

Overview of Splines

The size of mesh elements generated by AlgoMesh along any feature in the mesh domain is influenced by the spacing between input points along the feature. If the input geometry includes polylines that are too finely sampled, an excessive quantity of cells may be generated near those polylines. Conversely, it can sometimes be useful to increase the density of input points along a feature to promote refinement into smaller elements.

In the context of AlgoMesh, a spline is a curve that is generated to fit through a given sequence of points (i.e. a polyline). For smoothly curving features, AlgoMesh uses so-called cardinal cubic splines, which are sequences of cubic curves fitting the input points to create a smoothly curving line (continuous in the first derivative), with a controllable degree of tension. These curves are sampled to produce points and line segments in the mesh domain prior to mesh generation, with the spacing between points manually controllable through parameters set on each spline.

Splines are not only useful for smoothly curving features, however; they may also be used to generate purely linear polylines, as an alternative to adding points directly from GIS into a mesh domain. This is useful for controlling the spacing between subsequent points in the polyline, which in turn affects the sizing of elements produced by the mesh generator.

AlgoMesh creates linear polylines by default when splines are added or imported, but any one or more of these may be changed to a cubic spline curve easily by altering their spline properties.

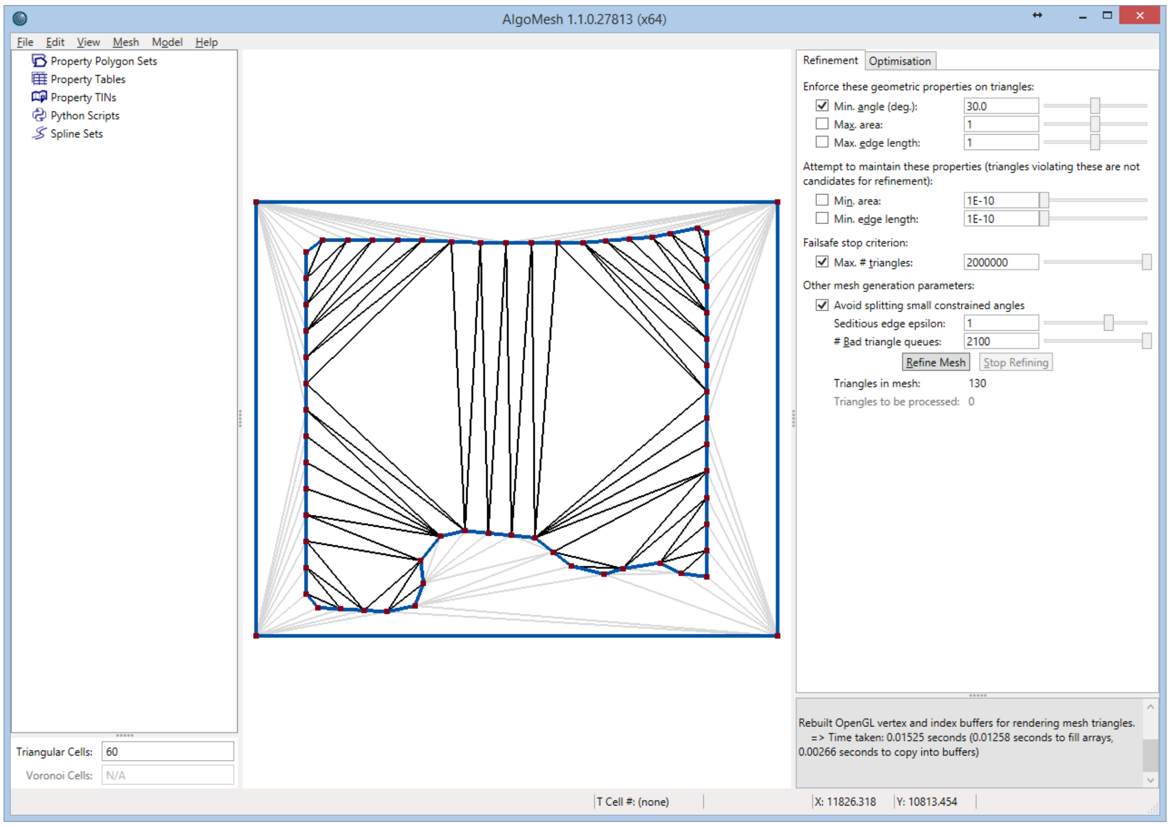
Spline sets may also host individual points, which may be used to represent groundwater wells, for example.

Manually Adding a River Spline and Wells

Start AlgoMesh and load up the example project mesh-domain-without-river.amproj.

- Go to the **File** menu and choose **Open Project**.
- Browse to the **Samples** folder of your AlgoMesh installation.
- Select **mesh-domain-without-river.amproj** and click **Open**.

AlgoMesh shows a triangulated empty mesh domain as in Figure 61. We are going to add a spline for a river through the centre of this mesh domain, and a few points representing wells.



Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.01525 seconds (0.01258 seconds to fill arrays, 0.00266 seconds to copy into buffers)

Triangular Cells: 60
Voronoi Cells: N/A

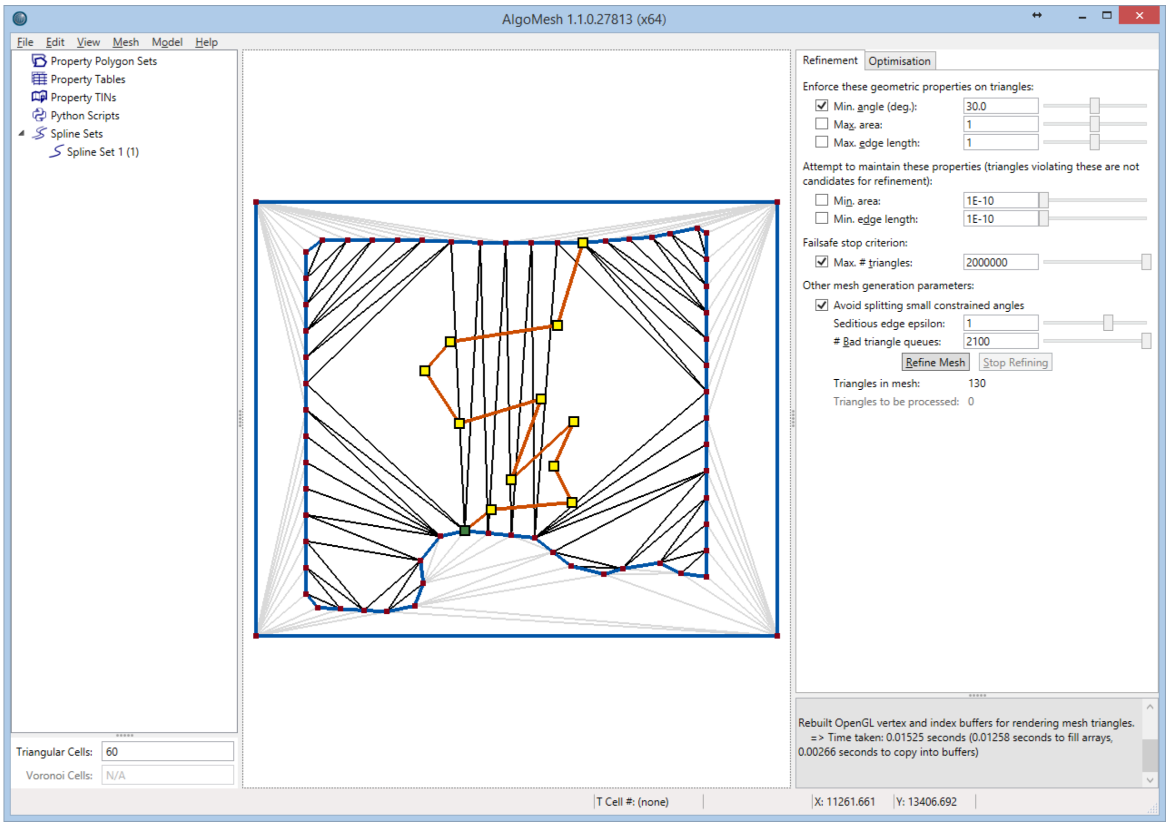
T Cell #: (none) | X: 11826.318 | Y: 10813.454

Figure 61: The initial triangulated mesh domain.

We will now switch to AlgoMesh's Edit Splines mode and create a river spline using some basic mouse and keyboard controls.

- In the **Edit** menu, choose **Edit Splines** to enable this mode. This will change what happens when we click inside the main view.
- Holding down **Alt** on the keyboard, click with the **left mouse button** on one of the vertices along the top of the mesh extents polygon to start a new spline snapped to this point. See Figure 62 for reference.
- Holding down both **Ctrl** and **Shift**, **left-click** a sequence of points to produce a shape for the river similar to the one shown in the figure.
- To add the last point, hold down both **Alt** and **Shift** when clicking instead, to ensure that the point is snapped to the vertex at the bottom of the mesh extents polygon.

You should now have a linear polyline for the river similar to that shown in the figure.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.01525 seconds (0.01258 seconds to fill arrays,
0.00266 seconds to copy into buffers)
```

Triangular Cells: 60
Voronoi Cells: N/A
T Cell #: (none)
X: 11261.661
Y: 13406.692

Figure 62: The manually-digitised river polyline.

Clicking in the view in Edit Splines mode has a different effect depending on the Ctrl, Shift and Alt modifier keys that are held down:

- **Click (no modifier keys)**: Selects a point or segment of an existing spline under the mouse cursor. Clicking and dragging allows moving a spline point to a new location.
- **Ctrl+click**: Starts a new spline by creating a new point at the location of the mouse cursor, and selects the new point. If the mouse cursor is on top of a segment of an existing spline, the new point will snap to the existing spline, and it will be inserted between the two endpoints of the segment, instead of creating a new spline.
- **Alt+click**: Starts a new spline by creating a new point at the location of the nearest existing vertex in the underlying mesh, and selects the new point. This is useful for snapping to existing mesh geometry.
- **Shift+click**: Adds a new segment to an existing spline from the selected spline point to the spline point under the mouse cursor. Does nothing if no spline point is selected, or if the mouse is not on top of another spline point.
- **Ctrl+Shift+click**: Creates a new point as per Ctrl-click and adjoins it with a spline segment from the selected point.
- **Alt+Shift+click**: Creates a new point snapped to a mesh vertex as per Alt-click and adjoins it with a spline segment from the selected point.

If you make a mistake, you can move a point by clicking and dragging it, or delete a point by clicking it to select it and then pressing the **Delete** key.

Notice that a new entry has been created in the list on the left-hand side of the AlgoMesh window called Spline Set 1. A spline set is simply a collection of splines, which can be named and made active or inactive. By default, AlgoMesh will add any splines you create into the first active spline set.

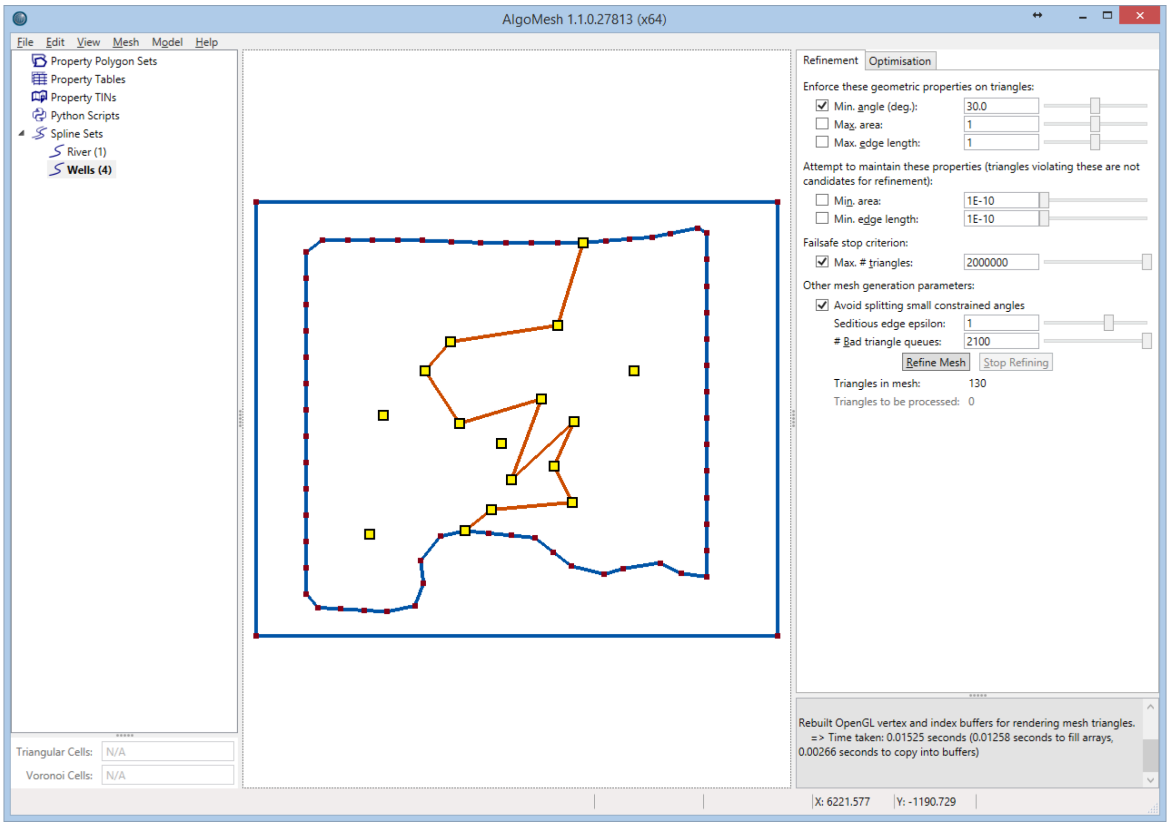
It will create one if no such set exists (as it did here). However, you can also create new, empty spline sets to assist in organisation, when you have splines representing many different features in a model. If an active spline set is selected (bold) in the list when a new spline is created, the new spline will belong to the selected set, instead of the default.

Let us rename this new spline set so that we can easily identify it.

- **Right-click** on **Spline Set 1**.
- Choose **Rename** from the context menu that appears.
- Type **River** and hit **Enter**. The set name is changed to River in the list.

We will now create some single-point splines to represent the wells in our model, hosting them in a second spline set called Wells.

- First, in the **View** menu, turn off **Triangles** so that you can better see the splines in the mesh domain.
- **Right-click** on the **Spline Sets** header.
- Choose **New Spline Set** from the context menu that appears. Another spline set named **Spline Set 1** appears.
- **Right-click** on this new spline set, **Spline Set 1**.
- Choose **Rename** from the context menu that appears.
- Type **Wells** and hit **Enter**. The set name is changed to Wells in the list.
- The **Wells** spline set should now be selected (showing in bold type and highlighted background); **left-click** it to select it if it is not.
- Now **create a few wells** in the model by **Ctrl-clicking** once for each well. Create at least four wells as shown in Figure 63.



Triangular Cells: N/A
Voronoi Cells: N/A

Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.01525 seconds (0.01258 seconds to fill arrays,
0.00266 seconds to copy into buffers)
```

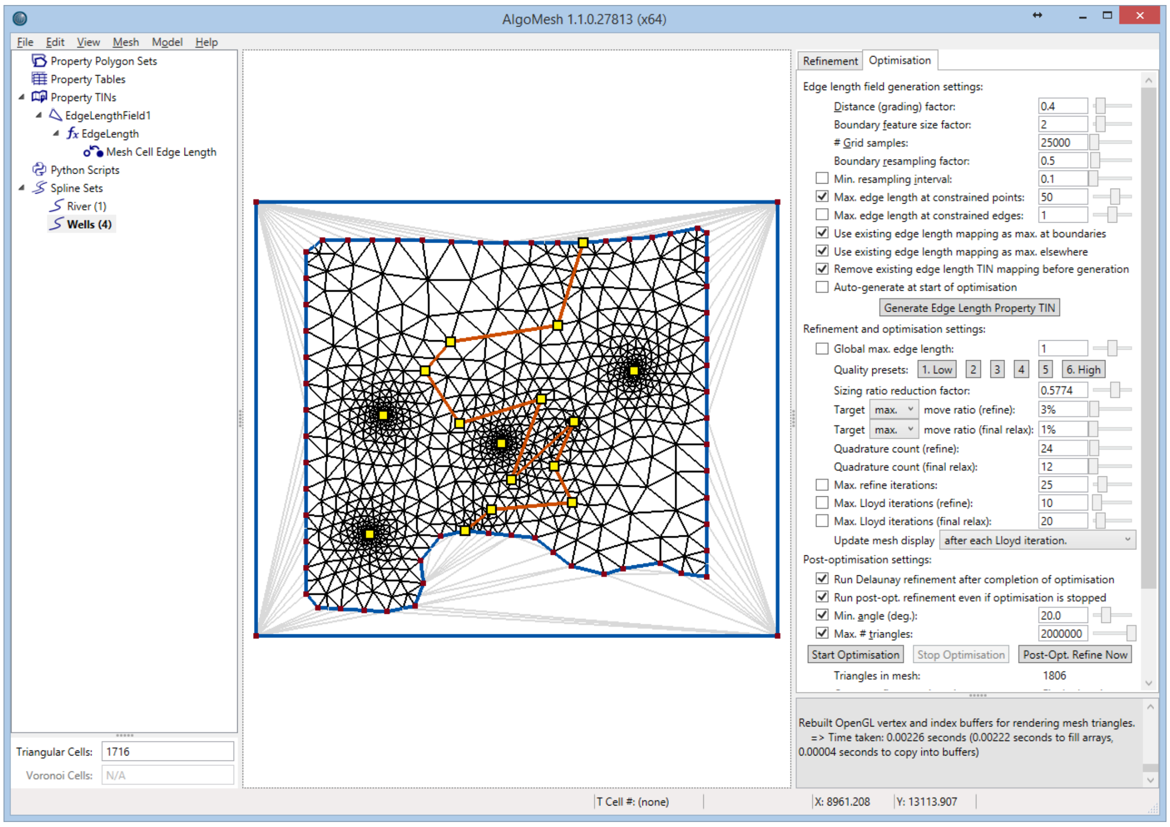
X: 6221.577 Y: -1190.729

Figure 63: The wells that were manually added to the Wells spline set are shown in their respective locations around the river.

Now we can try generating a mesh using our spline features.

- ➢ In the **View** menu, turn back on **Triangles**.
- ➢ Select the **Optimisation** tab.
- ➢ Tick the box next to **Max. edge length at constrained points**.
- ➢ Set the value of **Max. edge length at constrained points** to **50**. This will ensure that the mesh has some refinement around each of our wells.
- ➢ Click the **Start Optimisation** button and wait for mesh generation to complete.

You should now see a mesh similar to Figure 64.



Triangular Cells: 1716
Voronoi Cells: N/A

Edge length field generation settings:

	Distance (grading) factor:	0.4
	Boundary feature size factor:	2
	# Grid samples:	25000
	Boundary resampling factor:	0.5
☐	Min. resampling interval:	0.1
☑	Max. edge length at constrained points:	50
☐	Max. edge length at constrained edges:	1
☑	Use existing edge length mapping as max. at boundaries	
☑	Use existing edge length mapping as max. elsewhere	
☑	Remove existing edge length TIN mapping before generation	
☐	Auto-generate at start of optimisation	

Generate Edge Length Property TIN

Refinement and optimisation settings:

☐	Global max. edge length:	1
	Quality presets: 1. Low 2 3 4 5 6. High	
	Sizing ratio reduction factor:	0.5774
	Target max. move ratio (refine):	3%
	Target max. move ratio (final relax):	1%
	Quadrature count (refine):	24
	Quadrature count (final relax):	12
☐	Max. refine iterations:	25
☐	Max. Lloyd iterations (refine):	10
☐	Max. Lloyd iterations (final relax):	20
	Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

☑	Run Delaunay refinement after completion of optimisation	
☑	Run post-opt. refinement even if optimisation is stopped	
☑	Min. angle (deg.):	20.0
☑	Max. # triangles:	2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 1806

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00226 seconds (0.00222 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

X: 8961.208 Y: 13113.907

Figure 64: The first attempt at generating a mesh for our manually digitised river domain.

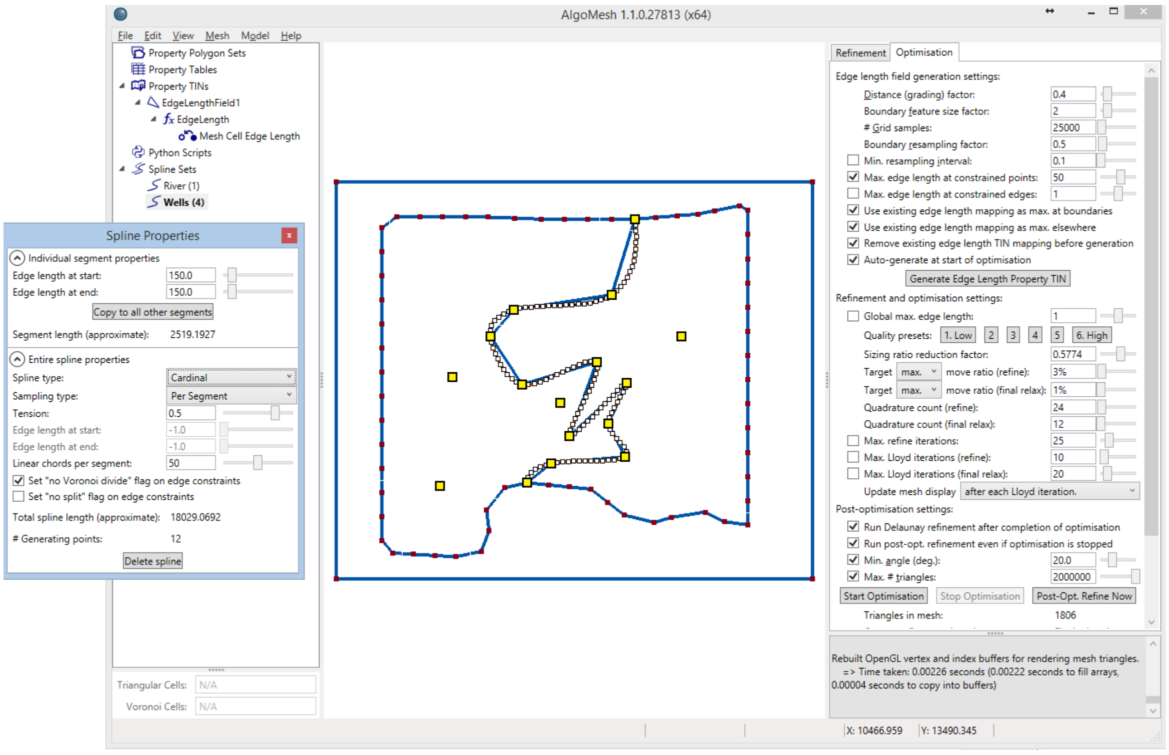
This mesh is not too bad, but the cells along the river vary substantially in size, and the line of the river itself is quite coarse. We can adjust some of the river's spline properties to remedy this.

Altering the River's Spline Properties

By default, AlgoMesh samples splines as linear polylines, and only the vertices of the polyline (the yellow squares, which are also called *generating points*) are added into the mesh domain, with line segments between them. Now, we will change this so that AlgoMesh makes a smooth curve running through the generating points, and tell it to resample the curve such that more points are added into the mesh along its length at regular intervals.

- In the **View** menu, turn off **Triangles** so that you have a clear view of the splines.
- **Move the mouse cursor** anywhere **on top of the river spline**; it should highlight when you move close to it.
- **Right-click** to open up the **Spline Properties** window. This shows all the properties that determine the spline's shape, as well as how it is sampled to produce vertices and edges of the mesh.
- Under **Individual segment properties**, change **Edge length at start** to **150**.
- Under **Individual segment properties**, change **Edge length at end** to **150**.
- Under **Individual segment properties**, click the **Copy to all other segments** button.
- Under **Entire spline properties**, change **Spline type** to **Cardinal**.

You should see the river change to a curved spline, with small white rectangles denoting sampling points at a spacing of approximately 150m along the length of the spline, as in Figure 65.



Spline Properties

Individual segment properties
Edge length at start: 150.0
Edge length at end: 150.0
Segment length (approximate): 2519.1927

Entire spline properties
Spline type: Cardinal
Sampling type: Per Segment
Tension: 0.5
Edge length at start: -1.0
Edge length at end: -1.0
Linear chords per segment: 50
☑ Set "no Voronoi divide" flag on edge constraints
☐ Set "no split" flag on edge constraints
Total spline length (approximate): 18029.0692
Generating points: 12

Triangular Cells: N/A
Voronoi Cells: N/A

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☑ Max. edge length at constrained points: 50
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☑ Auto-generate at start of optimisation

Refinement and optimisation settings:
☐ Global max. edge length: 1
Quality presets: 1. Low 2 3 4 5 6. High
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000
Triangles in mesh: 1806

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.00226 seconds (0.00222 seconds to fill arrays, 0.00004 seconds to copy into buffers)

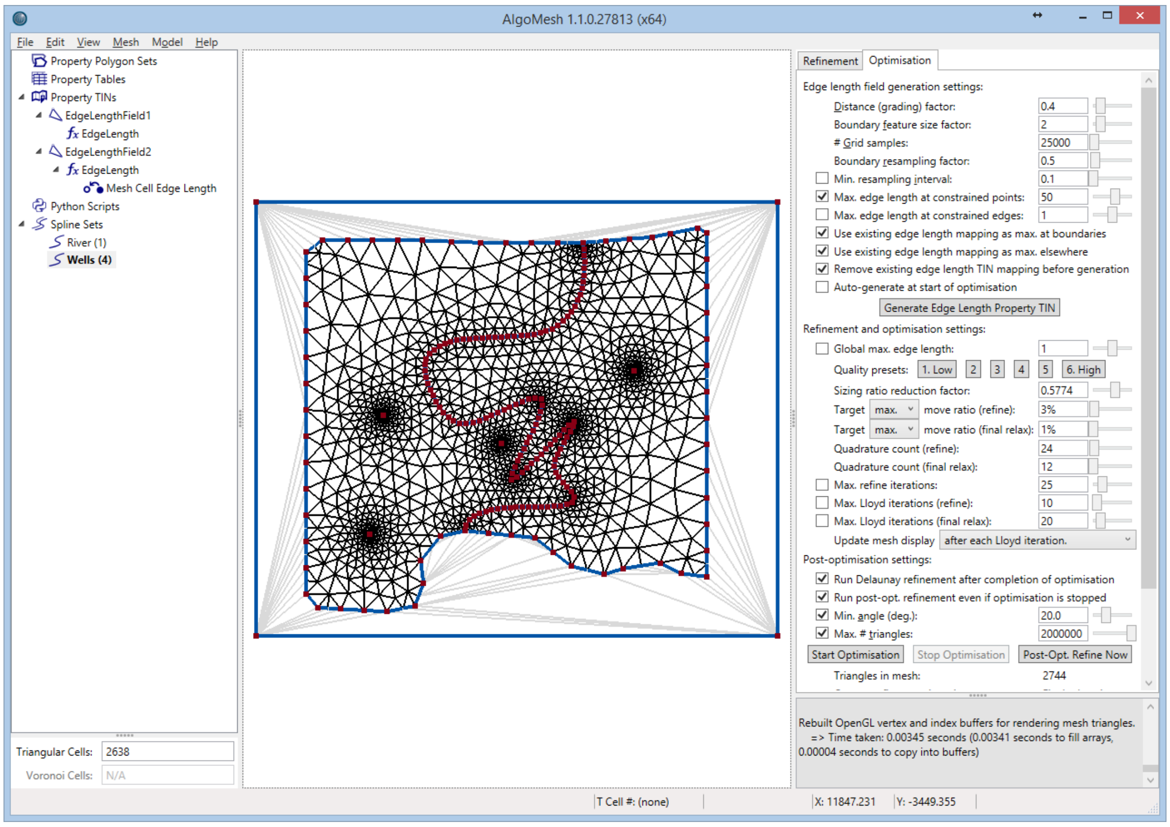
X: 10466.959 Y: 13490.345

Figure 65: The river spline after changing its type to Cardinal and resampling it at 150m.

Regenerate the mesh to see how it looks after adjusting our river spline.

➢ In the **View** menu, turn **Triangles** back on.
➢ In the **Optimisation** tab, click the **Start Optimisation** button and wait for mesh generation to complete.
➢ In the **View** menu, turn off **Splines** so that you can clearly see the generated mesh.

Figure 66 shows the updated mesh. Along most of its length, the triangles generated are now smaller and more uniform in size and shape.



Triangular Cells: 2638
Voronoi Cells: N/A

Edge length field generation settings:

- Distance (grading) factor: 0.4
- Boundary feature size factor: 2
- # Grid samples: 25000
- Boundary resampling factor: 0.5
- [ ] Min. resampling interval: 0.1
- [✓] Max. edge length at constrained points: 50
- [ ] Max. edge length at constrained edges: 1
- [✓] Use existing edge length mapping as max. at boundaries
- [✓] Use existing edge length mapping as max. elsewhere
- [✓] Remove existing edge length TIN mapping before generation
- [ ] Auto-generate at start of optimisation

Refinement and optimisation settings:

- [ ] Global max. edge length: 1
- Quality presets:
- Sizing ratio reduction factor: 0.5774
- Target max. move ratio (refine): 3%
- Target max. move ratio (final relax): 1%
- Quadrature count (refine): 24
- Quadrature count (final relax): 12
- [ ] Max. refine iterations: 25
- [ ] Max. Lloyd iterations (refine): 10
- [ ] Max. Lloyd iterations (final relax): 20
- Update mesh display: after each Lloyd iteration.

Post-optimisation settings:

- [✓] Run Delaunay refinement after completion of optimisation
- [✓] Run post-opt. refinement even if optimisation is stopped
- [✓] Min. angle (deg.): 20.0
- [✓] Max. # triangles: 2000000

Triangles in mesh: 2744

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.00345 seconds (0.00341 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

T Cell #: (none) | X: 11847.231 | Y: -3449.355

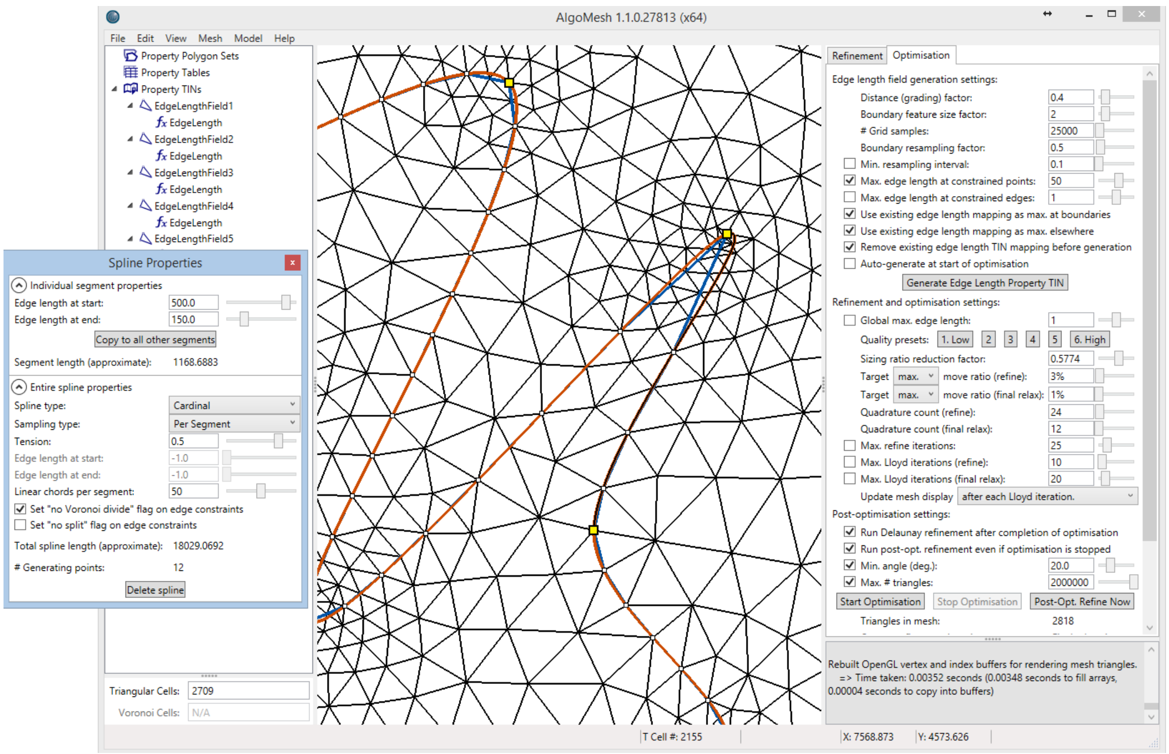
Figure 66: The smoothly-curving and uniformly resampled river spline produces more appropriate refinement along the majority of its length.

Note that there is some over-refinement around the sharper bends in the river. It would be ideal to reduce this, so that we get a more even distribution of cell sizes along the river.

Each spline is logically divided into a series of segments, with each segment running between two successive generating points. The Spline Properties window allows us to change the spacing between points over individual segments separately to the rest of the spline, if desired. We can also use a different spacing at the start of a segment to the end of a segment, varying the edge length along the segment in a geometric progression between the two spacing values. Do this now to increase the spacing between points around one of the sharp bends in the river.

➢ In the **View** menu, turn **Splines** back on.
➢ Use the **mouse scroll wheel** to **zoom in on the sharpest bend** in the river, where you can see a dark spot indicating an abundance of triangles.
➢ **Right-click** on the segment of the spline immediately before the sharp bend (to the west). It should display with a black line through its centre to show that it is selected. The Spline Properties window appears and the **Individual segment properties** section now relates to this particular segment of the spline.
➢ Under **Individual segment properties**, change **Edge length at end** to **500**. You should see the spacing between the white dots on the spline change accordingly in the main view.
➢ **Right-click** on the segment immediately following the same sharp bend (to the east).
➢ Under Individual segment properties, change **Edge length at start** to **500**.
➢ In the **Optimisation** tab, click the **Start Optimisation** button and wait for mesh generation to complete.

Figure 67 shows the result. Although there is still some over-refinement - this is inevitable at such a tight angle in the geometry - the number of triangles has been reduced substantially by reducing the number of sampled points in the vicinity of the bend.



Spline Properties

Individual segment properties
Edge length at start: 500.0
Edge length at end: 150.0
Copy to all other segments
Segment length (approximate): 1168.6883

Entire spline properties
Spline type: Cardinal
Sampling type: Per Segment
Tension: 0.5
Edge length at start: -1.0
Edge length at end: -1.0
Linear chords per segment: 50
☑ Set "no Voronoi divide" flag on edge constraints
☐ Set "no split" flag on edge constraints
Total spline length (approximate): 18029.0692
Generating points: 12
Delete spline

Triangular Cells: 2709
Voronoi Cells: N/A

Edge length field generation settings:
Distance (grading) factor: 0.4
Boundary feature size factor: 2
Grid samples: 25000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☑ Max. edge length at constrained points: 50
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation
Generate Edge Length Property TIN

Refinement and optimisation settings:
☐ Global max. edge length: 1
Quality presets: 1. Low 2 3 4 5 6. High
Sizing ratio reduction factor: 0.5774
Target max. move ratio (refine): 3%
Target max. move ratio (final relax): 1%
Quadrature count (refine): 24
Quadrature count (final relax): 12
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 20.0
☑ Max. # triangles: 2000000
Start Optimisation | Stop Optimisation | Post-Opt. Refine Now
Triangles in mesh: 2818

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00352 seconds (0.00348 seconds to fill arrays,
0.00004 seconds to copy into buffers)
```

X: 7568.873 Y: 4573.626

Figure 67: The interval between sampled points along a spline may be controlled on a per-segment basis, and may be varied smoothly from one length to another.

*Point of interest: if desired, instead of entering distance values directly for **Edge length at start** and **Edge length at end** settings, you can enter a negative number to indicate that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number. So entering **-20** would result in 20 sampled intervals along the spline segment.*

At this point, you may wish to zoom back out to see the entire mesh, and save your work.

➢ Go to the **View** menu, select **Zoom to Extents**.
➢ Go to the **File** menu and choose **Save Project**.
➢ **Browse to a location** to which you have write access, such as your Documents folder.
➢ **Type a name** for the project file to be saved, e.g. **SplineMeshDomain.amproj** and click Save.

The Relationship between Polygons and Splines

AlgoMesh also uses splines to represent the boundary of property polygons. The difference between the two types is that splines are added directly into the mesh, whereas property polygons are not - they are used to influence cell properties within a certain region or to restrict cell sizing in a particular area of the mesh (see the Use Property Polygons to Control Mesh Cell Sizing how-to for more information).

However, the same process that we used in this tutorial to manually digitise spline curves may also be used to digitise property polygons - just use the **Edit Property Polygons** mode instead of the **Edit Splines** mode. Polygon boundaries may also be cardinal cubic spline curves, like we used for the river in this tutorial. Point sampling does not come into play for property polygons, however, as no points are added into the mesh; the polygons are only used to delineate regions within which property values are mapped.

Dealing with Large Numbers of Splines

If you have a large set of polylines imported from GIS - for example, a detailed streamline dataset - it may be too time-consuming to individually select and edit properties for each feature. You can instead right-click on the spline set to which they belong, and choose **Set Properties for All Splines in Set**. This allows you to set up the same properties you manipulated for an individual spline in this tutorial, and apply it to every spline in the set. Note that when altering properties in this way, you must click the **Apply to All in Set** button before any changes you make will take effect.

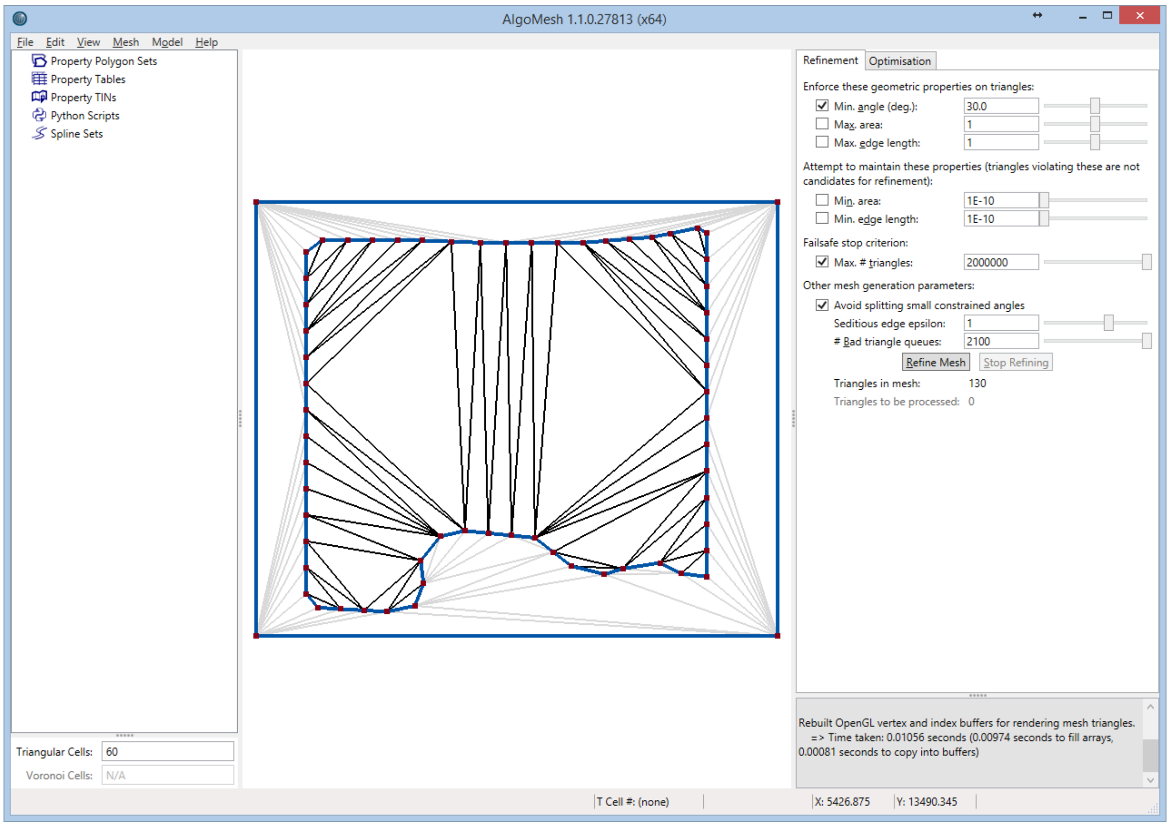
3.8 Manually Modify a Mesh

In addition to importing geometry from GIS, AlgoMesh allows you to manually add points and line segments directly into a mesh. In general, it is recommended to use GIS imports and splines to define mesh input geometry, but in some circumstances it may be useful to directly edit the mesh domain - in particular to mark certain regions of the domain as being inside or outside the mesh (e.g. cutting "holes" out of the mesh). This how-to will teach you about the mouse and keyboard controls you can use in AlgoMesh's Edit Mesh mode to directly manipulate the mesh.

Start by loading the AlgoMesh project mesh-domain-without-river.amproj.

- Go to the **File** menu and choose **Open Project**.
- Browse to the **Samples** folder of your AlgoMesh installation.
- Select **mesh-domain-without-river.amproj** and click **Open**.

AlgoMesh shows a triangulated empty mesh domain, as in Figure 68.



Triangular Cells: 60
Voronoi Cells: N/A

Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 130
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.01056 seconds (0.00974 seconds to fill arrays,
0.00081 seconds to copy into buffers)
```

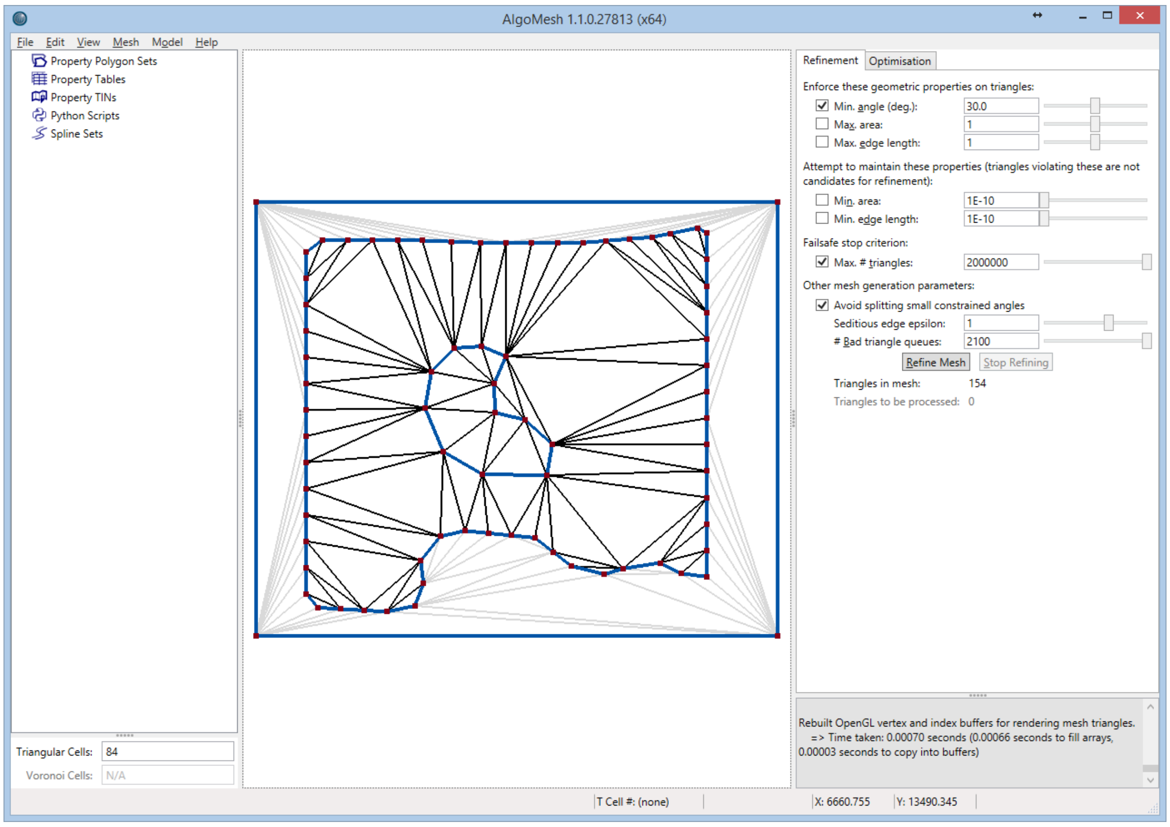
T Cell #: (none) | X: 5426.875 | Y: 13490.345

Figure 68: The initial triangulated mesh domain.

Before generating the mesh, we are going to manually digitise a region in the centre of the domain. To do this, we use a combination of keyboard modifier keys (Ctrl, Shift) while clicking clicking with the left mouse button in the view. Ctrl adds a new point at the clicked position. Shift adjoins the last created point with the point nearest to the clicked position. Ctrl and Shift together create a new point at the clicked position and adjoin it to the last created point.

➢ First go to the **Edit** menu and ensure that **Edit Mesh** mode is enabled. This tells AlgoMesh that any Ctrl- or Shift-clicks in the main view should operate directly on mesh vertices and edges.
➢ **Holding down Ctrl, click somewhere** near the centre of the mesh. A red point should appear where you clicked, and the triangulation is adjusted accordingly.
➢ Next, **hold down both Ctrl and Shift** simultaneously, and **click to create another point** near the first point you created. In addition to creating the point (because Ctrl was held down), holding down Shift caused a blue line segment to be created adjoining the previous point and the new point you just created.
➢ Continue holding **Ctrl+Shift** to **create points in a loop**, as in Figure 69, ending up nearby, but not on top of, the first point you created. If needed, you can use the mouse wheel to zoom in and out and get a better picture of what you're doing. Holding down the left mouse button and dragging (without Ctrl or Shift held down) pans the view.
➢ Now **hold Shift** (without Ctrl) and **click on the first point** you created. This adjoins the two points without creating a new point, closing our new polygon.

If the triangles are obscuring your view of the lines while you are creating them, you can turn off the display of these from the View menu (View -> Triangles) - but once you are done, make sure they are turned on for the next step.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 154
Triangles to be processed: 0

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
  => Time taken: 0.00070 seconds (0.00066 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

Triangular Cells: 84
Voronoi Cells: N/A

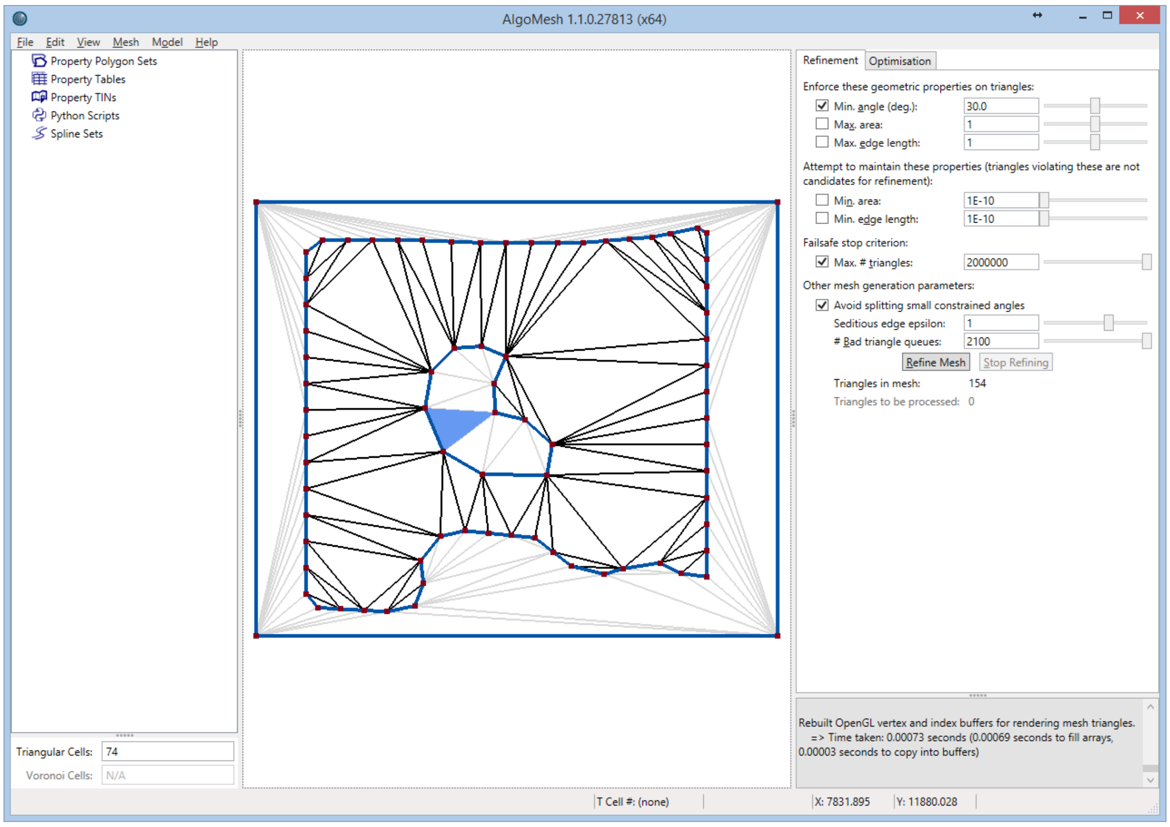
T Cell #: (none) X: 6660.755 Y: 13490.345

Figure 69: The newly created polygonal region near the centre of the mesh.

At times, it is useful to remove some parts of the mesh (e.g. as no-flow boundaries). You can do this in AlgoMesh by selecting a triangle in the region of the mesh you want to remove, and pressing the '-' (minus) key.

- **Click on any triangle inside the polygon** you just created. It should turn blue to indicate that it is selected.
- Press the **'-' (minus)** key. The entire set of triangles inside the polygon should turn a light grey colour as in Figure 70, indicating that this area is now considered "outside" the mesh, and will not form a part of the domain for mesh generation. If you make a mistake and select the wrong region, simply select the triangle and press the '+' (plus) key.
- Press **Esc** to deselect the triangle.

When removing or adding a region from or to the mesh like this, it is important that the region is bounded by a polygon of line segment constraints (the thick blue lines). If there is a gap in the boundary, triangles outside the boundary will also be removed or added.



Enforce these geometric properties on triangles:

☑ Min. angle (deg.): 30.0

☐ Max. area: 1

☐ Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

☐ Min. area: 1E-10

☐ Min. edge length: 1E-10

Failsafe stop criterion:

☑ Max. # triangles: 2000000

Other mesh generation parameters:

☑ Avoid splitting small constrained angles

Seditious edge epsilon: 1

Bad triangle queues: 2100

Triangles in mesh: 154

Triangles to be processed: 0

Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
=> Time taken: 0.00073 seconds (0.00069 seconds to fill arrays, 0.00003 seconds to copy into buffers)

Triangular Cells: 74

Voronoi Cells: N/A

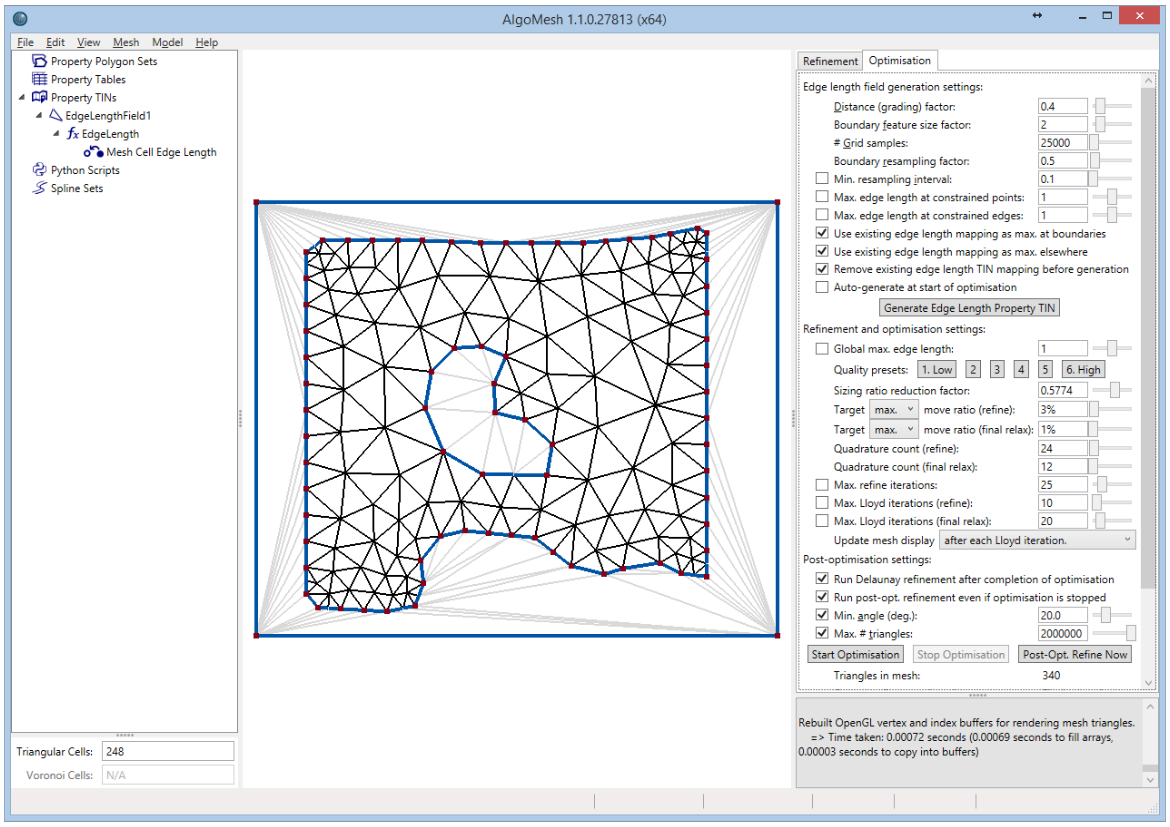
T Cell #: (none) | X: 7831.895 | Y: 11880.028

Figure 70: Pressing the '-' (minus) key with a triangle selected causes the bounded region containing that triangle to be removed from the mesh domain.

Now that we have successfully cut out a polygonal hole, we can generate a mesh from the modified mesh domain.

➢ Select the **Optimisation** tab on the right-hand side of the AlgoMesh window.
➢ Click the **Start Optimisation** button near the bottom of the Optimisation tab, and wait for mesh generation to complete.

Figure 71 shows the resulting mesh. Note that you can use this same process for Voronoi meshes; just turn off Triangles and turn on Voronoi Cells from the View menu.



Triangular Cells: 248
Voronoi Cells: N/A

Edge length field generation settings:

Distance (grading) factor:	0.4
Boundary feature size factor:	2
# Grid samples:	25000
Boundary resampling factor:	0.5
☐ Min. resampling interval:	0.1
☐ Max. edge length at constrained points:	1
☐ Max. edge length at constrained edges:	1

☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☐ Auto-generate at start of optimisation

Generate Edge Length Property TIN

Refinement and optimisation settings:

☐ Global max. edge length:	1
Quality presets:	1. Low, 2, 3, 4, 5, 6. High
Sizing ratio reduction factor:	0.5774
Target max. move ratio (refine):	3%
Target max. move ratio (final relax):	1%
Quadrature count (refine):	24
Quadrature count (final relax):	12
☐ Max. refine iterations:	25
☐ Max. Lloyd iterations (refine):	10
☐ Max. Lloyd iterations (final relax):	20
Update mesh display	after each Lloyd iteration.

Post-optimisation settings:

☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped

☑ Min. angle (deg.):	20.0
☑ Max. # triangles:	2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 340

```
Rebuilt OpenGL vertex and index buffers for rendering mesh triangles.
   => Time taken: 0.00072 seconds (0.00069 seconds to fill arrays,
0.00003 seconds to copy into buffers)
```

Figure 71: The resulting mesh, with the polygonal hole removed from the centre.

More Manual Editing Controls

The Esc key clears both the selected triangle and the selected vertex. To create edge constraints (blue lines) between existing vertices, first press Esc to ensure that any selected vertex is cleared, then hold Shift (without Ctrl), click the first vertex, then Shift-click the vertex you want to join to it with an edge. To remove an edge constraint, use the same process but holding Alt instead of Shift. Note, however, that removal of an edge constraint does not adjust the triangulation. To obtain an appropriately adjusted Delaunay triangulation after removing an edge constraint, go to the Mesh menu and choose Retriangulate.

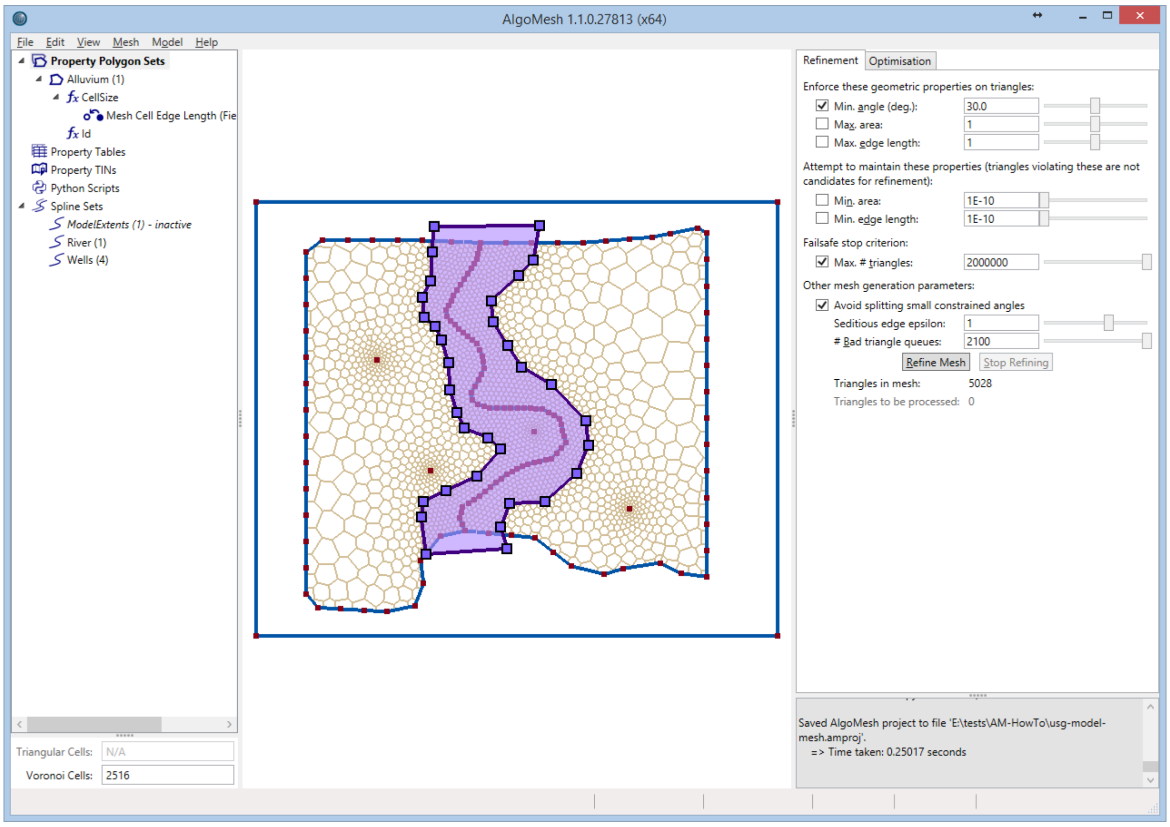
3.9 Set up a Steady-State MODFLOW-USG Model

In this how-to, you will build a complete steady-state MODFLOW-USG model including a river, rainfall recharge, evapotranspiration, pumping wells and constant head boundaries, starting from a generated mesh.

Start by loading up the sample project usg-model-mesh.amproj.

➢ Go to the **File** menu and choose **Open Project**.
➢ Browse to the **Samples** folder of your AlgoMesh installation.
➢ Select **usg-model-mesh.amproj** and click **Open**.

This project file includes a pre-generated Voronoi mesh, as shown in Figure 72. Overlying the mesh is an alluvium polygon, which was used to restrict cell size during mesh generation in a region around the river.



Enforce these geometric properties on triangles:

- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:

- [✓] Max. # triangles: 2000000

Other mesh generation parameters:

- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 5028
Triangles to be processed: 0

Triangular Cells: N/A
Voronoi Cells: 2516

```
Saved AlgoMesh project to file 'E:\tests\AM-HowTo\usg-model-mesh.amproj'.
   => Time taken: 0.25017 seconds
```

Figure 72: The pre-generated mesh for our steady-state model.

To build a steady-state model, we need to do the following:

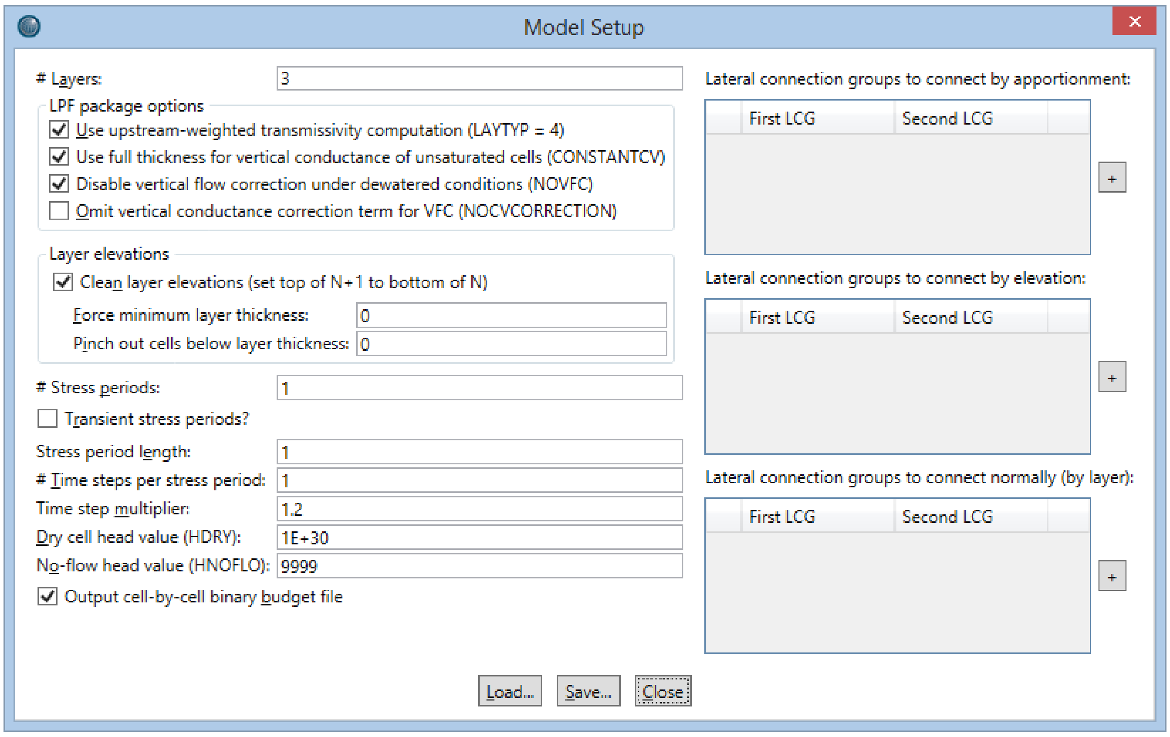
1. Set the number of layers in Model Setup.
2. Map property values to elevations for all layers.
3. Map initial head and hydraulic conductivities Kx and Kz for all layers. Note that anisotropy is not supported in the current version of AlgoMesh, so Ky == Kx.
4. Map boundary conditions to cells that should contain them. In our model, we will be using the constant head (CHD), evapotranspiration (EVT), recharge (RCH), river (RIV) and well (WEL) boundary condition packages.
5. Export the MODFLOW-USG model, make any desired manual modifications, and run it.

Model Setup

First, tell AlgoMesh how many layers we need in our model. For this example we are building a three-layer model.

- ➢ Go to the **Model** menu and choose **Model Setup**.
- ➢ Change **# Layers** to **3**.
- ➢ Leave all other values at their defaults; in particular, for our steady-state model, **# Stress periods** should be **1** and **Transient stress periods** should be turned **off**.
- ➢ Click the **Close** button to close the Model Setup window.

Figure 73 shows the Model Setup window with the number of layers adjusted.


Model Setup

Layers: 3

LPF package options

- [✓] Use upstream-weighted transmissivity computation (LAYTYP = 4)
- [✓] Use full thickness for vertical conductance of unsaturated cells (CONSTANTCV)
- [✓] Disable vertical flow correction under dewatered conditions (NOVFC)
- [ ] Omit vertical conductance correction term for VFC (NOCVCORRECTION)

Layer elevations

- [✓] Clean layer elevations (set top of N+1 to bottom of N)

Force minimum layer thickness:	0
Pinch out cells below layer thickness:	0

# Stress periods:	1
Transient stress periods?	[ ]
Stress period length:	1
# Time steps per stress period:	1
Time step multiplier:	1.2
Dry cell head value (HDRY):	1E+30
No-flow head value (HNOFLO):	9999

- [✓] Output cell-by-cell binary budget file

Lateral connection groups to connect by apportionment:

First LCG	Second LCG

Lateral connection groups to connect by elevation:

First LCG	Second LCG

Lateral connection groups to connect normally (by layer):

First LCG	Second LCG

Figure 73: The model setup window.

Mapping Elevations

Next we need to assign top and bottom elevations to our layers. We will do this by assigning a top elevation to layer 1, and bottom elevations to layers 1-3. AlgoMesh will automatically assign each missing top elevation to the bottom elevation of the layer above.

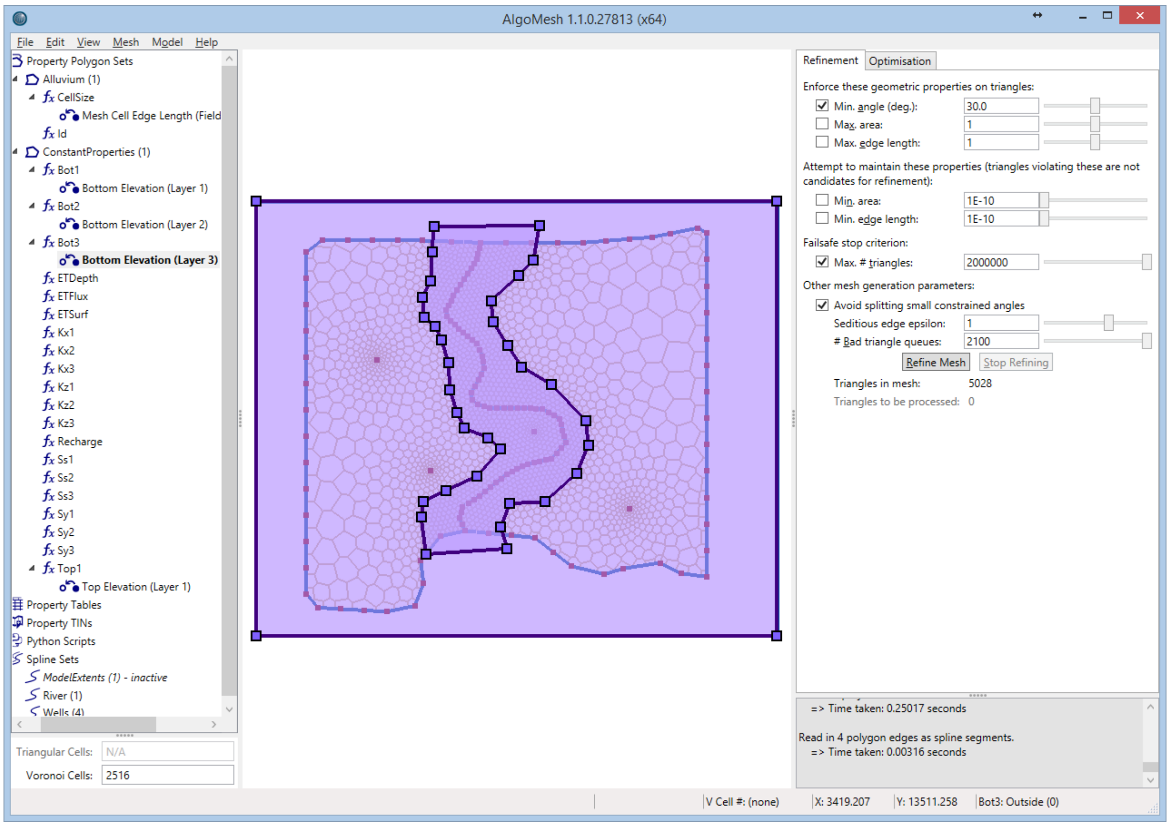
For this model, all of our layer elevations will be constant across the model domain, and we will use a polygon covering the entire model to assign values for these. In a more realistic model, you will likely need to assign elevations that vary over the domain; this can be done using property TINs. See the Remove or Pinch Out Unnecessary MODFLOW-USG Model Cells how-to for an example of this.

The file ConstantProperties.shp is a polygon shapefile containing a rectangle covering the entire model, with attributes specifying the layer elevations in addition to a number of material properties and boundary condition values. We will import this now to allow us to assign our elevations.

- **Right-click** the **Property Polygon Sets** header in the project explorer on the left-hand side of the AlgoMesh window.
- Choose **Import GIS Polygons** from the context menu that appears.
- Select **ConstantProperties.shp** and click **Open**. A new polygon appears over the top of the mesh in the main view, and the ConstantProperties layer is added under Property Polygon Sets, with a number of elevation, boundary condition and material properties listed underneath it.
- Find the **Top1** property in the list, and **right-click** it.
- From the context menu that appears, choose **Map to Mesh Variable as Overlay** and then **Top Elevation**. The mapping Top Elevation (Layer 1) appears beneath the Top1 property, which is what we want.
- **Right-click** the **Bot1** property and choose **Map to Mesh Variable as Overlay** and then **Bottom Elevation**. The mapping Bottom Elevation (Layer 1) appears beneath the Bot1 property, which is what we want.

- **Right-click** the **Bot2** property and choose **Map to Mesh Variable as Overlay** and then **Bottom Elevation**. The mapping Bottom Elevation (Layer 1) appears beneath the Bot2 property, but this property should be mapped to Layer 2 instead.
- **Right-click** the **Bottom Elevation** mapping under **Bot2** and choose **Edit Layers**. The Layer 1 becomes editable.
- Type **2** and hit **Enter** to replace the mapping with a Bottom Elevation (Layer 2) mapping.
- **Right-click** the **Bot3** property and choose **Map to Mesh Variable as Overlay** and then **Bottom Elevation**. The mapping Bottom Elevation (Layer 1) appears beneath the Bot3 property, but this property should be mapped to Layer 3 instead.
- **Right-click** the **Bottom Elevation** mapping under **Bot3** and choose **Edit Layers**.
- Type **3** and hit **Enter** to replace the mapping with a Bottom Elevation (Layer 3) mapping.

Your property mappings should now look like those in Figure 74.


AlgoMesh 1.1.0.27813 (x64)
File Edit View Mesh Model Help
Refinement Optimisation
Refine Mesh Stop Refining

Property Polygon Sets
- Alluvium (1)
  - CellSize
    - Mesh Cell Edge Length (Field
  - Id
- ConstantProperties (1)
  - Bot1
    - Bottom Elevation (Layer 1)
  - Bot2
    - Bottom Elevation (Layer 2)
  - Bot3
    - **Bottom Elevation (Layer 3)**
  - ETDepth
  - ETFlux
  - ETSurf
  - Kx1
  - Kx2
  - Kx3
  - Kz1
  - Kz2
  - Kz3
  - Recharge
  - Ss1
  - Ss2
  - Ss3
  - Sy1
  - Sy2
  - Sy3
  - Top1
    - Top Elevation (Layer 1)

Property Tables
Property TINs
Python Scripts
Spline Sets
- ModelExtents (1) - inactive
- River (1)
- Wells (4)

Triangular Cells: N/A
Voronoi Cells: 2516

Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 5028
Triangles to be processed: 0

```
=> Time taken: 0.25017 seconds

Read in 4 polygon edges as spline segments.
=> Time taken: 0.00316 seconds
```

V Cell #: (none) | X: 3419.207 | Y: 13511.258 | Bot3: Outside (0)

Figure 74: The steady-state model with layer elevations mapped.

If you click on each of the elevation properties and move your mouse cursor over the model domain, you will see that the model consists of a 50m thick top layer from 100m to 50m elevation, a 25m middle layer from 50m to 25m elevation, and a 125m bottom layer from 25m to -100m elevation.

Mapping Initial Head and Hydraulic Conductivities

We also need to provide initial head values for our steady-state simulation. For this model, we will set these for all layers to the model top elevation (100m, mapped from the Top1 property).

- **Right-click** on the **Top1** property and choose **Map to Mesh Variable as Overlay** and then **Initial Head**.

- **Right-click** on the new **Initial Head** mapping underneath the Top1 property and choose **Edit Layers**.
- Type **all** and hit **Enter**. This will replace the default Layer 1 mapping with a mapping applied to all layers in the model. You should see Initial Head (All Layers) as the new label for the mapping. You could also have typed * to represent all layers, or give a specific range like **1-3** or **1,2,3**.

Next, hydraulic conductivities must be assigned across all of the layers. We will again use values that are constant across each layer, with the exception of the first layer which will contain different conductivities in the alluvium. The values to be mapped (in units m/day) are as follows:

	Kx	**Kz**
Layer 1 (Regolith / Alluvium)	5 / 10	0.5 / 1
Layer 2 (Aquitard)	1e-5	1e-6
Layer 3 (Aquifer)	1e-4	1e-5

- Using the mapping technique you used for elevation and initial head, construct the following mappings:
  - **Kx1** => **Kx (Layer 1)**
  - **Kx2** => **Kx (Layer 2)**
  - **Kx3** => **Kx (Layer 3)**
  - **Kz1** => **Kz (Layer 1)**
  - **Kz2** => **Kz (Layer 2)**
  - **Kz3** => **Kz (Layer 3)**

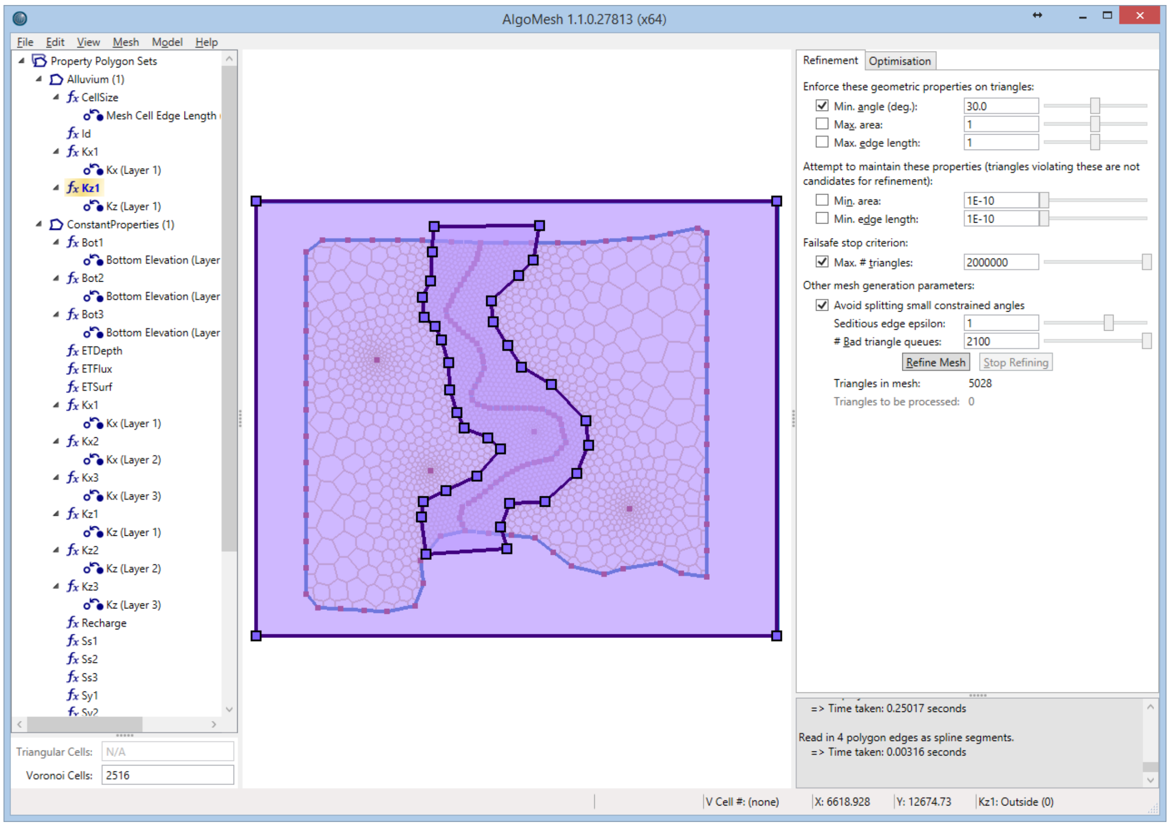
We also need to map the Kx and Kz values for the alluvium. As we do not have properties for these yet, we will add new properties to our Alluvium polygon set.

- **Right-click** the **Alluvium** property polygon set.
- Choose **New Property** from the context menu that appears.
- Give the new property the name **Kx1** and the value **10**.
- Map this new **Kx1** property to **Kx (Layer 1)**.
- **Right-click** the **Alluvium** property polygon set once again.
- Choose **New Property** from the context menu that appears.
- Give the new property the name **Kz1** and the value **1**.
- Map this new **Kz2** property to **Kz (Layer 1)**.

As the Alluvium property polygon set is above the ConstantProperties set in the list, these values will override the regolith values for any cells that lie within the alluvium polygon.

Note that the ConstantProperties shapefile also contains storage properties Ss and Sy. As these are not needed for a steady-state model, we will ignore them for now. They will be used later, in the Set up a Transient MODFLOW-USG Model how-to.

Your property mappings should now look like those in Figure 75.



Triangular Cells: N/A
Voronoi Cells: 2516

Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 5028
Triangles to be processed: 0

```
   => Time taken: 0.25017 seconds

Read in 4 polygon edges as spline segments.
   => Time taken: 0.00316 seconds
```

V Cell #: (none) | X: 6618.928 | Y: 12674.73 | Kz1: Outside (0)

Figure 75: Hydraulic conductivity mappings for the steady-state model.

Mapping Boundary Conditions

Now that we have mapped our elevations, initial head and material properties, we need some boundary conditions to complete the model.

First, we will map recharge and evapotranspiration properties. These will be applied across the entire top of the model domain (layer 1). Recharge will be active in this model at a rate of 0.4 mm/day and evapotranspiration at a rate of 0.5 mm/day to an extinction depth of 2m below ground surface. These properties have been included as fields in the ConstantProperties layer: Recharge, ETDepth (evapotranspiration extinction depth), ETFlux (maximum evapotranspiration rate) and ETSurf (evapotranspiration surface elevation, set to the model top elevation of 100m).

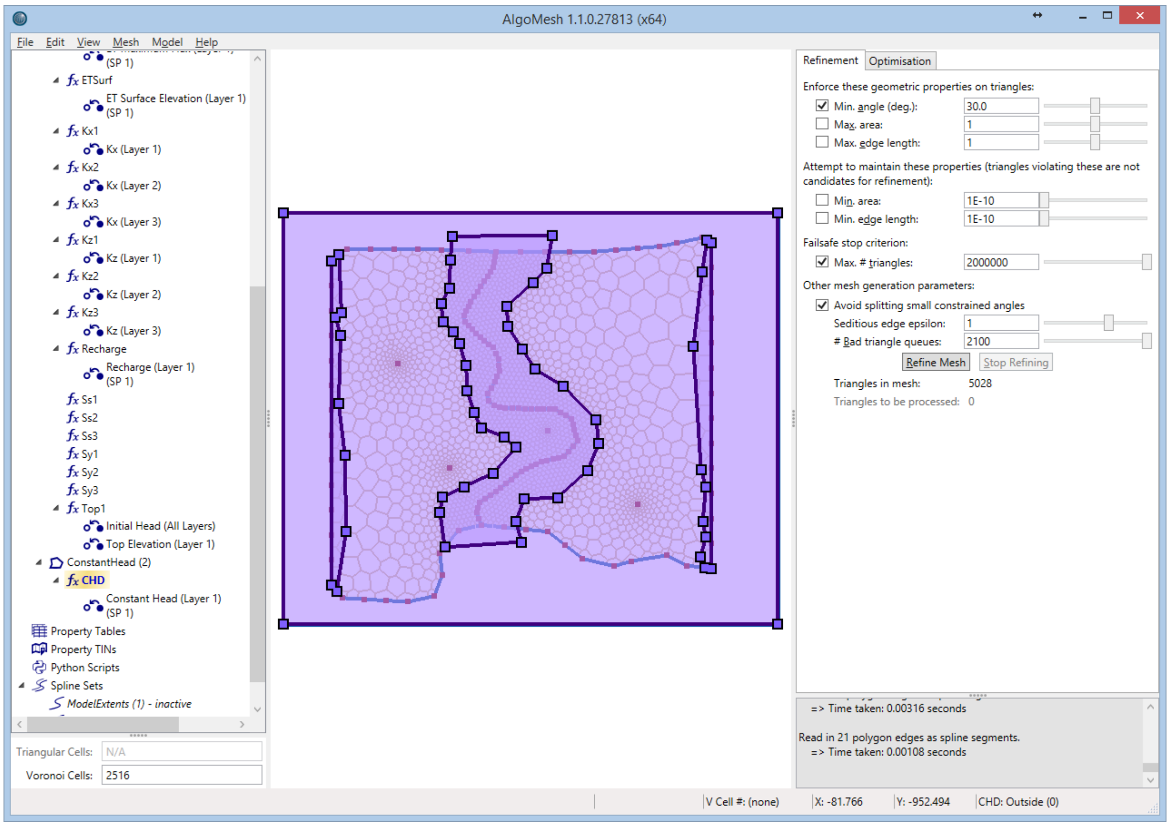
➢ Under the **ConstantProperties** polygon set, construct the following mappings:
  ➢ **ETDepth** => **ET Extinction Depth (Layer 1) (SP 1)**
  ➢ **ETFlux** => **ET Maximum Flux (Layer 1) (SP 1)**
  ➢ **ETSurf** => **ET Surface Elevation (Layer 1) (SP 1)**
  ➢ **Recharge** => **Recharge (Layer 1) (SP 1)**

Notice that these mappings all have a stress period (SP) listed after them. These can all be left at SP 1 for the time being, as we will only have a single steady-state stress period in our model. Later, when we want to build a transient model, these stress periods will need to be changed.

We will now apply constant-head boundaries along the left and right sides of the model. On the left, we will have a constant head of 90m, and on the right we will have a constant head of 100m. These will be added by bringing in a polygon on each side which covers the cells at each respective edge of the model.

- **Right-click** on the **Property Polygon Sets** header.
- Choose **Import GIS Polygons** from the context menu that appears.
- Select **ConstantHead.shp** and click **Open**. Two new polygons appear in the main view - one on either edge of the model - and the ConstantHead set appears below the ConstantProperties set, with a single CHD property underneath it.
- Construct a mapping from the **CHD** property to **Constant Head (Layer 1).**

You should now have a screen that looks like Figure 76. Note how the constant head polygons overlap most of the area of the outermost cells on the left and right model edges. They also slightly overlap some cells further in, but only cells whose cell centres lie inside the polygons will have the constant head boundary condition applied. To see precisely where the cell centres lie, turn on Triangles from the View menu. In general, a Voronoi cell has its centre at the triangle vertex that lies inside it. Alternatively, export the mesh to a GIS shapefile and find the X, Y coordinates of the cell centres in the CentreX and CentreY fields of the layer data table.



Triangular Cells: N/A
Voronoi Cells: 2516

Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles
- Seditious edge epsilon: 1
- # Bad triangle queues: 2100

Triangles in mesh: 5028
Triangles to be processed: 0

```
   => Time taken: 0.00316 seconds

Read in 21 polygon edges as spline segments.
   => Time taken: 0.00108 seconds
```

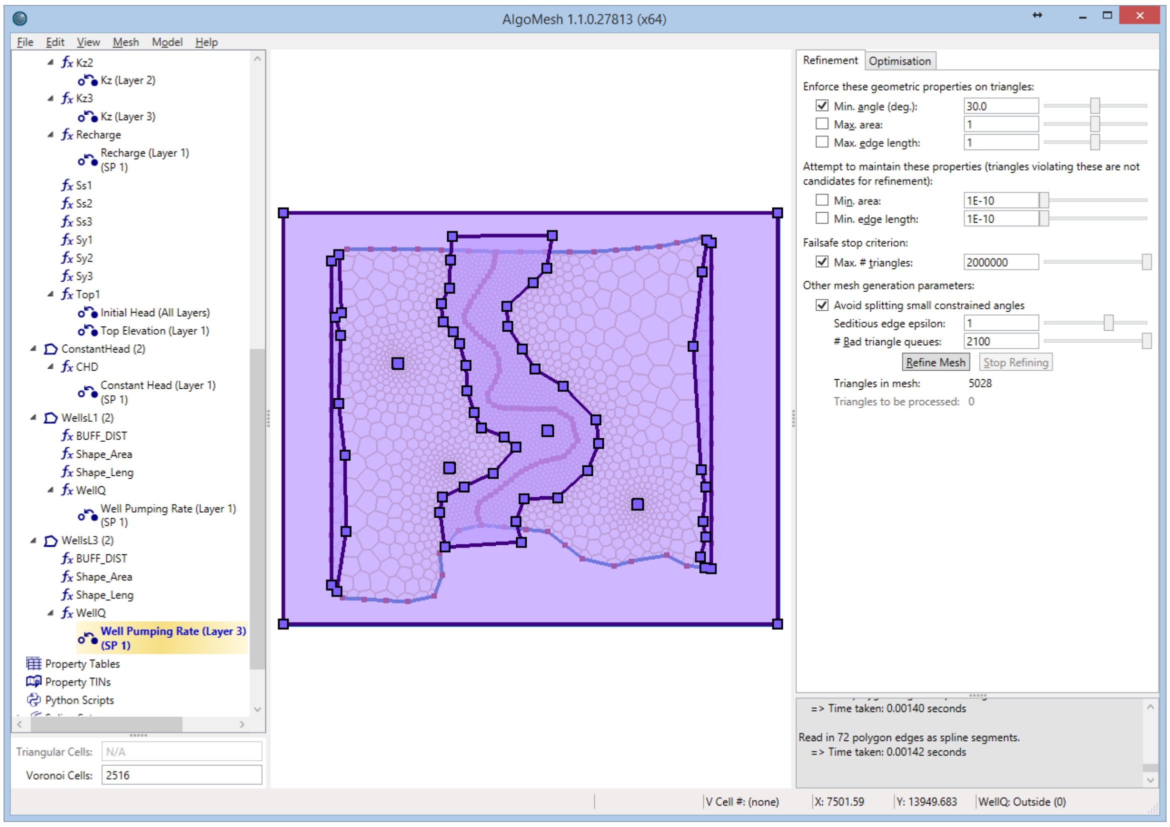
V Cell #: (none) | X: -81.766 | Y: -952.494 | CHD: Outside (0)

Figure 76: Polygons are added along the western and eastern edges of the model to facilitate constant-head mappings of 90m on the west and 100m on the east.

Next up are the pumping wells. We have four of these in the model - two pumping from layer 1 and two pumping from layer 3. To map the boundary conditions for these we have produced a polygon shapefile for each of the layers 1 and 3. Each polygon in these shapefiles is a small, roughly circular polygon buffered at a radius of 20m around its respective well location (the lone constrained points you can see in the mesh). Each polygon has a WellQ field containing the pumping rate of its well in m^3 / day. From the furthest western to the furthest eastern well, these pumping rates are -1000 (Layer 1), -1250 (Layer 3), -800 (Layer 3) and -500 (Layer 1).

- ➢ **Right-click** on the **Property Polygon Sets** header.
- ➢ Choose **Import GIS Polygons** from the context menu that appears.
- ➢ Select **WellsL1.shp** and click **Open**. The WellsL1 set is added to the list of polygon sets, and small polygons surrounding two of the well locations appear in the main view.
- ➢ Find the **WellQ** property under **WellsL1** and map it **Well Pumping Rate (Layer 1) (SP 1)**.
- ➢ **Right-click** on the **Property Polygon Sets** header.
- ➢ Choose **Import GIS Polygons** from the context menu that appears.
- ➢ Select **WellsL3.shp** and click **Open**. The WellsL3 set is added to the list of polygon sets, and small polygons surrounding the remaining two well locations appear in the main view.
- ➢ Find the **WellQ** property under **WellsL3** and map it **Well Pumping Rate (Layer 3) (SP 1)**. Note the change to **Layer 3** here.

The resulting mappings can be seen in Figure 77.



Triangular Cells: N/A
Voronoi Cells: 2516

Enforce these geometric properties on triangles:
- [✓] Min. angle (deg.): 30.0
- [ ] Max. area: 1
- [ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
- [ ] Min. area: 1E-10
- [ ] Min. edge length: 1E-10

Failsafe stop criterion:
- [✓] Max. # triangles: 2000000

Other mesh generation parameters:
- [✓] Avoid splitting small constrained angles

Seditious edge epsilon: 1
Bad triangle queues: 2100

Triangles in mesh: 5028
Triangles to be processed: 0

```
   => Time taken: 0.00140 seconds

Read in 72 polygon edges as spline segments.
   => Time taken: 0.00142 seconds
```

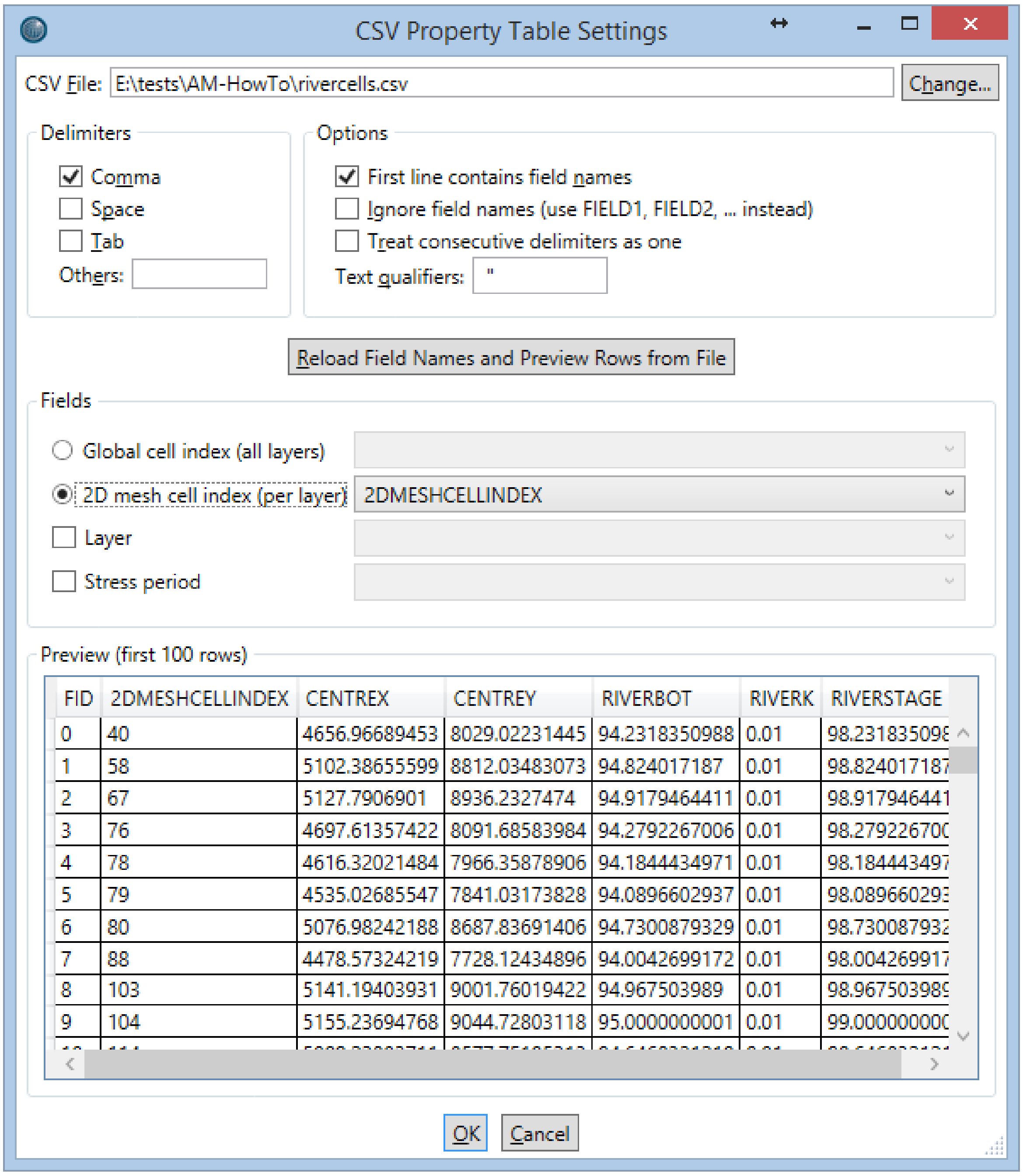
V Cell #: (none) | X: 7501.59 | Y: 13949.683 | WellQ: Outside (0)

Figure 77: Well polygons are added and pumping rates are mapped in their respective layers.

The final remaining boundary condition to map for this model is the river. This is a bit trickier than the others, as the river bed elevation for this model varies spatially, and we would like the river only to be active across a single cell's width. To deal with this, we are going to use a direct cell-by-cell mapping imported from a comma-separate values (CSV) file. The CSV file we are going to use specifies a list of mesh cell indices making up the river, and for each cell, a river bottom elevation, stage and conductivity (conductance per unit area) is given. The river bed elevation varies from 95m at the top of the model to 90m at the bottom, a uniform conductivity of 0.01 m/day is used and stage is constant at 4m above the river bed.

In this case, the CSV file for the river is provided, and you can simply use this to map the river boundary conditions as detailed below. In building your own models, making such a CSV file would typically be done externally in GIS software. First you would export the mesh from AlgoMesh to a shapefile by going to File -> Voronoi Mesh -> Save Cell Geometry. Bringing this shapefile into your GIS package, you would select the cells along the river line and export these to a new shapefile. Within that shapefile, you would add new fields for bed elevation, conductivity and stage, and calculate appropriate values for these from corresponding surface elevations or from an external data source. Once this is done, the data table for this shapefile can be exported as a CSV file and brought into AlgoMesh for boundary condition mapping.

- Find the **Property Tables** header in the project explorer on the left-hand side of the AlgoMesh window, and **right-click** on it.
- Select the file **rivercells.csv** and click **Open**. The CSV Property Table Settings window appears with a collection of settings for reading the data table, and a preview of the data it contains.
- In the **Fields** box, select **2D mesh cell index (per layer)** and choose the field **2DMESHCELLINDEX** from the drop-down menu next to it. This tells AlgoMesh that in each line of the CSV file, the 2DMESHCELLINDEX column contains the index of a cell to have boundary conditions or properties mapped to it. Other columns may contain data values for the mappings themselves.
- Your CSV Property Table Settings window should now look like the one in Figure 78. Your CSV file path may vary from the one shown. **Click OK** to accept the table settings.



CSV File: E:\tests\AM-HowTo\rivercells.csv

Change...

Delimiters

- [✓] Comma
- [ ] Space
- [ ] Tab
- Others:

Options

- [✓] First line contains field names
- [ ] Ignore field names (use FIELD1, FIELD2, ... instead)
- [ ] Treat consecutive delimiters as one
- Text qualifiers: "

Reload Field Names and Preview Rows from File

Fields

- ○ Global cell index (all layers)
- ● 2D mesh cell index (per layer): 2DMESHCELLINDEX
- [ ] Layer
- [ ] Stress period

Preview (first 100 rows)

FID	2DMESHCELLINDEX	CENTREX	CENTREY	RIVERBOT	RIVERK	RIVERSTAGE
0	40	4656.96689453	8029.02231445	94.2318350988	0.01	98.23183509
1	58	5102.38655599	8812.03483073	94.824017187	0.01	98.82401718
2	67	5127.7906901	8936.2327474	94.9179464411	0.01	98.9179464411
3	76	4697.61357422	8091.68583984	94.2792267006	0.01	98.27922670
4	78	4616.32021484	7966.35878906	94.1844434971	0.01	98.18444349
5	79	4535.02685547	7841.03173828	94.0896602937	0.01	98.08966029
6	80	5076.98242188	8687.83691406	94.7300879329	0.01	98.73008793
7	88	4478.57324219	7728.12434896	94.0042699172	0.01	98.00426991
8	103	5141.19403931	9001.76019422	94.967503989	0.01	98.96750398
9	104	5155.23694768	9044.72803118	95.0000000001	0.01	99.00000000

OK Cancel

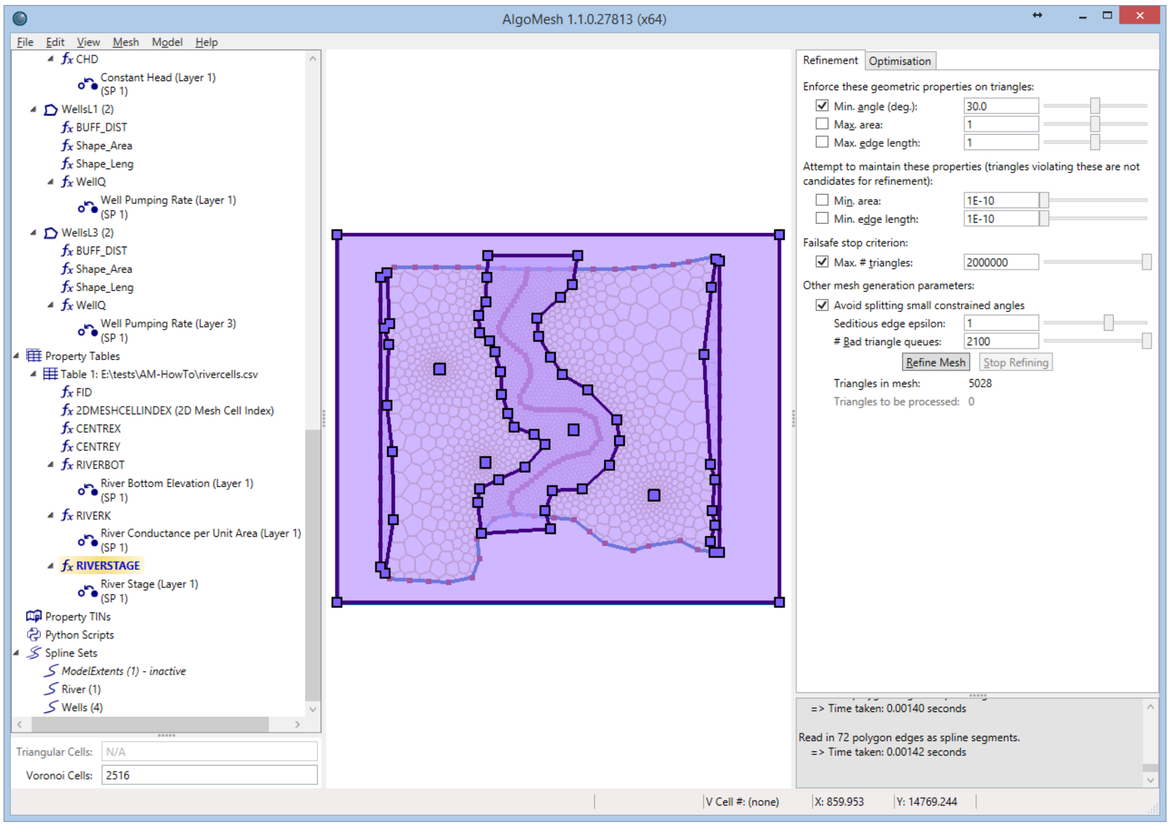
Figure 78: The CSV property table settings for the river cells file.

Note: the 2D mesh cell index refers to the indexing AlgoMesh applies to the cells in the two-dimensional mesh. Each cell in the 2D mesh has a unique index, which can be found by moving the mouse over the cell in the main view and reading the "V Cell #" that appears in the status bar - or by inspecting the data table of the GIS shapefile that is exported by going to File -> Voronoi Mesh -> Save Cell Geometry. The 2D mesh cell index for each cell is the same regardless of the layer, and hence it must be combined with a layer number for any mappings that use it. There is a secondary indexing scheme called the *global cell index*, in which the same mesh cells in different layers have different indices. The global cell index is the index that is used by MODFLOW-USG to uniquely identify any cell over the entire 3D model. When it writes MODFLOW-USG model files, AlgoMesh also writes layer-to-global-index.csv and global-to-layer-index.csv files which detail how the 2D and 3D mappings correlate to each other.

Once the CSV table settings have been accepted, a new table Table 1 appears under the Property Tables header, and beneath it is a list containing all of the columns in the CSV file. You can now use any of these columns for mappings, much like we did previously for GIS shapefile fields.

- Under **Table 1**, **right-click RIVERBOT** and map it to the property **River Bottom Elevation (Layer 1) (SP 1)**.
- Similarly, map **RIVERK** to **River Conductance per Unit Area (Layer 1) (SP 1)**. The conductance per unit area property will cause AlgoMesh to calculate conductance values on a cell-by-cell basis as the product of the given value and the area of the cell to which the boundary condition is being applied. You could instead use the River Conductance property if you had your own pre-computed conductance values for each cell; this is useful if you want to represent a fixed-width river bed, for example.
- Finally, map **RIVERSTAGE** to **River Stage (Layer 1) (SP 1)**.

After doing this, you should have a set of river mappings that look like those in Figure 79.



CHD
Constant Head (Layer 1) (SP 1)
WellsL1 (2)
BUFF_DIST
Shape_Area
Shape_Leng
WellQ
Well Pumping Rate (Layer 1) (SP 1)
WellsL3 (2)
BUFF_DIST
Shape_Area
Shape_Leng
WellQ
Well Pumping Rate (Layer 3) (SP 1)
Property Tables
Table 1: E:\tests\AM-HowTo\rivercells.csv
FID
2DMESHCELLINDEX (2D Mesh Cell Index)
CENTREX
CENTREY
RIVERBOT
River Bottom Elevation (Layer 1) (SP 1)
RIVERK
River Conductance per Unit Area (Layer 1) (SP 1)
RIVERSTAGE
River Stage (Layer 1) (SP 1)
Property TINs
Python Scripts
Spline Sets
ModelExtents (1) - inactive
River (1)
Wells (4)

Triangular Cells: N/A
Voronoi Cells: 2516

Enforce these geometric properties on triangles:
[✓] Min. angle (deg.): 30.0
[ ] Max. area: 1
[ ] Max. edge length: 1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):
[ ] Min. area: 1E-10
[ ] Min. edge length: 1E-10

Failsafe stop criterion:
[✓] Max. # triangles: 2000000

Other mesh generation parameters:
[✓] Avoid splitting small constrained angles
Seditious edge epsilon: 1
Bad triangle queues: 2100

Triangles in mesh: 5028
Triangles to be processed: 0

```
=> Time taken: 0.00140 seconds

Read in 72 polygon edges as spline segments.
=> Time taken: 0.00142 seconds
```

V Cell #: (none) X: 859.953 Y: 14769.244

Figure 79: River boundary condition properties are mapped from the rivercells CSV file.

Exporting and Running the MODFLOW-USG Model

We now have a complete set of material properties and boundary conditions mapped for our model, and we are ready to build MODFLOW-USG input files for it.

- Go to the **File** menu, then **Voronoi Mesh** and **Export to MODFLOW-USG**.

- ➢ **Browse to a location** to which you have write access, such as your Documents folder.
- ➢ *(Recommended)* **Create a new working folder** for the model files to be saved to. This will keep the numerous model files that are produced separate from other files on your computer.
- ➢ **Type a model name prefix**, e.g. **usgss** and click **Save**. Note that you should not include a file name extension here, as this name is used to create a number of model files, each with a different extension.

AlgoMesh will write all of the files necessary to run the USG model. When it is done, you should see a message in the grey log pane in the bottom-right corner of the AlgoMesh window saying "Wrote MODFLOW-USG input files for Voronoi mesh."

Take a moment to open the working folder to which you saved the model files in Windows Explorer, and inspect the files that are there. You should have the following:

- usgss.**BAS** (Basic package containing IBOUND and initial head arrays)
- usgss.**DISU** (Unstructured discretisation package containing all of the spatial and layer discretisation information for your model as well as stress period details)
- usgss.**EVT** (Evapotranspiration package)
- usgss.**GNC** (Ghost node correction package - note that this is typically not needed for Voronoi grids, and the included NAM file will not incorporate it by default)
- usgss.**GSF** (Grid specification file containing additional geometric information about your grid that is not included in the DISU file - this is useful to allow utilisation of other third-party MODFLOW-USG utilities with your model)
- usgss.**LPF** (Layer property flow package containing flow options, layer types and hydraulic conductivities)
- usgss.**NAM** (Name file listing all of the packages to be included in the model run)
- usgss.**OC** (Output control file telling MODFLOW-USG to save head and budget information at each time step)
- usgss.**RCH** (Recharge package)
- usgss.**RIV** (River package)
- usgss.**SMS** (Sparse matrix solver package containing a set of default solver options that perform usually well on typical complex models)
- usgss.**VTK** (Visualisation toolkit file allowing 3D display of the model geometry using the free third-party software ParaView®)
- usgss.**WEL** (Well package)
- usgss**_LCG_Connections.CSV** (An auxiliary visualisation data file allowing 3D display of detailed cell-to-cell connection information)
- usgss**-global-to-layer-index.CSV** (Global cell index to 2D mesh cell index + layer lookup file)
- usgss**-layer-to-global-index.CSV** (2D mesh cell index + layer to global cell index lookup file)

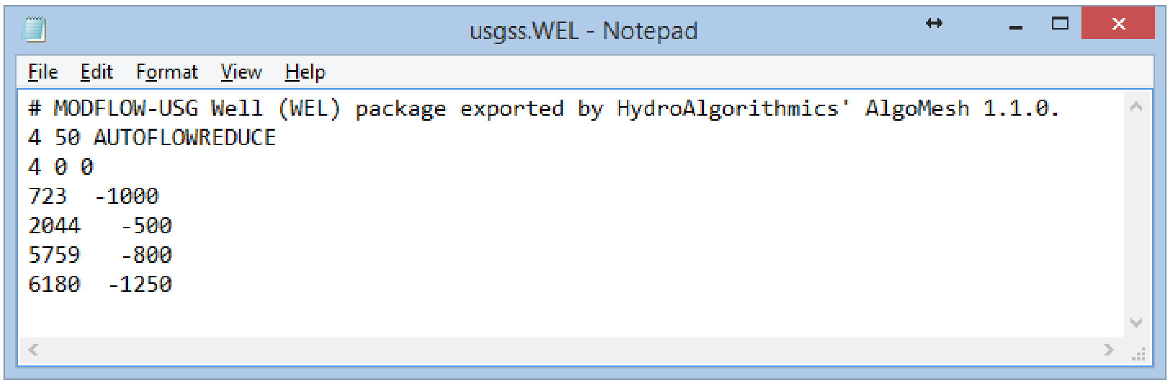
More information can be found on most of these file formats in the MODFLOW-USG documentation. If any of these files are missing, it is likely that you missed one of the property mappings during the tutorial - adding any missing mappings and re-exporting should result in the file appearing.

Before running the model, there is one manual change we need to make: to apply the AUTOFLOWREDUCE keyword to the header of the WEL package input file. This keyword tells MODFLOW-USG that wells should stop pumping as soon as the water level reaches the bottom of the cell they are pumping from. In our model we have quite high pumping rates from the aquifer in Layer 3, and applying this keyword will stop the model from producing unrealistic heads below the bottom of the model from pumping when the cells are dry.

- ➢ Open the file **usgss.WEL** in your favourite text editor (you can use **Notepad**, which comes with Windows).

- At the end of the second line of the file, type **a space** and then the keyword **AUTOFLOWREDUCE**.
- **Save** the file and **Close** the text editor.

Your WEL package file should now look like the one in Figure 80.



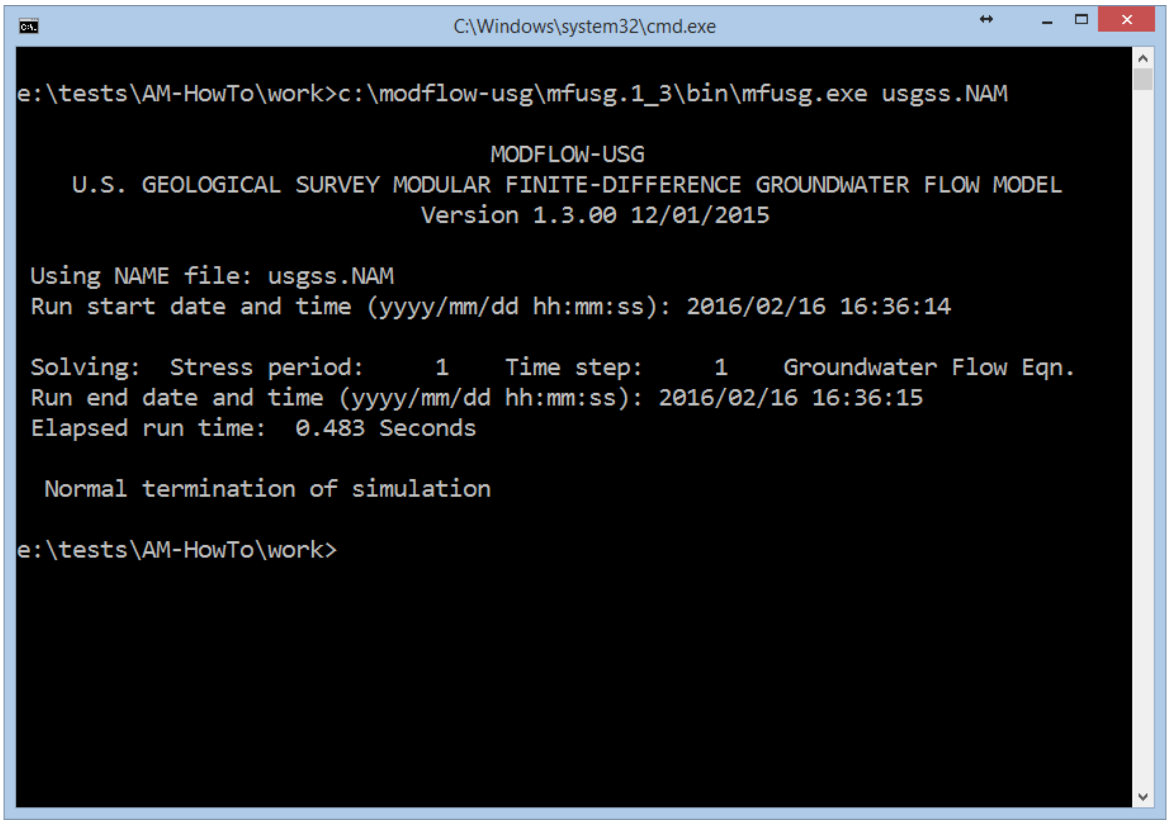
```
# MODFLOW-USG Well (WEL) package exported by HydroAlgorithmics' AlgoMesh 1.1.0.
4 50 AUTOFLOWREDUCE
4 0 0
723  -1000
2044   -500
5759   -800
6180  -1250
```

Figure 80: The WEL package input file, modified to include the AUTOFLOWREDUCE keyword at the end of the header line.

To run the model, you need to have the MODFLOW-USG executables on your computer. You can download MODFLOW-USG free of charge from the USGS Groundwater website. At the time of writing, this may be found at the following link: http://water.usgs.gov/ogw/mfusg/

It is assumed here that you have worked with MODFLOW models before, and have some prior knowledge on running MODFLOW and working with model inputs and outputs. If not, it is recommended that you refer to the literature available from the USGS website to familiarise yourself with the ins and outs of MODFLOW and MODFLOW-USG modelling.

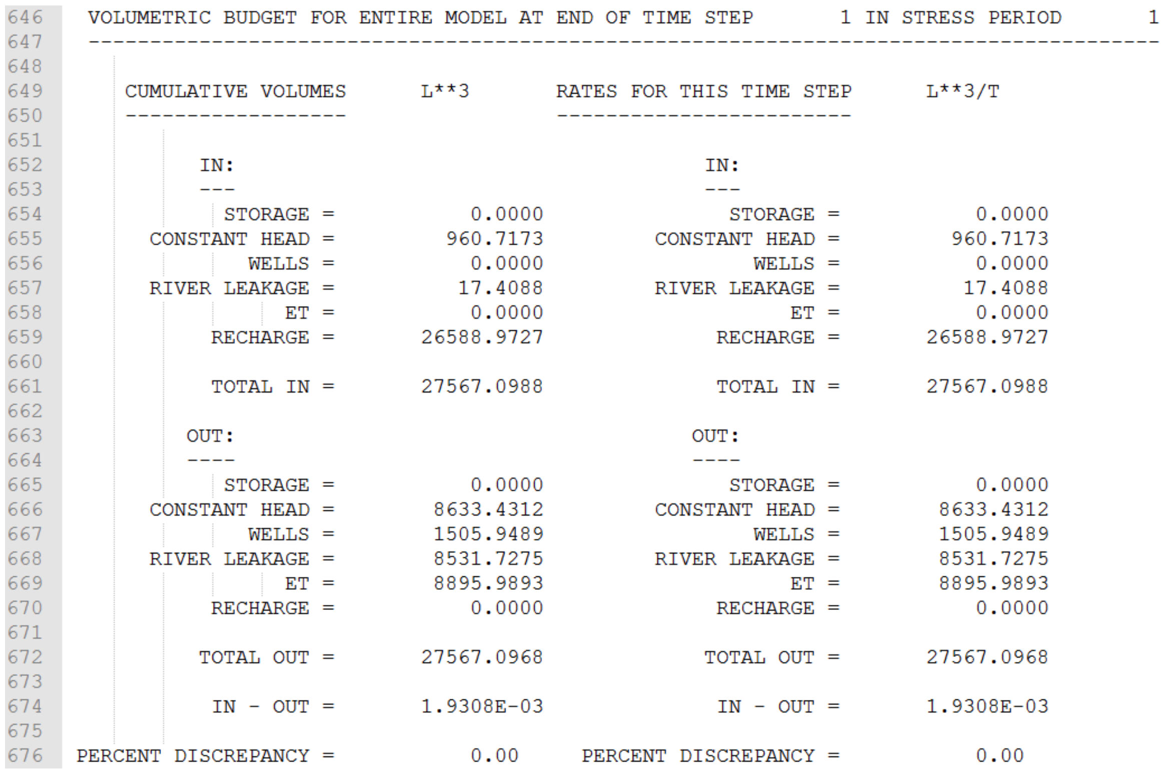
- Start a command prompt (press the Windows key or click the **Start** button, then type **Command Prompt** to search for it, and select **Command Prompt** from the list).
- Change to the path of your working folder, e.g. type **cd /d "e:\tests\AM-HowTo\work"** (replace *e:\tests\AM-HowTo\work* with the full path to your model files).
- Execute MODFLOW-USG, passing the usgss.NAM file as a parameter on the command-line. For example, if you downloaded MODFLOW-USG and unzipped its contents into the folder **c:\modflow-usg\mfusg.1_3**, you would type **c:\modflow-usg\mfusg.1_3\bin\mfusg.exe usgss.NAM** (for the 32-bit version) or **c:\modflow-usg\mfusg.1_3\bin\mfusg_x64.exe usgss.NAM** (for the 64-bit version).
- Wait for the simulation to complete. As this is a small, simple model, it should not take long. You should see a run-time report similar to that shown in Figure 81.



```
e:\tests\AM-HowTo\work>c:\modflow-usg\mfusg.1_3\bin\mfusg.exe usgss.NAM

                                 MODFLOW-USG
    U.S. GEOLOGICAL SURVEY MODULAR FINITE-DIFFERENCE GROUNDWATER FLOW MODEL
                            Version 1.3.00 12/01/2015

 Using NAME file: usgss.NAM
 Run start date and time (yyyy/mm/dd hh:mm:ss): 2016/02/16 16:36:14

 Solving:  Stress period:     1    Time step:     1    Groundwater Flow Eqn.
 Run end date and time (yyyy/mm/dd hh:mm:ss): 2016/02/16 16:36:15
 Elapsed run time:  0.483 Seconds

  Normal termination of simulation

e:\tests\AM-HowTo\work>
```

Figure 81: The result of running MODFLOW-USG on our model files from the command-line.

Once the simulation is complete, you should check the listing file that was created (**usgss.LST**), scroll to the end and ensure that the mass balance results are acceptable. For this tutorial model, you should see only a very small discrepancy in budget results, as in Figure 82.



```
 VOLUMETRIC BUDGET FOR ENTIRE MODEL AT END OF TIME STEP       1 IN STRESS PERIOD       1
 ------------------------------------------------------------------------------------

    CUMULATIVE VOLUMES      L**3       RATES FOR THIS TIME STEP      L**3/T
    ------------------                 ------------------------

          IN:                                      IN:
          ---                                      ---
             STORAGE =          0.0000                STORAGE =          0.0000
       CONSTANT HEAD =        960.7173          CONSTANT HEAD =        960.7173
               WELLS =          0.0000                  WELLS =          0.0000
       RIVER LEAKAGE =         17.4088          RIVER LEAKAGE =         17.4088
                  ET =          0.0000                     ET =          0.0000
            RECHARGE =      26588.9727               RECHARGE =      26588.9727

            TOTAL IN =      27567.0988               TOTAL IN =      27567.0988

         OUT:                                     OUT:
         ----                                     ----
             STORAGE =          0.0000                STORAGE =          0.0000
       CONSTANT HEAD =       8633.4312          CONSTANT HEAD =       8633.4312
               WELLS =       1505.9489                  WELLS =       1505.9489
       RIVER LEAKAGE =       8531.7275          RIVER LEAKAGE =       8531.7275
                  ET =       8895.9893                     ET =       8895.9893
            RECHARGE =          0.0000               RECHARGE =          0.0000

           TOTAL OUT =      27567.0968              TOTAL OUT =      27567.0968

            IN - OUT =      1.9308E-03               IN - OUT =      1.9308E-03

PERCENT DISCREPANCY =          0.00     PERCENT DISCREPANCY =          0.00
```

Figure 82: The volumetric budget results from the steady-state simulation, obtained from the end of the listing file.

Finally, you may wish to inspect the head output from the simulation, stored in the file **usgss.HDS**. The Load MODFLOW-USG Results how-to explains in detail how to do this.

Finishing Up

You have now set up and run a complete steady-state MODFLOW-USG model. The Set up a Transient MODFLOW-USG Model section follows on from this tutorial to explain how to adapt this to a transient simulation model. Before continuing, it is a good idea to save your work.

➢ Go to the **File** menu and choose **Save Project**.
➢ **Browse to a location** to which you have write access, such as your Documents folder.
➢ **Type a name** for the project file to be saved, e.g. **USGSteadyState.amproj** and click **Save**.

3.10 Remove or Pinch Out Unnecessary MODFLOW-USG Model Cells

One of the benefits of MODFLOW-USG over traditional structured grid versions of MODFLOW is the ability to eliminate cells where geology pinches out or outcrops. In this how-to, we look at how this can be done in AlgoMesh.

Before starting this tutorial, you must have completed the Set up a Steady-State MODFLOW-USG Model how-to, as we will start from the AlgoMesh project for the steady-state model that was built in that section.

➢ Go to the **File** menu and choose **Open Project**.
➢ Browse to the folder in which you saved your model from the steady-state tutorial, select the

.amproj file you saved (e.g. **USGSteadyState.amproj**), and click **Open**.

Mapping Spatially-Varying Elevation Data

For this how-to, we are going to alter the second layer of our steady-state model to vary its thickness spatially across the domain. We will do this by mapping a spatially-varying bottom elevation ranging from 25m (the original full thickness) to 50m (zero thickness, i.e. pinched out). A map of the new layer 2 bottom elevation that we will be using is shown in Figure 83.

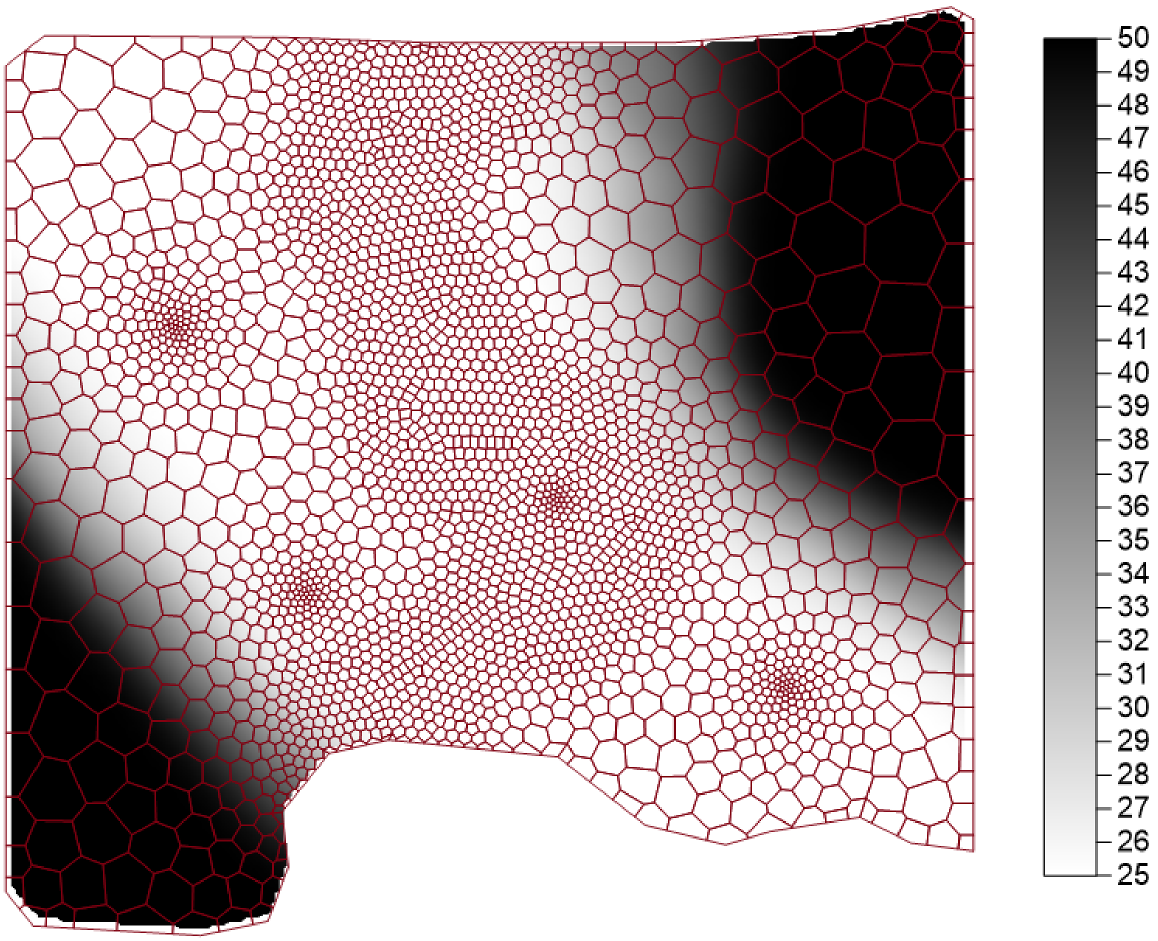


Figure 83: The new spatially-varying layer 2 bottom elevation, ranging from 25m to 50m.

The raster grid of elevations for this map has been provided in a comma-delimited XYZ format in the file layer2-bottom.xyz. In a more realistic model, you would typically use such a grid for every layer elevation in the model. In this model, we are only doing it for a single layer's bottom elevation, to illustrate the process. Feel free to open this file in a text editor to inspect its contents.

First, we will remove the old constant-value mapping for Bottom Elevation (Layer 2) from the model.

- In the project explorer on the left-hand side of the AlgoMesh window, find the mapping entry **Bottom Elevation (Layer 2)** under the **Bot2** property of the **ConstantProperties** polygon set.
- **Right-click** the **Bottom Elevation (Layer 2)** entry.
- Choose **Remove Mapping** from the context menu that appears. The mapping disappears and Layer 2's bottom elevation is now unmapped.

Ensure you leave the Top Elevation (Layer 1), Bottom Elevation (Layer 1) and Bottom Elevation (Layer 3) mappings intact, as we will not be changing these. Your ConstantProperties set property mappings should now look like those in Figure 84.

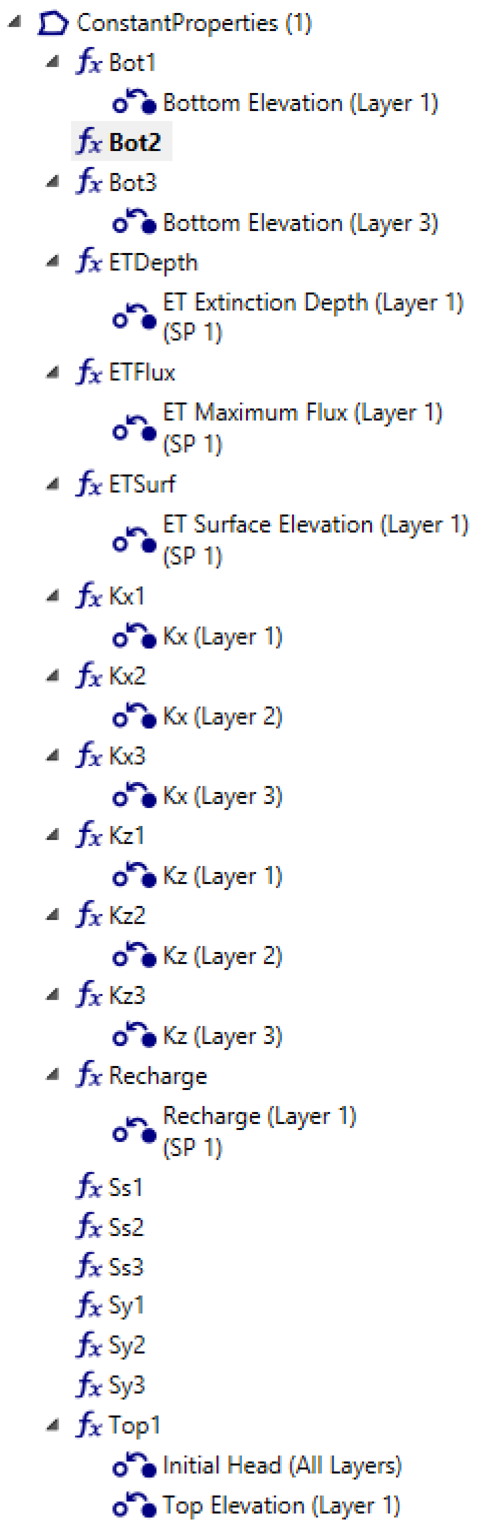

ConstantProperties (1)
- Bot1
  - Bottom Elevation (Layer 1)
- Bot2
- Bot3
  - Bottom Elevation (Layer 3)
- ETDepth
  - ET Extinction Depth (Layer 1) (SP 1)
- ETFlux
  - ET Maximum Flux (Layer 1) (SP 1)
- ETSurf
  - ET Surface Elevation (Layer 1) (SP 1)
- Kx1
  - Kx (Layer 1)
- Kx2
  - Kx (Layer 2)
- Kx3
  - Kx (Layer 3)
- Kz1
  - Kz (Layer 1)
- Kz2
  - Kz (Layer 2)
- Kz3
  - Kz (Layer 3)
- Recharge
  - Recharge (Layer 1) (SP 1)
- Ss1
- Ss2
- Ss3
- Sy1
- Sy2
- Sy3
- Top1
  - Initial Head (All Layers)
  - Top Elevation (Layer 1)

Figure 84: ConstantProperties mappings with Bot2's mapping removed.

We are now ready to bring in the new spatially-varying layer 2 bottom elevation. To do this, we will create a *property TIN* from the XYZ file containing the elevations. A property TIN is a triangulated surface that AlgoMesh produces to interpolate among a given set of sample points or grid points.

Once constructed, AlgoMesh can retrieve an interpolated value at any point within the TIN's extents.

➢ Scroll down in the project explorer pane until you see the **Property TINs** header.
➢ **Right-click** the **Property TINs** header.
➢ Choose **New Property TIN** from the context menu that appears.
➢ Browse to the **Samples** folder of your AlgoMesh installation.
➢ Select **layer2-bottom.xyz** and click **Open**. AlgoMesh creates a new layer2-bottom TIN for the elevation dataset and an entry for this appears below the Property TINs header, with a Z property beneath it.
➢ To investigate the **Z** property, **left-click** it to select it, and **move the mouse cursor** slowly over the mesh domain. The interpolated Z value shows in the bottom-right corner of the AlgoMesh window; note how the Z value varies gradually from 50 in the south-west and north-east corners of the model to 25 in the centre.
➢ **Right-click** the **Z** property, and choose **Map to Mesh Variable** and then **Bottom Elevation**. A new Bottom Elevation (Layer 1) entry appears beneath the Z property.
➢ **Right-click** the new **Bottom Elevation (Layer 1)** entry and choose **Edit Layers**.
➢ Type **2** and hit **Enter** to change the mapping to **Bottom Elevation (Layer 2)**.

You should now have the new elevation data mapped to the bottom elevation for layer 2, as in Figure 85.

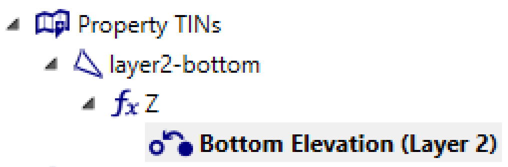
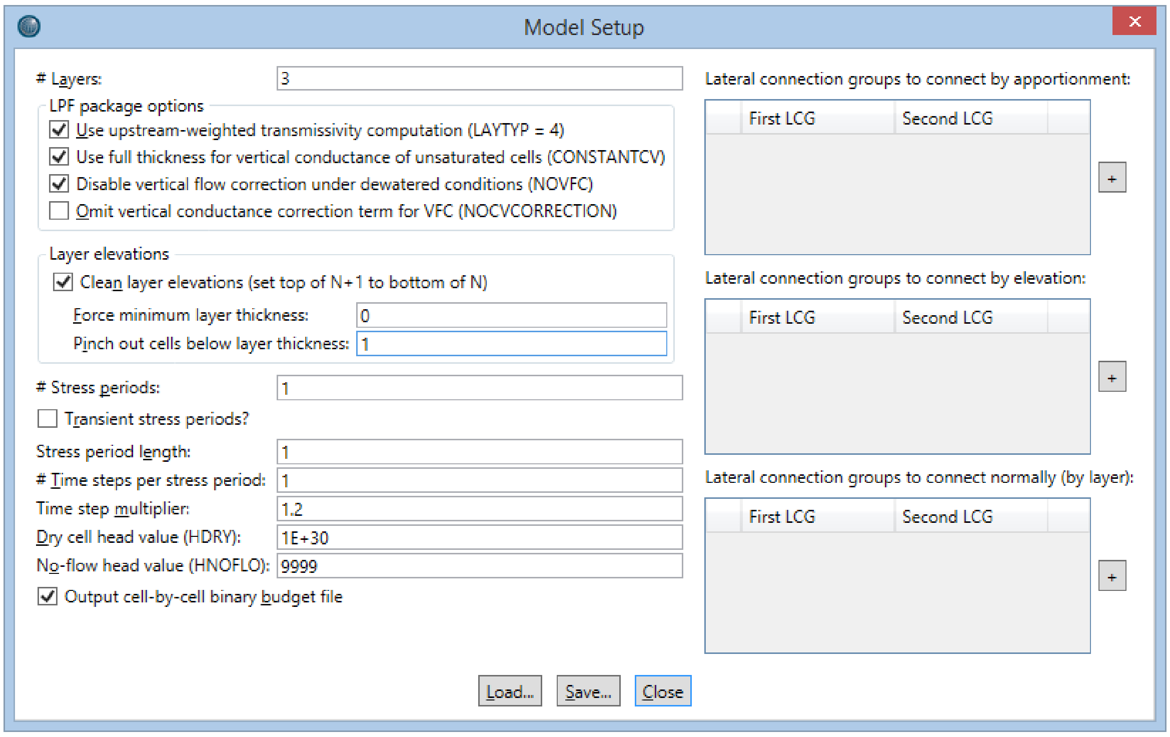


Figure 85: The new property TIN-based layer 2 bottom elevation mapping.

Pinching out Cells Based on Layer Thickness

Now that we have updated layer 2 such that it thins out to nothing in the south-west and north-east corners of the model, we can tell AlgoMesh to remove cells where they are not needed. For this example, we will tell it to pinch out any cells that have a thickness of less than 1m.

➢ Go to the **Model** menu and select **Model Setup**.
➢ In the **Layer elevations** box, change **Pinch out cells below layer thickness** to **1**. Your Model Setup window should now look like the one shown in Figure 86.
➢ Click the **Close** button to close the Model Setup window.


Model Setup

Layers: 3

LPF package options
- [✓] Use upstream-weighted transmissivity computation (LAYTYP = 4)
- [✓] Use full thickness for vertical conductance of unsaturated cells (CONSTANTCV)
- [✓] Disable vertical flow correction under dewatered conditions (NOVFC)
- [ ] Omit vertical conductance correction term for VFC (NOCVCORRECTION)

Layer elevations
- [✓] Clean layer elevations (set top of N+1 to bottom of N)

Force minimum layer thickness: 0
Pinch out cells below layer thickness: 1

Stress periods: 1
- [ ] Transient stress periods?

Stress period length: 1
Time steps per stress period: 1
Time step multiplier: 1.2
Dry cell head value (HDRY): 1E+30
No-flow head value (HNOFLO): 9999
- [✓] Output cell-by-cell binary budget file

Lateral connection groups to connect by apportionment:

First LCG	Second LCG

Lateral connection groups to connect by elevation:

First LCG	Second LCG

Lateral connection groups to connect normally (by layer):

First LCG	Second LCG

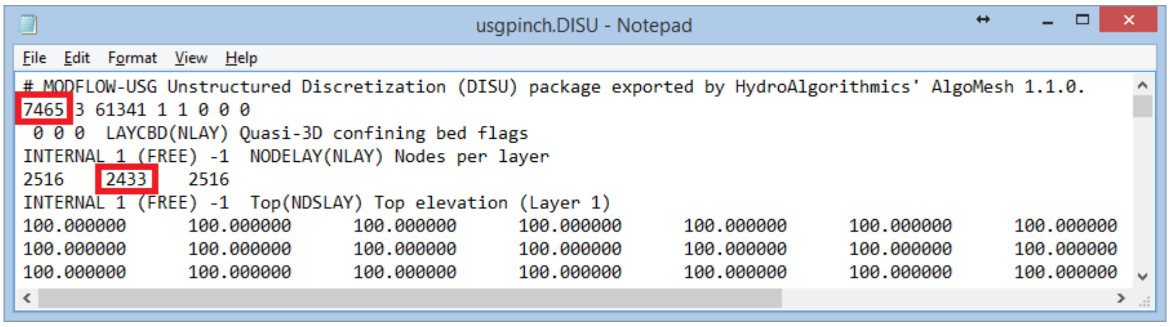
Figure 86: The model settings, adjusted to remove cells below 1m in thickness.

That is all we need to do to vary our layer 2 thickness across the model and pinch out cells below 1m thickness. We can now export MODFLOW-USG model files.

- Go to the **File** menu, select **Voronoi Mesh** and then **Export to MODFLOW-USG**.
- **Browse to your working folder**.
- Type an appropriate prefix for your adjusted model, e.g. **usgpinch**, and click **Save**. AlgoMesh will write the model files to disk.

It is a good idea to inspect the model files to check that the cells we wanted to remove are gone.

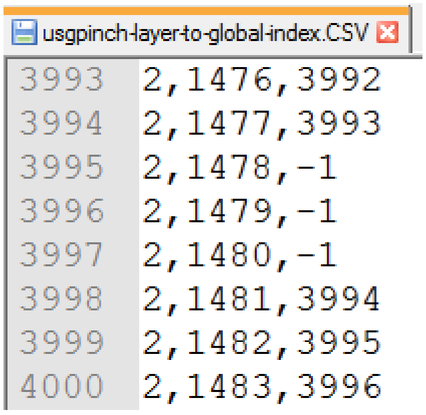
- Open the **usgpinch.DISU** file in your favourite text editor (e.g. **notepad**).
- Read the numbers in the fifth line in the file - immediately after the line that says **Nodes per layer**. There should be three numbers there - one for each of the layers, in order. The second number, for layer 2, should report a lower number of cells than the other two numbers. If this is the case, some cells have successfully been pinched out. Figure 87 shows an example of the DISU file contents.
- You may also want to **compare the numbers** in this file to those that were produced **for the previous version** of the model without pinch-outs. The first number on the second line of the file reports the total number of cells in the model, and this should also have reduced from the previous version.



```
# MODFLOW-USG Unstructured Discretization (DISU) package exported by HydroAlgorithmics' AlgoMesh 1.1.0.
7465 3 61341 1 1 0 0 0
 0 0 0  LAYCBD(NLAY) Quasi-3D confining bed flags
INTERNAL 1 (FREE) -1  NODELAY(NLAY) Nodes per layer
2516    2433    2516
INTERNAL 1 (FREE) -1  Top(NDSLAY) Top elevation (Layer 1)
100.000000      100.000000      100.000000      100.000000      100.000000      100.000000      100.000000
100.000000      100.000000      100.000000      100.000000      100.000000      100.000000      100.000000
100.000000      100.000000      100.000000      100.000000      100.000000      100.000000      100.000000
```

Figure 87: The unstructured discretisation (DISU) file produced by AlgoMesh for the adjusted model with cells pinched out in layer 2. The upper red rectangle highlights the total model cell count, and the lower red rectangle highlights the layer 2 cell count. Both of these are reduced from the previous version of the model.

You may also wish to explore the contents of the usgpinch-global-to-layer-index.CSV and usgpinch-layer-to-global-index.CSV files, as the removal of cells in layer 2 affects the global cell index values. Any -1 value for a global cell index in the usgpinch-layer-to-global-index.CSV file means that the cell has been removed from the model; see for example Figure 88. In the excerpt shown in the figure, cells with 2D mesh cell indices 1478, 1479 and 1480 in layer 2 have been pinched out. In the reverse mapping file, usgpinch-global-to-layer-index.CSV, these cells will not be present at all, as they have no corresponding global cell index.



```
3993  2,1476,3992
3994  2,1477,3993
3995  2,1478,-1
3996  2,1479,-1
3997  2,1480,-1
3998  2,1481,3994
3999  2,1482,3995
4000  2,1483,3996
```

Figure 88: The -layer-to-global-index.CSV mapping file contains a -1 value for any cells that do not exist in the model (i.e. they have no global cell index).

Visualising the Model in ParaView®

To validate further, it may also be useful to visualise the model in a tool such as ParaView. ParaView® is an open-source, third-party visualisation application, which at the time of writing is available for free download from the internet. When AlgoMesh writes MODFLOW-USG model files, it also produces a .VTK file that can be read by ParaView. Loading the file **usgpinch.vtk** into ParaView gives us a 3D view of the model geometry, as shown in Figure 89.

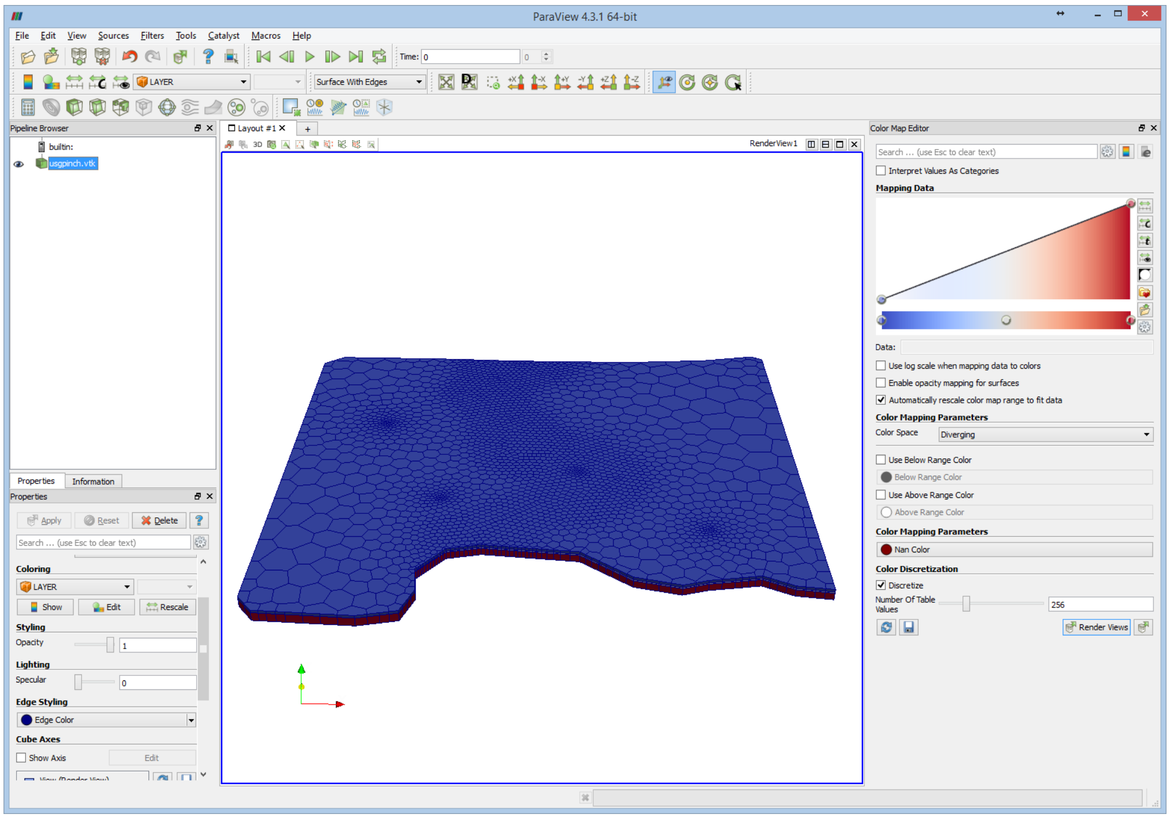


Figure 89: The open-source third-party tool ParaView allows visualisation of 3D model geometry produced by AlgoMesh.

Adding a transform filter to apply some vertical exaggeration, and extracting just the cells in layer 2, allows us to see the effect of our varied layer thickness and pinched-out cells (Figure 90).

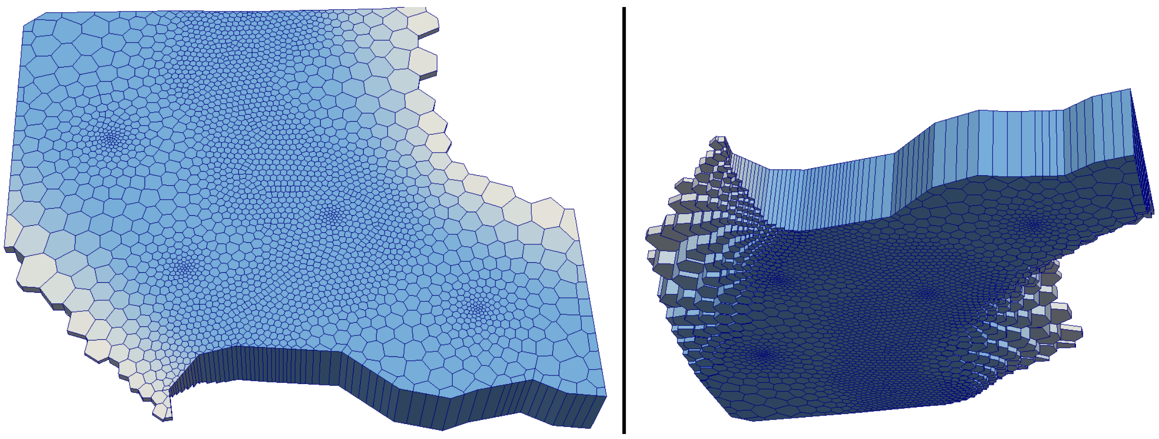


Figure 90: 3D views of the cells in layer 2 from the top and from underneath, showing the varied cell thicknesses and the omission of cells where the layer pinches out.

Other Methods of Removing Cells

If you want more control over which cells are removed from a model, there are alternative methods you can use to tell AlgoMesh which cells to include and which to exclude.

The first is to use a polygon mapping to the Active Model Area property. This property should be set to 1 in areas that are to be included in the model, and 0 for areas that should be removed from the model. The default value for this property is 1; that is, any areas of the model that are not covered by an Active Model Area mapping are assumed to be included in the model. You can either use a single mapping from polygons with a field value of 0 to exclude certain areas from the model, or you can use a constant property mapping to a value of 0 over the entire domain to exclude cells by default, and then map a smaller nested polygon with a field value of 1 inside the domain where your active model area is. As these mappings can be put in place using either the same or different polygons for each layer, this gives substantial flexibility in controlling the inclusion or exclusion of cells in the model.

The other method is to produce a custom CSV table with a field mapped to the Active Model Area property. The property values work in the same way as for the polygon method, i.e. 1 to include a cell and 0 to remove a cell, but this method allows control on a precise cell-by-cell basis. Note that when using this method, cells must be indexed by a combination of their layer and 2D mesh cell index. Global cell indices may not be used when mapping to Active Model Area, as the act of excluding cells from the model necessarily alters the global cell numbering.

3.11 Load MODFLOW-USG Results

Although AlgoMesh does not offer any comprehensive visualisation or post-processing capabilities, it does allow you to read in the binary head or drawdown files produced by MODFLOW-USG, interrogate them or use them for property mapping as a property TIN, and export the values to a CSV format to be processed in other tools. This how-to details the primary options available for dealing with MODFLOW-USG results.

Before starting this tutorial, you must have completed the Set up a Steady-State MODFLOW-USG Model how-to, as we will start from the AlgoMesh project for the steady-state model that was built in that section.

- Go to the **File** menu and choose **Open Project**.
- Browse to the folder in which you saved your model from the steady-state tutorial, select the .amproj file you saved (e.g. **USGSteadyState.amproj**), and click **Open**.

Important note: *when using AlgoMesh to read in binary result files from MODFLOW-USG, it is critical that the mesh that is loaded and any elevation or active area mapping data are identical to that used to export the MODFLOW-USG model files for the run. As it uses an unstructured grid formulation, MODFLOW-USG references each cell in its results output by its global cell index; there are no row, column indices as there would be in a structured grid version of MODFLOW. All the geometry of the three-dimensional grid used to build the model must remain the same; otherwise, the global cell numbering will change and result values may be assigned to the wrong cells. So be sure to use the same AlgoMesh project for reading results as you used for model file export, and do not alter the grid in any way once the model files have been exported.*

With the steady-state model loaded, we will first ask AlgoMesh to convert the binary head file usgss.HDS to a property TIN, so that we can load and manipulate the head surface for each layer in the AlgoMesh user interface.

- Go to the **File** menu, select **Voronoi Mesh** and then **Convert MODFLOW-USG Binary Head/ Drawdown to MDD**.
- **Browse to your working folder**.
- Select the binary head file that was produced by the MODFLOW-USG run, e.g. **usgss.HDS**, and click **Open**. The open file window closes and a new save file window appears, prompting you for a filename for the property TIN to save.

- Type an appropriate name for the steady-state head property TIN, e.g. **usgss-heads.mdd**, and click **Save**.
- Once it has converted the file, you will be asked if you would like to load this MDD file now as a property TIN. Select **Yes**. Note that if you selected No here, you could still come back later and load the property TIN into AlgoMesh by right-clicking on the Property TINs header and choosing New Property TIN.
- **Scroll down** in the project explorer pane on the left-hand side of the AlgoMesh window until you can see the **Property TINs section**. A new entry for the **usgss-heads** TIN should have appeared beneath the Property TINs header, as in Figure 91.

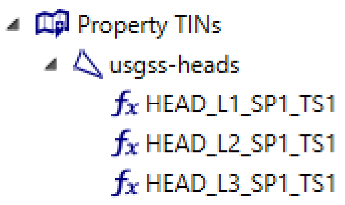


Figure 91: The head results from the MODFLOW-USG simulation have been added to the AlgoMesh project as a property TIN.

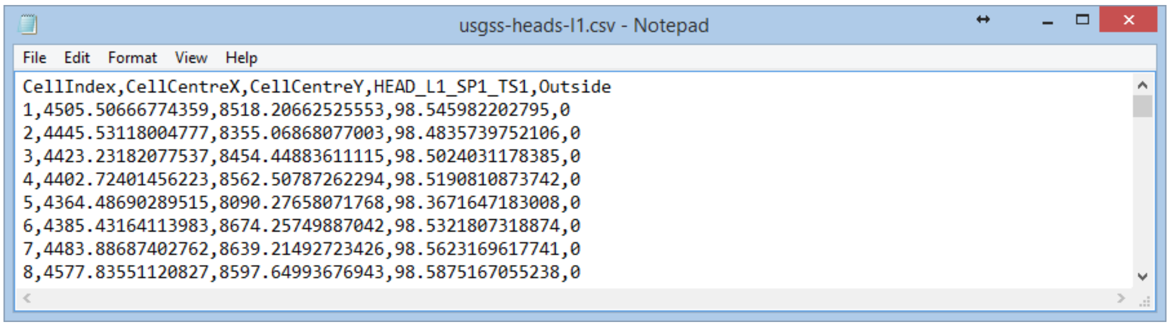
Note that the new property TIN was written to disk in the MDD (Mesh Domain Description) file format. This is an AlgoMesh-specific textual file format that allows the geometry for a triangulation (or mesh domain) to be specified. During the conversion process you just performed, AlgoMesh wrote into this file the X, Y coordinate location of a point at each mesh cell centre, with three corresponding property values attached to each: layer 1 head (HEAD_L1_SP1_TS1), layer 2 head (HEAD_L2_SP1_TS1) and layer 3 head (HEAD_L3_SP1_TS1). There is only one property for each layer here, as our model only had the one steady-state stress period comprising a single time step. If we had loaded in results from a transient simulation, we would see one property for each layer, stress period and time step, each with a name of the form *HEAD_L<Layer #>_SP<Stress Period #>_TS<Time Step #>*.

Much like any other property in the AlgoMesh project, you can select one of these and interrogate it to obtain an interpolated value at any given point by moving your mouse over the mesh and looking at the value displayed in the lower-right corner of the AlgoMesh window. You can even map these to a model property such as Initial Head, for use in a subsequent transient simulation - as we will do in the Set up a Transient MODFLOW-USG Model how-to.

To do more than this - for example to produce colour-fill or contour maps from the simulated head data - we need to export this data to CSV and use a third-party tool, such as Golden Software's Surfer®.

- **Right-click** the **HEAD_L1_SP1_TS1** property.
- From the context menu that appears, choose **Voronoi Mesh** and then **Export to CSV (Cell Centre Values).**
- Type a name for the CSV file to be exported, e.g. **usgss-heads-l1.csv**, and click **Save**.

AlgoMesh writes a list of entries to the CSV file including 2D mesh cell index, cell centre X and Y coordinates, and the head result value from the property HEAD_L1_SP1_TS1 (layer 1 head at the end of the steady-state simulation). You can open this file in a text editor to inspect the contents (see Figure 92).



```
CellIndex,CellCentreX,CellCentreY,HEAD_L1_SP1_TS1,Outside
1,4505.50666774359,8518.20662525553,98.545982202795,0
2,4445.53118004777,8355.06868077003,98.4835739752106,0
3,4423.23182077537,8454.44883611115,98.5024031178385,0
4,4402.72401456223,8562.50787262294,98.5190810873742,0
5,4364.48690289515,8090.27658071768,98.3671647183008,0
6,4385.43164113983,8674.25749887042,98.5321807318874,0
7,4483.88687402762,8639.21492723426,98.5623169617741,0
8,4577.83551120827,8597.64993676943,98.5875167055238,0
```

Figure 92: The first few lines of the exported CSV file containing layer 1 heads at each cell centre from the steady-state model run.

Now that the data is outside of AlgoMesh, other software tools may be used to post-process or visualise it. For example, Figure 93 shows a colour-filled contour map of the layer 1 heads produced with Surfer®.

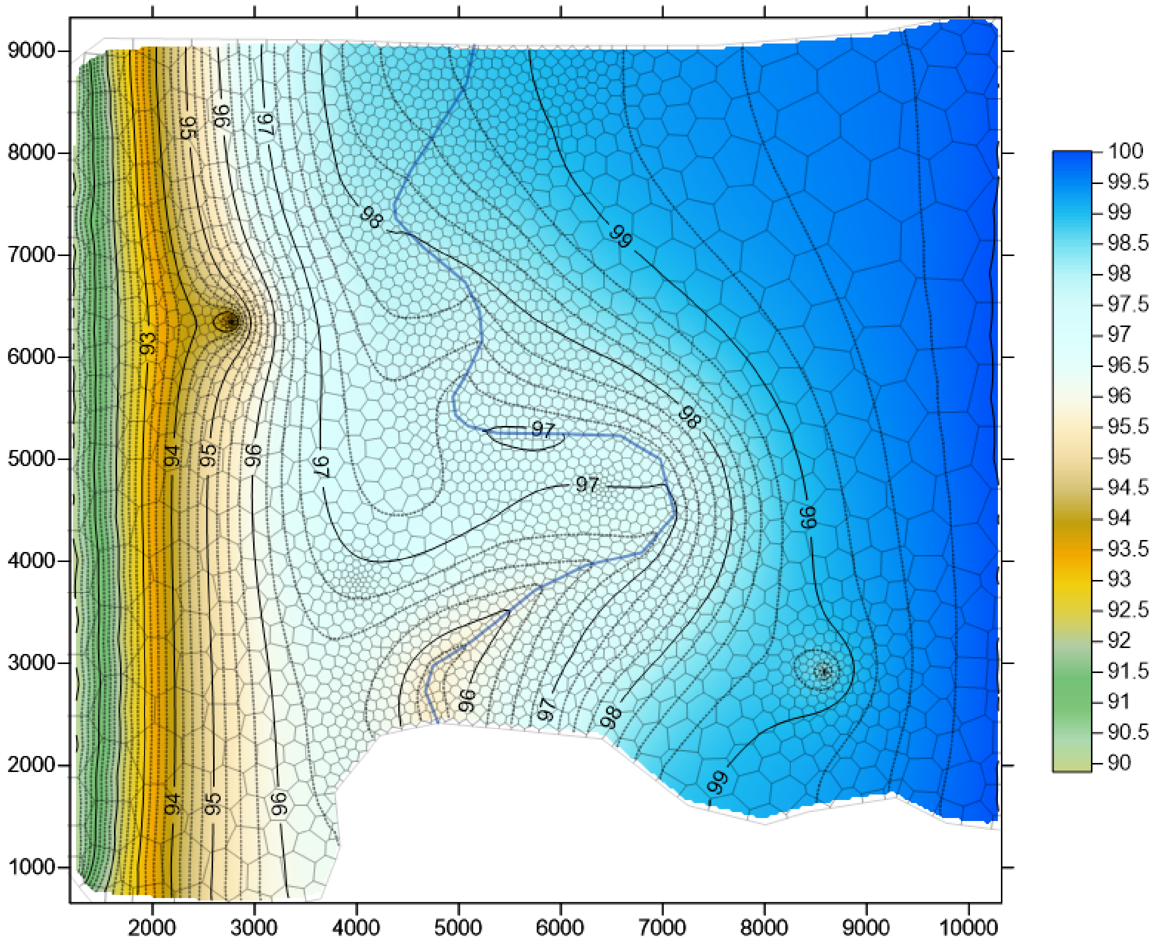
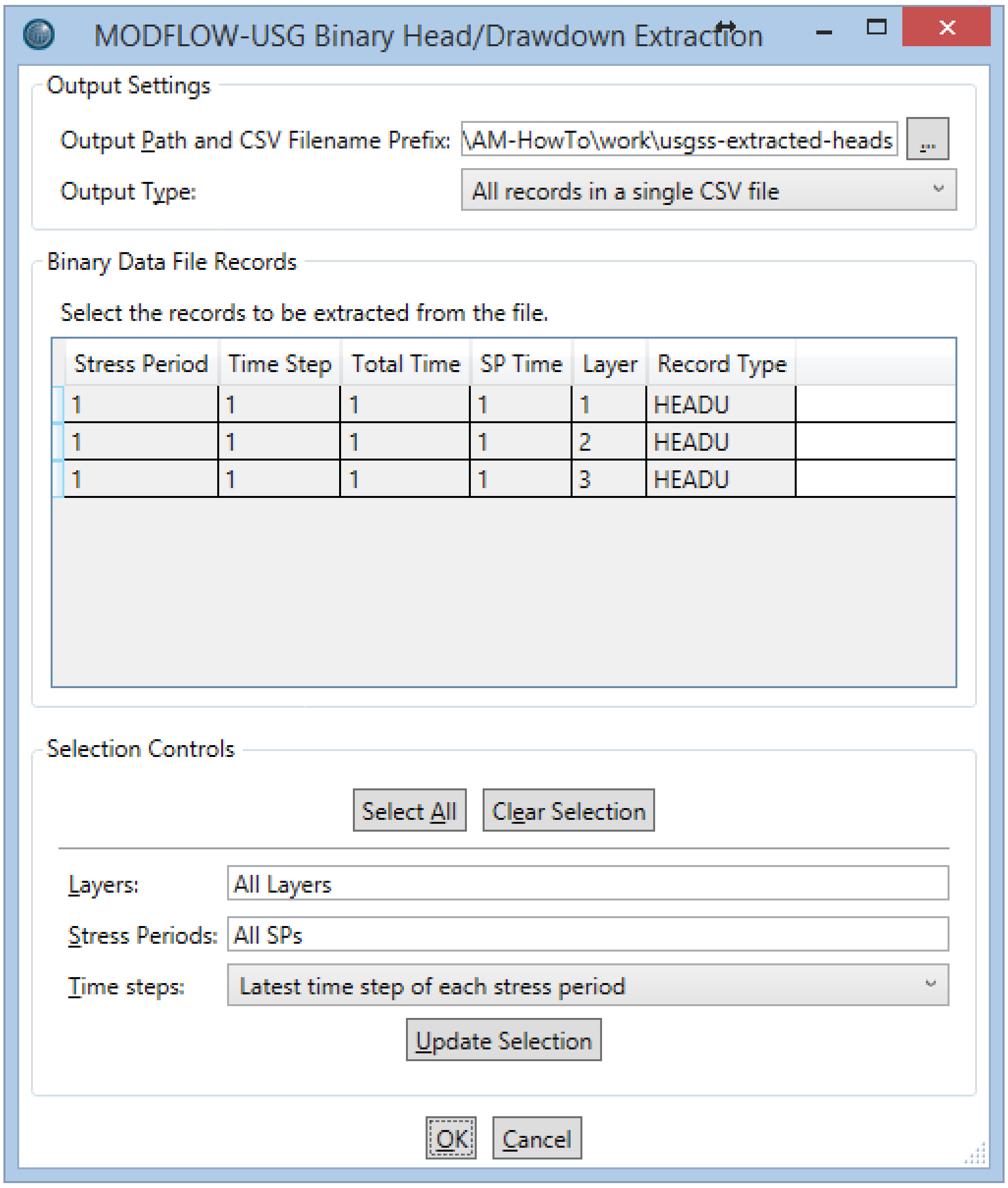


Figure 93: A colour-fill plot and contour map of layer 1 steady-state heads produced from the AlgoMesh CSV output by Golden Software's Surfer®.

Bulk Extraction of Heads from Large Transient Models

With a large transient model, this method of converting to property TINs can become unwieldy, as you must deal with one property per layer, stress period and time step combination that you need. To extract head results from many layers and time steps in bulk, AlgoMesh has an alternative tool.

- Go to the **File** menu, select **Voronoi Mesh** and then **Extract MODFLOW-USG Binary Head/ Drawdown to CSV**.
- Select the **usgss.HDS** file and click **Open**. A window appears listing all of the result records in the file, as shown in Figure 94.
- Provide an appropriate path and CSV filename prefix, e.g. <your working folder>**usgss-extracted-heads**
- Choose an **output type**: **All records in a single CSV file** if you want all data to be listed in a single file, or **One CSV file per record** if you want multiple CSV files to be generated.
- By default, records for all layers corresponding to the latest time step in each stress period will be selected - since we only had one time step in our simulation, this is all three records in the file. Leave this as-is.
- Click **OK** to generate the CSV file. An information message pops up telling you how many rows have been written to the CSV file; click **OK** to dismiss this.



Output Settings

Output Path and CSV Filename Prefix: \AM-HowTo\work\usgss-extracted-heads

Output Type: All records in a single CSV file

Binary Data File Records

Select the records to be extracted from the file.

Stress Period	Time Step	Total Time	SP Time	Layer	Record Type
1	1	1	1	1	HEADU
1	1	1	1	2	HEADU
1	1	1	1	3	HEADU

Selection Controls

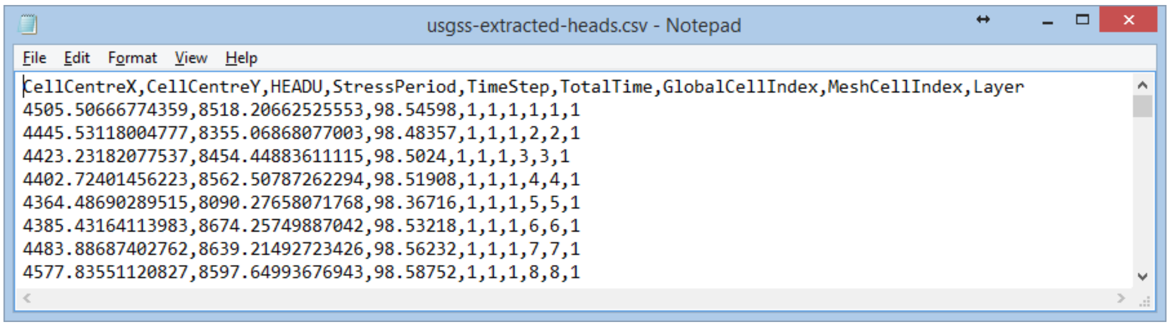
Layers: All Layers

Stress Periods: All SPs

Time steps: Latest time step of each stress period

Figure 94: The bulk head result extraction window. Any combination of records from a binary result file may be selected for export to CSV, and results may be written to one or many CSV files.

The resulting CSV may be opened in a text editor for inspection, and is of the form shown in Figure 95.


usgss-extracted-heads.csv - Notepad
File Edit Format View Help

```
CellCentreX,CellCentreY,HEADU,StressPeriod,TimeStep,TotalTime,GlobalCellIndex,MeshCellIndex,Layer
4505.50666774359,8518.20662525553,98.54598,1,1,1,1,1,1
4445.53118004777,8355.06868077003,98.48357,1,1,1,2,2,1
4423.23182077537,8454.44883611115,98.5024,1,1,1,3,3,1
4402.72401456223,8562.50787262294,98.51908,1,1,1,4,4,1
4364.48690289515,8090.27658071768,98.36716,1,1,1,5,5,1
4385.43164113983,8674.25749887042,98.53218,1,1,1,6,6,1
4483.88687402762,8639.21492723426,98.56232,1,1,1,7,7,1
4577.83551120827,8597.64993676943,98.58752,1,1,1,8,8,1
```

Figure 95: The content of the extracted CSV file.

Further Post-Processing of MODFLOW-USG Results

A number of other third-party utilities are available for post-processing MODFLOW-USG files. In particular, the Groundwater Data Utilities suite that is available alongside the PEST software package contains a number of command-line utilities that may be used to process binary results files. For example, the utility USGMOD2OBS may be used for extracting time-series groundwater hydrographs from a transient model run. At the time of writing, these utilities are available for free download from pesthomepage.org.

3.12 Set up a Transient MODFLOW-USG Model

In this how-to, you will learn how to convert an existing steady-state MODFLOW-USG model into a transient model.

Before starting this tutorial, you must have completed the Set up a Steady-State MODFLOW-USG Model how-to, as we will start from the AlgoMesh project for the steady-state model that was built in that section. It is also recommended that you first complete the Load MODFLOW-USG Results tutorial, as we will be reading binary head results from the steady-state simulation to use as initial heads for our transient simulation.

- ➢ Go to the **File** menu and choose **Open Project**.
- ➢ Browse to the folder in which you saved your model from the steady-state tutorial, select the .amproj file you saved (e.g. **USGSteadyState.amproj**), and click **Open**.

To convert our steady-state model to transient, we need to do the following:

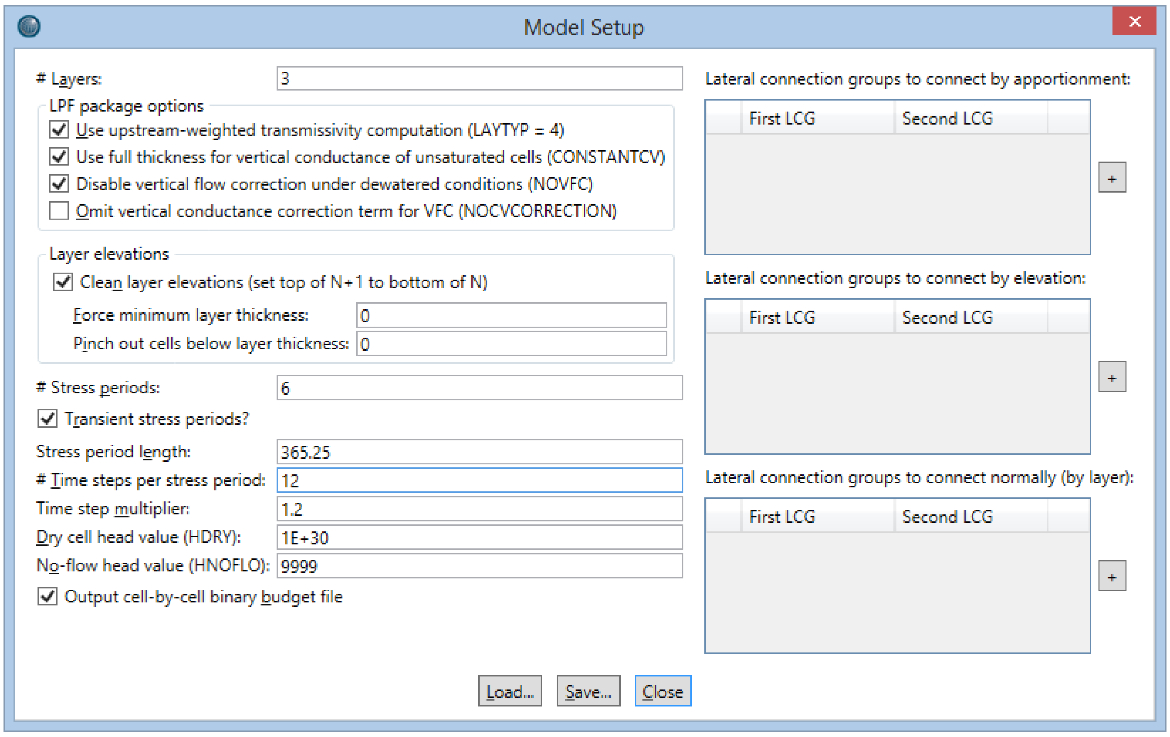
1. Change the stress period settings in Model Setup.
2. Map property values for specific storage (Ss) and specific yield (Sy).
3. Map appropriate values for Initial Head at the start of the transient simulation (here we will use the heads output from the steady-state simulation).
4. Set the stress periods for which existing boundary conditions should be active. Optionally, we could add more boundary conditions that are only present in the transient simulation, or change the values mapped for existing boundary conditions depending on the stress period. for this tutorial, we will simply activate all of the existing boundary conditions from the steady-state model for the entire transient simulation, except for some of the wells, for which pumping will be turned on and off in alternating stress periods.
5. Export the MODFLOW-USG model, make any desired manual modifications, and run it.

Model Setup

First, we need to tell AlgoMesh that we are running a transient model, and give it information about the stress periods in the simulation. For this how-to, our simulation will comprise six one-year transient stress periods.

➢ Go to the **Model** menu and select **Model Setup**. The Model Setup window appears.
➢ Set **# Stress periods** to **6**.
➢ Turn on **Transient stress periods**.
➢ Set **Stress period length** to **365.25** (one year in days).
➢ Set **# Time steps per stress period** to **12**.
➢ Leave **Time step multiplier** at **1.2**. Your Model Setup window should look like Figure 96.
➢ Click the **Close** button to close the window.

Note that AlgoMesh assumes that all of your stress periods have the same length and number of time steps. To assign different lengths or time step configuration to each stress period, you can use a text editor to edit the DISU file that AlgoMesh produces after you export the MODFLOW-USG files - the stress period definitions are specified in a fairly straightforward format at the very end of the file (see the MODFLOW-USG input/output format documentation for more details).


Model Setup

# Layers: 3

LPF package options
- [✓] Use upstream-weighted transmissivity computation (LAYTYP = 4)
- [✓] Use full thickness for vertical conductance of unsaturated cells (CONSTANTCV)
- [✓] Disable vertical flow correction under dewatered conditions (NOVFC)
- [ ] Omit vertical conductance correction term for VFC (NOCVCORRECTION)

Layer elevations
- [✓] Clean layer elevations (set top of N+1 to bottom of N)

Force minimum layer thickness: 0
Pinch out cells below layer thickness: 0

# Stress periods: 6
- [✓] Transient stress periods?

Stress period length: 365.25
# Time steps per stress period: 12
Time step multiplier: 1.2
Dry cell head value (HDRY): 1E+30
No-flow head value (HNOFLO): 9999
- [✓] Output cell-by-cell binary budget file

Lateral connection groups to connect by apportionment:

First LCG	Second LCG

Lateral connection groups to connect by elevation:

First LCG	Second LCG

Lateral connection groups to connect normally (by layer):

First LCG	Second LCG

Figure 96: To run a transient simulation, we need to tell AlgoMesh that transient stress periods are to be used, and give length and time stepping configuration for these.

Mapping Storage Properties

Storage properties need to be present in any transient MODFLOW-USG simulation. We can map these in AlgoMesh using the same techniques we used to map our hydraulic conductivities.

The specific storage (Ss) and specific yield (Sy) properties we will be using for our model are as follows:

	Ss	Sy
Layer 1 (Regolith / Alluvium)	1e-5 / 1e-5	0.1 / 0.2
Layer 2 (Aquitard)	1e-5	0.005
Layer 3 (Aquifer)	1e-5	0.02

Notice that the ConstantProperties layer we imported for our steady-state model already has properties Ss1, Ss2, Ss3, Sy1, Sy2 and Sy3, which cover everything except the storage in the alluvium. Start by mapping the ConstantProperties attributes.

- In the project explorer on the left-hand side of the AlgoMesh window, **scroll down** until you see the **Ss1** property beneath the **ConstantProperties** polygon set.
- **Right-click** on the **Ss1** property and choose **Map to Mesh Variable as Overlay** and then **Specific Storage**. A Specific Storage (Layer 1) mapping appears, which is what we want.
- **Right-click** on the **Ss2** property and choose **Map to Mesh Variable as Overlay** and then **Specific Storage**. A Specific Storage (Layer 1) mapping appears.
- **Right-click** on the **Specific Storage (Layer 1)** mapping underneath the **Ss2** property and select **Edit Layers**.
- Type **2** and hit **Enter** to set the mapping to Specific Storage (Layer 2).
- Repeat the above steps to construct the remaining mappings:
  - **Ss3** => **Specific Storage (Layer 3)**
  - **Sy1** => **Specific Yield (Layer 1)**
  - **Sy2** => **Specific Yield (Layer 2)**
  - **Sy3** => **Specific Yield (Layer 3)**

You should now have the specific storage and specific yield mappings set up as in Figure 97.

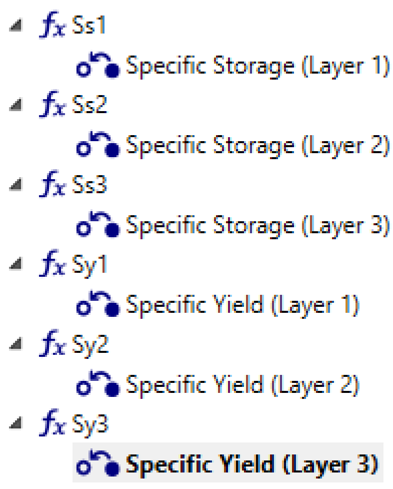


Figure 97: The storage property mappings for the ConstantProperties polygon set.

We have one more set of storage properties to map - those in the alluvium. We have a polygon set for the alluvium already, but it does not contain any Ss or Sy property fields, so we must add these as new properties in order to map them.

- In the project explorer on the left-hand side of the AlgoMesh window, **scroll up** until you see the **Alluvium** polygon set.
- **Right-click** on the **Alluvium** polygon set and choose **New Property**.
- Set the property name to **Ss1** and the default value to **1e-5**, and click **OK**. An Ss1 property entry

appears under the Alluvium polygon set.

- **Right-click** on the **Alluvium** polygon set again and choose **New Property**.
- Set the property name to **Sy1** and the default value to **0.2**, and click **OK**. An Sy1 property entry appears under the Alluvium polygon set.
- **Right-click** on the **Ss1** property and choose **Map to Mesh Variable as Overlay** and then **Specific Storage**. A Specific Storage (Layer 1) mapping appears, which is what we want.
- **Right-click** on the **Sy1** property and choose **Map to Mesh Variable as Overlay** and then **Specific Yield**. A Specific Yield (Layer 1) mapping appears, which is what we want.

Your alluvium polygon set properties should now look like those in Figure 98. The Alluvium polygon set should still be above the ConstantProperties set in the project explorer list, so the alluvium properties will override the regolith properties for any cells within the alluvium polygon.

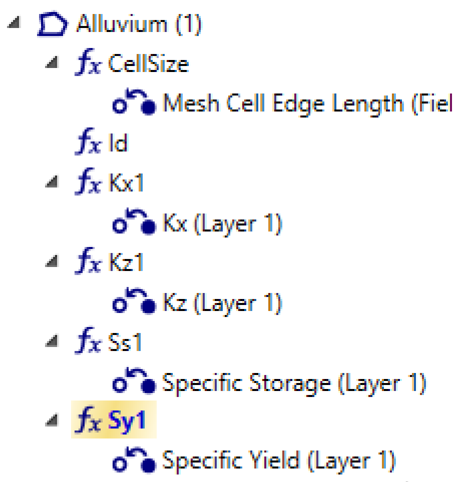


Figure 98: The alluvium properties after setting up the Ss1 and Sy1 mappings.

Mapping Steady-State Output Head as Transient Initial Head

The next step is to bring in the head outputs from our steady-state simulation and use them as starting heads for the transient simulation.

- In the project explorer, **scroll down** until you see the **Initial Head (All Layers)** mapping underneath the **ConstantProperties** set's **Top1** property.
- **Right-click** the **Initial Head (All Layers)** mapping.
- Select **Remove Mapping** from the context menu that appears. This will ensure that the initial heads we used in our steady-state simulation (top of model elevations) are no longer used.
- Go to the **File** menu, select **Voronoi Mesh** and then **Convert MODFLOW-USG Binary Head/ Drawdown to MDD**. An open file window appears.
- **Browse** to your **steady-state model working folder**.
- Select your steady-state head results file (e.g. **usgss.hds**) and click **Open**. The open file window closes and a Save As window opens, prompting you for the name of an MDD file to write.
- Give an appropriate name for the output MDD file, e.g. **usgss-heads.mdd**, and click **Save.**
- When prompted if you would like to load the MDD now as a property TIN, click **Yes**.
- In the project explorer, **scroll down** until you see the **usgss-heads** entry under **Property TINs**. There should be three head results entries underneath it: HEAD_L1_SP1_TS1, HEAD_L2_SP1_TS1 and HEAD_L3_SP1_TS1.
- **Right-click** the layer 1 head results entry **HEAD_L1_SP1_TS1** and choose **Map to Mesh Variable** and then **Initial Head**. An Initial Head (Layer 1) mapping appears.

- **Right-click** the layer 2 head results entry **HEAD_L2_SP1_TS1** and choose **Map to Mesh Variable** and then **Initial Head**. An Initial Head (Layer 2) mapping appears.
- **Right-click** the layer 3 head results entry **HEAD_L3_SP1_TS1** and choose **Map to Mesh Variable** and then **Initial Head**. An Initial Head (Layer 3) mapping appears.

Note that we did not need to adjust the layers of the new mappings this time; as we created them in layer order, AlgoMesh automatically assigned each new mapping to the next consecutive layer. This happens because we are mapping from property TIN properties, and only one property TIN mapping is allowed per mesh variable per layer (in this case Initial Head). Before, we were mapping from polygon properties, which "overlay" any other mappings in the order they are specified in the polygon set list and as such any number of polygon mappings is allowed for the same combination of mesh variable and layer.

You should now have a complete set of initial head mappings for the transient simulation based on the steady-state simulation head, as in Figure 99. To verify that the head values are reasonable, you can select any of the head properties that you just mapped and move your mouse cursor over the model domain, observing the head value reported in the bottom-right corner of the AlgoMesh window as you do so.

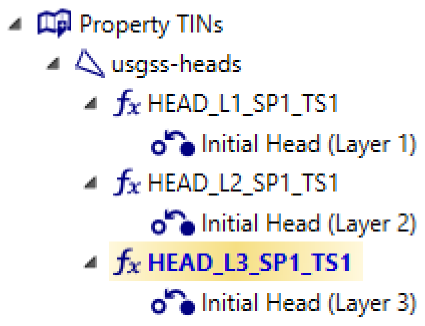


Figure 99: The steady-state output heads are mapped as initial heads for the transient simulation.

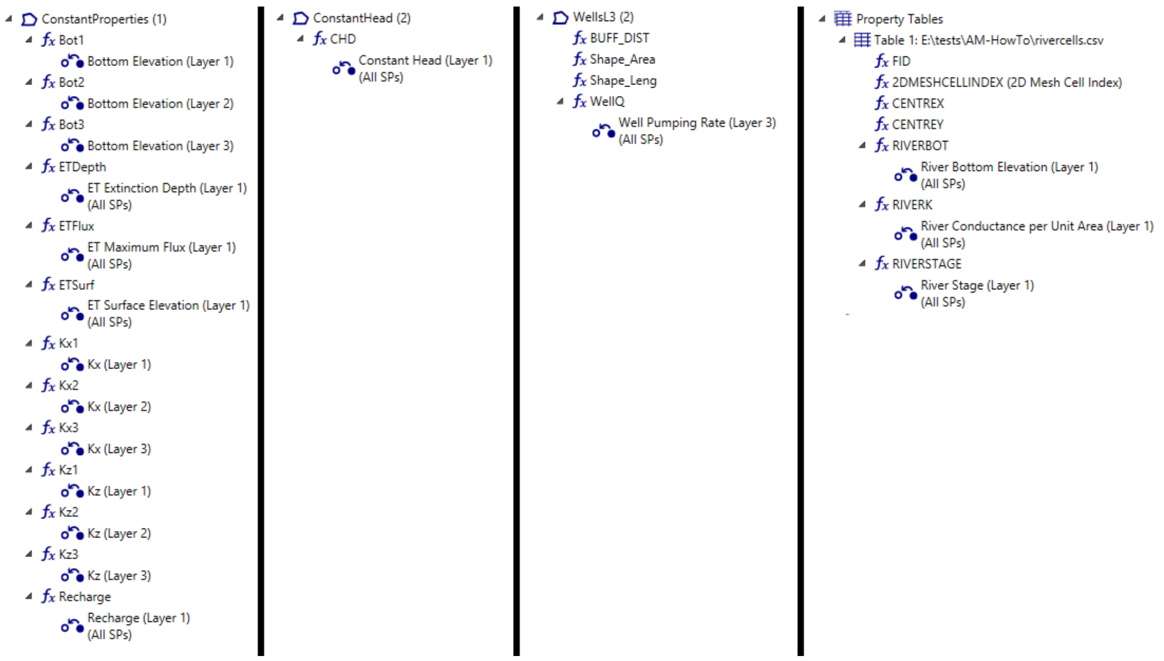
Setting Stress Periods for Boundary Condition Mappings

When we set up our steady-state model, we mapped recharge, evapotranspiration, constant head, river and well boundary conditions but left these at the default of only being active in stress period 1. This was sufficient for the steady-state simulation, as we only had one stress period. Now, however, we have six stress periods, so those boundary conditions that we want to be active for the entire simulation must be altered. We will do this now for all boundary conditions except the layer 1 pumping wells - we will alter those in a different way shortly.

- Find the **Recharge** property mapping beneath the **ConstantProperties** polygon set in the project explorer.
- **Right-click** the **Recharge (Layer 1) (SP 1)** mapping and choose **Edit Stress Periods** from the context menu that appears.
- Type **all** and hit **Enter** to change the mapping to **Recharge (Layer 1) (All SPs)**.
- Repeat the above steps to modify the existing mappings for the following variables such that they apply in all stress periods:
  - **ETDepth (ConstantProperties)** => **ET Extinction Depth (Layer 1) (All SPs)**
  - **ETFlux (ConstantProperties)** => **ET Maximum Flux (Layer 1) (All SPs)**
  - **ETSurf (ConstantProperties)** => **ET Surface Elevation (Layer 1) (All SPs)**
  - **CHD (ConstantHead)** => **Constant Head (Layer 1) (All SPs)**

- **WellQ (WellsL3)** => **Well Pumping Rate (Layer 3) (All SPs)** *(Note: be sure not to change the WellQ mapping in WellsL1 - only WellsL3's WellQ mapping should be changed)*
- **RIVERBOT (Table 1)** => **River Bottom Elevation (Layer 1) (All SPs)**
- **RIVERK (Table 1)** => **River Conductance per Unit Area (Layer 1) (All SPs)**
- **RIVERSTAGE (Table 1)** => **River Stage (Layer 1) (All SPs)**

The boundary condition mappings should now appear as in Figure 100.



```
ConstantProperties (1)
  Bot1
    Bottom Elevation (Layer 1)
  Bot2
    Bottom Elevation (Layer 2)
  Bot3
    Bottom Elevation (Layer 3)
  ETDepth
    ET Extinction Depth (Layer 1)
    (All SPs)
  ETFlux
    ET Maximum Flux (Layer 1)
    (All SPs)
  ETSurf
    ET Surface Elevation (Layer 1)
    (All SPs)
  Kx1
    Kx (Layer 1)
  Kx2
    Kx (Layer 2)
  Kx3
    Kx (Layer 3)
  Kz1
    Kz (Layer 1)
  Kz2
    Kz (Layer 2)
  Kz3
    Kz (Layer 3)
  Recharge
    Recharge (Layer 1)
    (All SPs)

ConstantHead (2)
  CHD
    Constant Head (Layer 1)
    (All SPs)

WellsL3 (2)
  BUFF_DIST
  Shape_Area
  Shape_Leng
  WellQ
    Well Pumping Rate (Layer 3)
    (All SPs)

Property Tables
  Table 1: E:\tests\AM-HowTo\rivercells.csv
    FID
    2DMESHCELLINDEX (2D Mesh Cell Index)
    CENTREX
    CENTREY
    RIVERBOT
      River Bottom Elevation (Layer 1)
      (All SPs)
    RIVERK
      River Conductance per Unit Area (Layer 1)
      (All SPs)
    RIVERSTAGE
      River Stage (Layer 1)
      (All SPs)
```

Figure 100: The boundary condition mappings are altered to apply to All SPs.

We now have only the layer 1 pumping wells remaining. We will activate these in every odd stress period only (stress periods 1, 3 and 5).

- Find the **WellQ** property mapping beneath the **WellsL1** polygon set in the project explorer.
- **Right-click** the **Well Pumping Rate (Layer 1) (SP 1)** mapping and choose **Edit Stress Periods** from the context menu that appears.
- Type **1,3,5** and hit **Enter** to change the mapping to **Recharge (Layer 1) (SPs 1,3,5)**.

Figure 101 shows the resulting layer 1 well mapping. As there is no Well Pumping Rate mapping for these wells in stress periods 2, 4 and 6, the wells will be deactivated in these stress periods. Alternatively, we could have used a second property mapped to Well Pumping Rate in SPs 2, 4 and 6 which specified a zero value for the pumping rate. If we did this, the wells would still appear for those stress periods in the WEL package file, but they would have the zero value specified for flux.

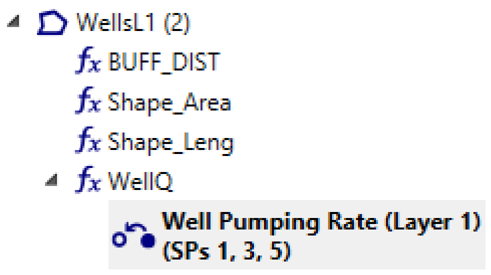


Figure 101: Layer 1 pumping wells are mapped only to the odd stress periods in the simulation (SPs 1, 3 and 5).

Exporting and Running the MODFLOW-USG Model

We now have a complete set of material properties and boundary conditions mapped for our transient model, and we are ready to build MODFLOW-USG input files for it.

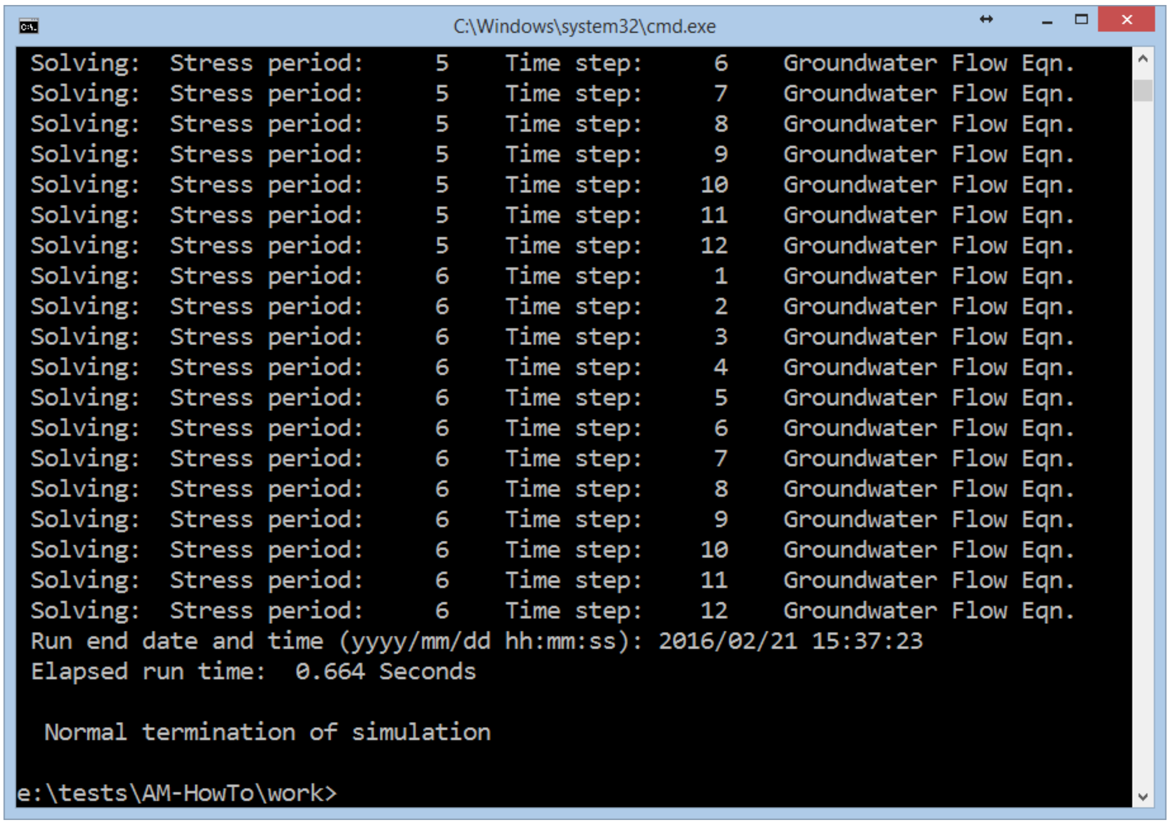
- Go to the **File** menu, then **Voronoi Mesh** and **Export to MODFLOW-USG**.
- **Browse to a location** to which you have write access, such as your Documents folder.
- *(Recommended)* **Create a new working folder** for the model files to be saved to. This will keep the numerous model files that are produced separate from other files on your computer. Alternatively, you can reuse the same working folder you used for the steady-state model.
- **Type a model name prefix**, e.g. **usgtr** and click **Save**. Note that you should not include a file name extension here, as this name is used to create a number of model files, each with a different extension.

AlgoMesh will write all of the files necessary to run the USG model. When it is done, you should see a message in the grey log pane in the bottom-right corner of the AlgoMesh window saying "Wrote MODFLOW-USG input files for Voronoi mesh."

As with the steady-state model, we need to make a manual change to apply the AUTOFLOWREDUCE keyword to the header of the WEL package input file before running the model.

- Open the file **usgtr.WEL** in your favourite text editor (you can use **Notepad**, which comes with Windows).
- At the end of the second line of the file, type **a space** and then the keyword **AUTOFLOWREDUCE**.
- **Save** the file and **Close** the text editor.

You can now run the model in the same way as you did for the steady-state model, but using the new **usgtr.NAM** file instead. This time you should see MODFLOW-USG proceed through all of the time steps and stress periods in the simulation, as in Figure 102.



```
Solving:  Stress period:    5   Time step:    6   Groundwater Flow Eqn.
Solving:  Stress period:    5   Time step:    7   Groundwater Flow Eqn.
Solving:  Stress period:    5   Time step:    8   Groundwater Flow Eqn.
Solving:  Stress period:    5   Time step:    9   Groundwater Flow Eqn.
Solving:  Stress period:    5   Time step:   10   Groundwater Flow Eqn.
Solving:  Stress period:    5   Time step:   11   Groundwater Flow Eqn.
Solving:  Stress period:    5   Time step:   12   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    1   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    2   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    3   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    4   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    5   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    6   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    7   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    8   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:    9   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:   10   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:   11   Groundwater Flow Eqn.
Solving:  Stress period:    6   Time step:   12   Groundwater Flow Eqn.
Run end date and time (yyyy/mm/dd hh:mm:ss): 2016/02/21 15:37:23
Elapsed run time:  0.664 Seconds

 Normal termination of simulation

e:\tests\AM-HowTo\work>
```

Figure 102: MODFLOW-USG progresses through all stress periods in the transient simulation until completion.

Inspecting the **usgtr.LST** file, once the simulation is complete, shows minimal mass balance error at all of the six stress periods.

You can use the **Extract MODFLOW-USG Binary Head/Drawdown to CSV** tool to output heads for each layer at the end of each stress period, or at any other time during the simulation, as detailed in the Load MODFLOW-USG Results how-to guide.

- Go to the **File** menu, select **Voronoi Mesh** and then **Extract MODFLOW-USG Binary Head/ Drawdown to CSV**.
- **Browse to the working folder.**
- Select **usgtr.HDS** and click **Open**.
- **Leave all the default values** in the head extraction window; this should be set to output a CSV file per record for all layers at the last time step in every stress period.
- **Click OK** to generate the CSV files.
- **Click OK** again when AlgoMesh notifies you of the number of rows exported to the CSV files.

After completing this process, you should have 18 CSV files in your working folder: usgtr_SP1_TS12_L1.csv, usgtr_SP1_TS12_L2.csv, usgtr_SP1_TS12_L3.csv, usgtr_SP2_TS12_L1.csv and so on. You can now bring these files into your favourite post-processing tool to produce image and contour maps.

Figure 103 shows examples of such maps produced from the transient model outputs for layer 1, at the end of each of the six stress periods. Note that the effect of activating the layer 1 wells in odd stress periods and deactivating them in even stress periods is evident in the resulting maps.

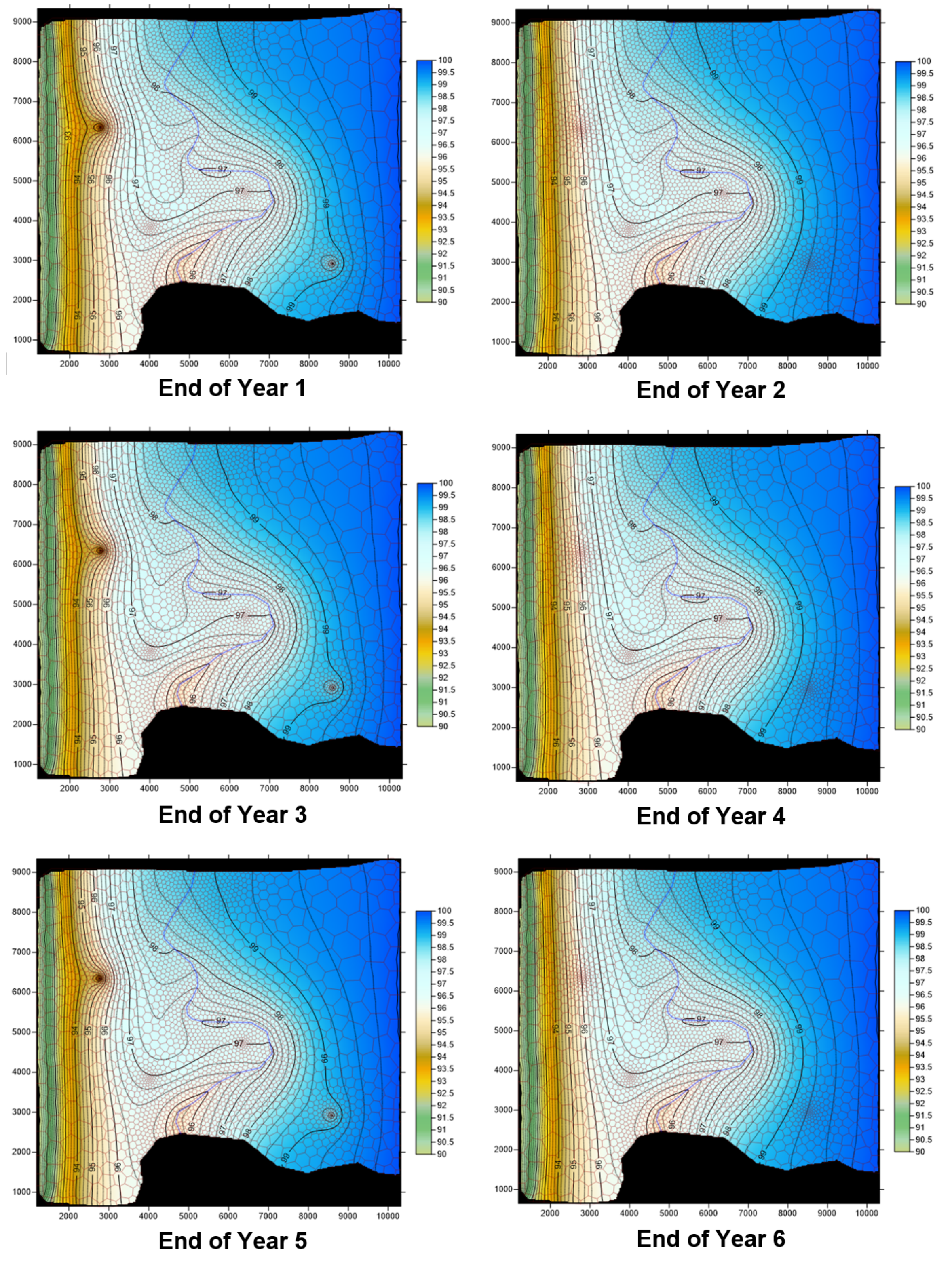


Figure 103: Head plots at the end of each year (stress period) of the transient simulation.

You have now set up and run a complete transient MODFLOW-USG model. Try experimenting with other boundary condition types, such as drains, and re-run the transient simulation to observe the effect your changes have on the results. You may wish to save your work before making any more changes.

- Go to the **File** menu and choose **Save Project**.
- **Browse to a location** to which you have write access, such as your Documents folder.
- **Type a name** for the project file to be saved, e.g. **USGTransient.amproj** and click **Save**.

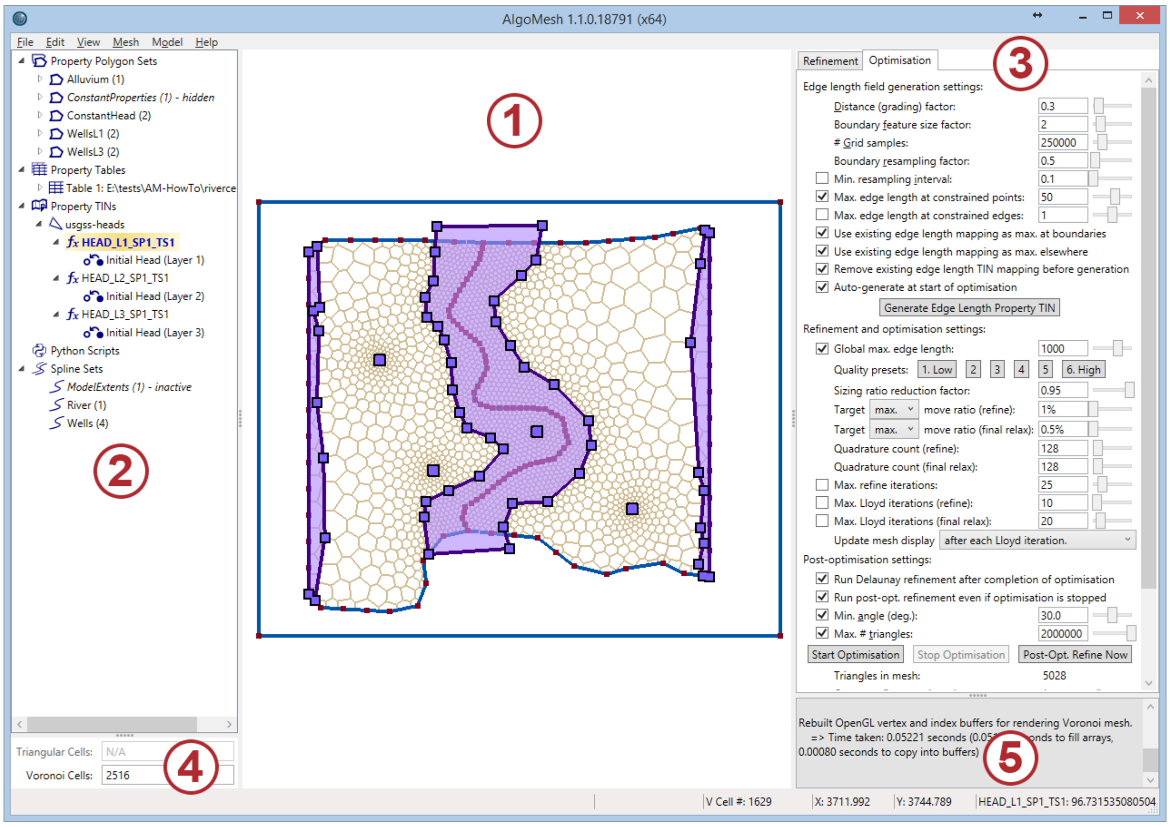
4 User Interface

The following sections present a reference of the AlgoMesh user interface and its various windows, menus and settings.

4.1 Main AlgoMesh Window

The main AlgoMesh window, shown in Figure 104, comprises a menu bar at the top, a status bar at the bottom, and five primary elements within the window:

1. The main view or *mesh view*, allowing display, navigation and interaction with the spatial data in an AlgoMesh project.
2. The project explorer, listing various types of input data and allowing mesh generation and model properties to be mapped.
3. The mesh generation panels, allowing the Delaunay refinement and multi-level optimisation algorithms to be set up and run.
4. The *cell count* display. If View->Triangles is enabled, the number of elements in the triangular mesh will be displayed. If View->Voronoi Cells is enabled, the number of polygonal cells in the Voronoi mesh will be displayed.
5. The *log panel*. This is updated whenever AlgoMesh performs an action on the mesh, or loads or saves a file. The time taken to perform most operations is reported here.



Triangular Cells: N/A
Voronoi Cells: 2516

Edge length field generation settings:
- Distance (grading) factor: 0.3
- Boundary feature size factor: 2
- # Grid samples: 250000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ☑ Max. edge length at constrained points: 50
- ☐ Max. edge length at constrained edges: 1
- ☑ Use existing edge length mapping as max. at boundaries
- ☑ Use existing edge length mapping as max. elsewhere
- ☑ Remove existing edge length TIN mapping before generation
- ☑ Auto-generate at start of optimisation

Generate Edge Length Property TIN

Refinement and optimisation settings:
- ☑ Global max. edge length: 1000
- Quality presets: 1. Low 2 3 4 5 6. High
- Sizing ratio reduction factor: 0.95
- Target max. move ratio (refine): 1%
- Target max. move ratio (final relax): 0.5%
- Quadrature count (refine): 128
- Quadrature count (final relax): 128
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:
- ☑ Run Delaunay refinement after completion of optimisation
- ☑ Run post-opt. refinement even if optimisation is stopped
- ☑ Min. angle (deg.): 30.0
- ☑ Max. # triangles: 2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 5028

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.05221 seconds (0.05[illegible] onds to fill arrays,
0.00080 seconds to copy into buffers)
```

V Cell #: 1629 | X: 3711.992 | Y: 3744.789 | HEAD_L1_SP1_TS1: 96.731535080504

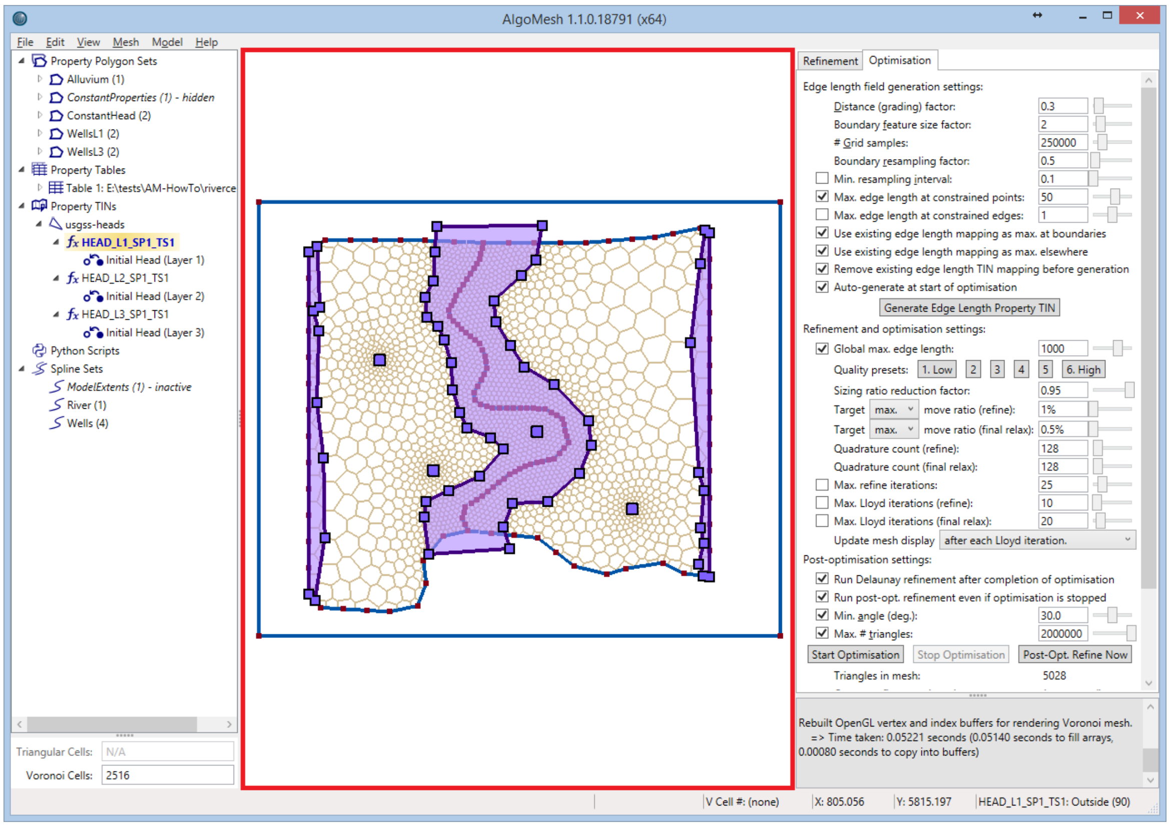
Figure 104: The main AlgoMesh window.

The status bar at the bottom of the AlgoMesh window presents the following information, from left to right, relevant to the current location of the mouse within the main view:

- The 2D triangular mesh cell index (*T Cell #*) of the triangular element at the mouse location. This is only shown if View->Triangles is enabled.
- The 2D Voronoi mesh cell index (*V Cell #*) of the Voronoi cell at the mouse location. This is only shown if View->Voronoi Cells is enabled.
- The X coordinate at the mouse location.
- The Y coordinate at the mouse location.
- The name and value of the selected property at the mouse location. This is only shown if a polygon set or TIN property entry is selected in the project explorer. If the mouse location is outside of the respective polygons or TIN, the value will be reported as Outside.

4.1.1 Main View

The main view is the central pane in the AlgoMesh window, outlined in red in Figure 105. It shows the current triangular and/or Voronoi mesh, constrained points and line segments, and any splines or property polygons that are active in the project explorer.



Triangular Cells: N/A
Voronoi Cells: 2516

Edge length field generation settings:
- Distance (grading) factor: 0.3
- Boundary feature size factor: 2
- # Grid samples: 250000
- Boundary resampling factor: 0.5
- [ ] Min. resampling interval: 0.1
- [✓] Max. edge length at constrained points: 50
- [ ] Max. edge length at constrained edges: 1
- [✓] Use existing edge length mapping as max. at boundaries
- [✓] Use existing edge length mapping as max. elsewhere
- [✓] Remove existing edge length TIN mapping before generation
- [✓] Auto-generate at start of optimisation

Refinement and optimisation settings:
- [✓] Global max. edge length: 1000
- Quality presets: 1. Low, 2, 3, 4, 5, 6. High
- Sizing ratio reduction factor: 0.95
- Target max. move ratio (refine): 1%
- Target max. move ratio (final relax): 0.5%
- Quadrature count (refine): 128
- Quadrature count (final relax): 128
- [ ] Max. refine iterations: 25
- [ ] Max. Lloyd iterations (refine): 10
- [ ] Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:
- [✓] Run Delaunay refinement after completion of optimisation
- [✓] Run post-opt. refinement even if optimisation is stopped
- [✓] Min. angle (deg.): 30.0
- [✓] Max. # triangles: 2000000

Triangles in mesh: 5028

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.05221 seconds (0.05140 seconds to fill arrays,
0.00080 seconds to copy into buffers)
```

V Cell #: (none) | X: 805.056 | Y: 5815.197 | HEAD_L1_SP1_TS1: Outside (90)

Figure 105: The main view, or mesh view, is the central pane of the AlgoMesh window, outlined here in red.

The View menu contains visibility switches that may be used to turn on and off display of constrained points, edges, triangles, Voronoi cells, splines and property polygons.

The main view may be navigated by clicking the left mouse button and dragging to pan, and by scrolling the mouse wheel up and down to zoom in and out. To restore the view such that it shows the entire mesh domain, use View->Zoom to Extents.

The mesh, splines and polygons may be manipulated in the main view using a combination of keyboard modifier keys and mouse buttons. The controls that may be used depend on the selected editing mode in the Edit menu:

- **Edit Mesh**:
  - **Click** to select a triangle in the triangular mesh.
  - **Ctrl+click** to create a new constrained point and select it.
  - **Ctrl+Shift+click** to create a new constrained point and join it to the selected triangular mesh vertex with an edge constraint.
  - **Shift+click** to join the selected triangular mesh vertex to the closest mesh vertex to the clicked mouse location, and select that vertex. If no mesh vertex is currently selected, this simply selects the closest mesh vertex to the clicked mouse location.
  - **Alt+click** to remove any edge constraint between the selected triangular mesh vertex and the closest mesh vertex to the clicked mouse location, and select that vertex. If no mesh vertex is currently selected, this simply selects the closest mesh vertex to the clicked mouse location.
  - **Esc** to deselect any selected mesh vertex or triangle.
- **Edit Splines**:

  - **Click** to select a point or segment of a spline under the mouse cursor.
  - **Right-click** to open the Spline Properties window for the spline under the mouse cursor. The individual segment properties shown in the window will relate to the specific spline segment under the mouse cursor when the right mouse button is clicked.
  - **Click and drag** to move a spline point under the mouse cursor to a new location.
  - **Ctrl+click** to start a new spline by creating a new point at the mouse location, and select the new point. If the mouse cursor is on top of a segment of an existing spline, the new point will snap to the existing spline, and it will be inserted between the two endpoints of the segment, instead of creating a new spline.
  - **Alt+click** to start a new spline by creating a point at the location of the nearest existing vertex in the underlying triangular mesh, and select the new point. This is useful for snapping to existing mesh geometry.
  - **Shift+click** to add a new segment to an existing spline from the selected spline point to the spline point under the mouse cursor. Does nothing if no spline point is selected, or if the mouse is not on top of another spline point.
  - **Ctrl+Shift+click** to create a new point as per Ctrl+click and adjoin it with a spline segment from the selected point.
  - **Alt+Shift+click** to create a new point snapped to a mesh vertex as per Alt+click and adjoin it with a spline segment from the selected point.
  - **Del** to delete the selected spline point.
  - **Esc** to deselect any selected spline point or segment, and close the Spline Properties window if it is open.
- **Edit Property Polygons**: identical controls to the Edit Splines mode, but acting on property polygons instead of splines.

4.1.2 Menus

The menus at the top of the user interface (see Figure 106) provide access to many of AlgoMesh's high level functions. Refer to the individual sections for details on each:

- The File menu.
- The Edit menu.
- The View menu.
- The Mesh menu.
- The Model menu.
- The Help menu.

Figure 106: The AlgoMesh menu bar.

4.1.2.1 File Menu

The File menu (see Figure 107) provides the functionality to load mesh data from an existing file, or to save the current mesh to a file, in any of the supported file formats.

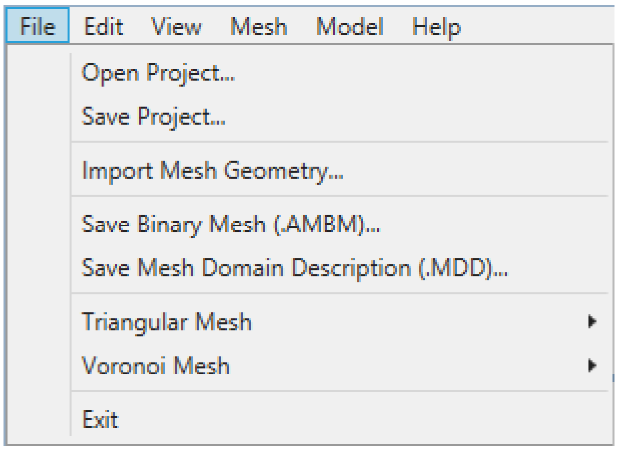


Figure 107: The File menu.

Open Project

Open a selected AlgoMesh project (.amproj) file. AlgoMesh projects store all mesh data, project explorer data and settings in the interface, except for CSV property tables, which are linked by a file name.

Save Project

Saves an AlgoMesh project (.amproj) file. AlgoMesh projects store all mesh data, project explorer data and settings in the interface, except for CSV property tables, which are linked by a file name.

Import Mesh Geometry

Reads point and line geometry from any of a number of common file formats (.shp, .dxf, .mid/.mif, and others) or AlgoMesh .mdd or .ambm formats, and adds it into the mesh.

Save Binary Mesh (.AMBM)

Saves the current triangular mesh in the proprietary AlgoMesh Binary Mesh (.ambm) format. Saving an AlgoMesh project (.amproj) file is typically recommended instead of .ambm, as it contains all information needed to accurately reconstruct a model, where .ambm does not.

Save Mesh Domain Description (.MDD)

Saves a Mesh Domain Description (.mdd) file containing the vertices, constrained line segments and inside/outside regions in the current triangular mesh, and optionally splines. This is useful when manually constructing property TINs.

Triangular and Voronoi Mesh Menus

Figure 108 shows the Triangular Mesh sub-menu. The Voronoi Mesh sub-menu is identical except that it does not contain the Export 2D Mesh to HydroGeoSphere option, as Voronoi meshes are not supported for this export method.

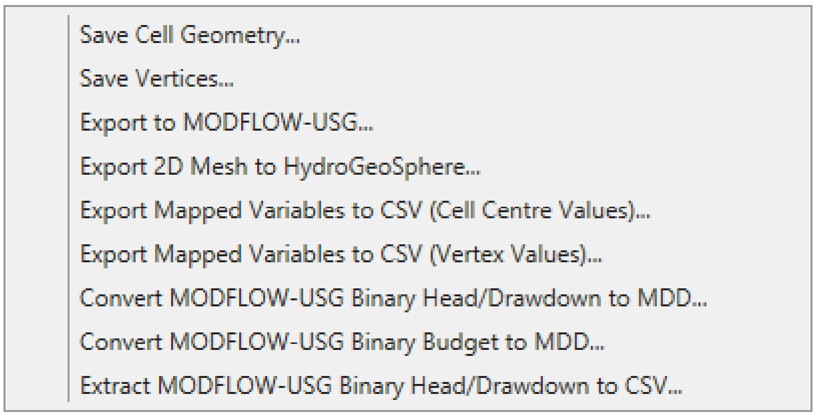

Save Cell Geometry...
Save Vertices...
Export to MODFLOW-USG...
Export 2D Mesh to HydroGeoSphere...
Export Mapped Variables to CSV (Cell Centre Values)...
Export Mapped Variables to CSV (Vertex Values)...
Convert MODFLOW-USG Binary Head/Drawdown to MDD...
Convert MODFLOW-USG Binary Budget to MDD...
Extract MODFLOW-USG Binary Head/Drawdown to CSV...

Figure 108: The File->Triangular Mesh menu. The File->Voronoi Mesh menu contains all of the same options except for HydroGeoSphere export.

Save Cell Geometry

Exports the triangular or Voronoi cells to one of a number of common file formats (.shp, .mid/.mif, .bna, and others). For formats that support attribute data, such as .shp, the 2D mesh cell indices, cell centre X and Y coordinates, and any mapped property and boundary condition values are exported with each cell as attributes.

Save Vertices

Exports the triangle vertices or Voronoi polygon vertices to one of a number of common file formats (.shp, .mid/.mif, .bna, and others). For formats that support attribute data, such as .shp, any mapped property and boundary condition values are exported with each cell as attributes.

Export to MODFLOW-USG

Writes MODFLOW-USG files for the current model. See the Set up a Steady-State MODFLOW-USG Model how-to section for an introduction to building MODFLOW-USG models using AlgoMesh.

Export 2D Mesh to HydroGeoSphere *(Triangular Mesh menu only)*

Exports the current 2D triangular mesh to an AlgoMesh-HydroGeoSphere 2D Mesh Interchange (.ah2) file.

Export Mapped Variables to CSV (Cell Centre Values)

Writes a comma-separated values (CSV) file containing all mapped property values at the centre of each cell in the mesh.

Export Mapped Variables to CSV (Vertex Values)

Writes a comma-separated values (CSV) file containing all mapped property values at each triangle vertex or Voronoi polygon vertex in the mesh.

Convert MODFLOW-USG Binary Head/Drawdown to MDD

Reads in a binary head or drawdown output from a MODFLOW-USG simulation based on the current mesh and model setup, and generates an MDD format file containing all of the result values, which may be loaded into AlgoMesh as a property TIN. See the Load MODFLOW-USG Results how-to guide for an introduction on reading MODFLOW-USG model results into AlgoMesh.

Convert MODFLOW-USG Binary Budget to MDD

Reads in a binary cell-by-cell budget output from a MODFLOW-USG simulation based on the current mesh and model setup, and generates an MDD format file containing all of the result values, which may be loaded into AlgoMesh as a property TIN.

Extract MODFLOW-USG Binary Head/Drawdown to CSV

Reads in a binary head or drawdown output from a MODFLOW-USG simulation based on the current mesh and model setup, and allows selection of all or a subset of result entries in the file for export to one or multiple CSV files. See the Load MODFLOW-USG Results how-to guide for an introduction on reading MODFLOW-USG model results into AlgoMesh, or see the MODFLOW-USG Binary Head/ Drawdown Extraction window section for a reference on the extraction options.

Exit

Shuts down AlgoMesh. Ensure you save your work before selecting this option, as you will not be prompted to save when exiting.

4.1.2.2 Edit Menu

The Edit menu (see Figure 109) controls how the mesh and its properties are edited through direct manipulation in the mesh view.

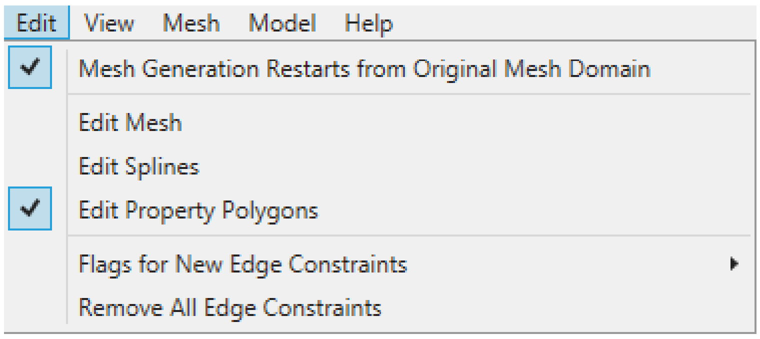

Edit
View
Mesh
Model
Help
✓ Mesh Generation Restarts from Original Mesh Domain
Edit Mesh
Edit Splines
✓ Edit Property Polygons
Flags for New Edge Constraints
Remove All Edge Constraints

Figure 109: The Edit menu.

Mesh Generation Restarts from Original Mesh Domain

When this flag is turned on (default), AlgoMesh will start from the original mesh domain when running the Delaunay refinement or multi-level optimisation mesh generation algorithms. When this is turned off, mesh generation will operate instead incrementally, adding elements to the current generated mesh.

Edit Mesh, Edit Splines and Edit Property Polygons

These are three edit modes that determine whether clicking in the main view (with or without Ctrl, Shift and Alt modifier keys) affects the mesh, splines or polygons. See the Main View section for a detailed description of the controls in each edit mode.

Flags for New Edge Constraints

This sub-menu, shown in Figure 110, allows setting the edge constraint flags that will be applied to any new edge constraints that are created. See the advanced topic section on Edge Constraint Flags and No Refinement Areas for more information.

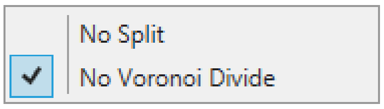


Figure 110: The Edit->Flags for New Edge Constraints menu.

No Split

When active for a given constrained edge, Prohibits AlgoMesh's mesh generator from creating new mesh vertices anywhere along the edge.

No Voronoi Divide

When active, results in whole Voronoi polygons being generated along constrained edges. When inactive, results in the Voronoi polygons being cut by the constrained edges.

Remove All Edge Constraints

Removes all existing edge constraints from the current mesh. Note that this does not prevent new constrained edges from being created.

4.1.2.3 View Menu

The View menu, shown in Figure 111 contains flags controlling the visibility of spatial elements of an AlgoMesh project in the main view.

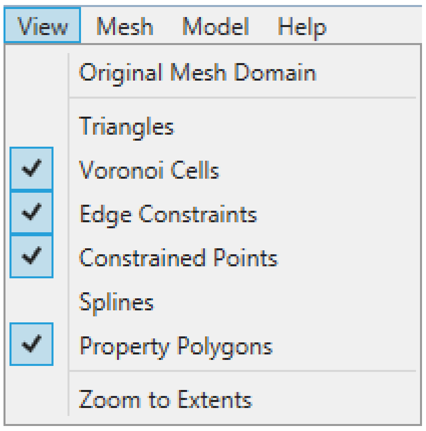


Figure 111: The View menu.

Original Mesh Domain

When turned on, this flag causes the main view to show the basic mesh domain instead of the generated mesh. When turned off (default), the main view shows the current generated mesh.

Triangles

This flag turns on and off the display of the triangular mesh in the main view.

Voronoi Cells

This flag turns on and off the display of the Voronoi mesh in the main view.

Edge Constraints

This flag turns on and off the display of constrained edges in the main view. Thick blue lines represent constrained edges with the "No Voronoi Divide" flag turned on, and thick green lines represent constrained edges with this flag off. A dotted line indicates that the "No Split" flag is turned on for an edge constraint, while a solid line indicates that this flag is turned off.

Constrained Points

This flag turns on and off the display of constrained points (red squares) in the main view.

Splines

This flag turns on and off the display of splines overlying the main view. Splines are shown as thick orange lines, spline generating points are shown as large yellow squares, and the sampled points to be added into the mesh are shown as small white squares.

Property Polygons

This flag turns on and off the display of property polygons overlying the main view. Polygons are shown as translucent purple regions bounded by thick purple lines. Generating points of the spline boundaries of polygons are shown as large purple squares.

Zoom to Extents

Selecting this option causes the main view to move and zoom as necessary to fit the entire mesh within it.

4.1.2.4 Mesh Menu

The Mesh menu, shown in Figure 112, contains a number of advanced utilities that operate on the mesh.

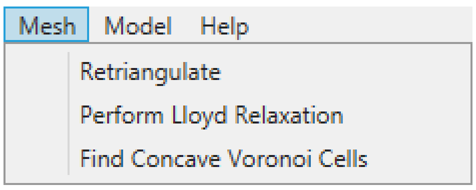


Figure 112: The Mesh menu.

Retriangulate

This option causes AlgoMesh to retriangulate the mesh. This may be useful after removing an edge constraint in Edit Mesh mode, to force a proper Delaunay triangulation to be regenerated.

Perform Lloyd Relaxation

Performs up to 50 iterations of the Lloyd relaxation process normally used within the multi-level optimisation mesh generation method to smooth the mesh according to the current Mesh Cell Edge Length mappings. In general, this is not recommended for use; use the complete optimisation method instead.

Find Concave Voronoi Cells

Searches for cells in the Voronoi mesh whose polygons possess one or more concave angles, and lists any that are found in the log window.

4.1.2.5 Model Menu

The Model menu, shown in Figure 113, provides access to model settings.

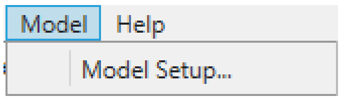


Figure 113: The Model menu.

Model Setup

Opens the Model Setup window to allow you to change settings for the current model.

4.1.2.6 Help Menu

The Help menu, shown in Figure 114, provides access to the AlgoMesh user guide and to more information about the software.

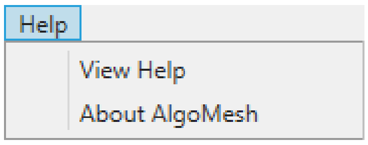


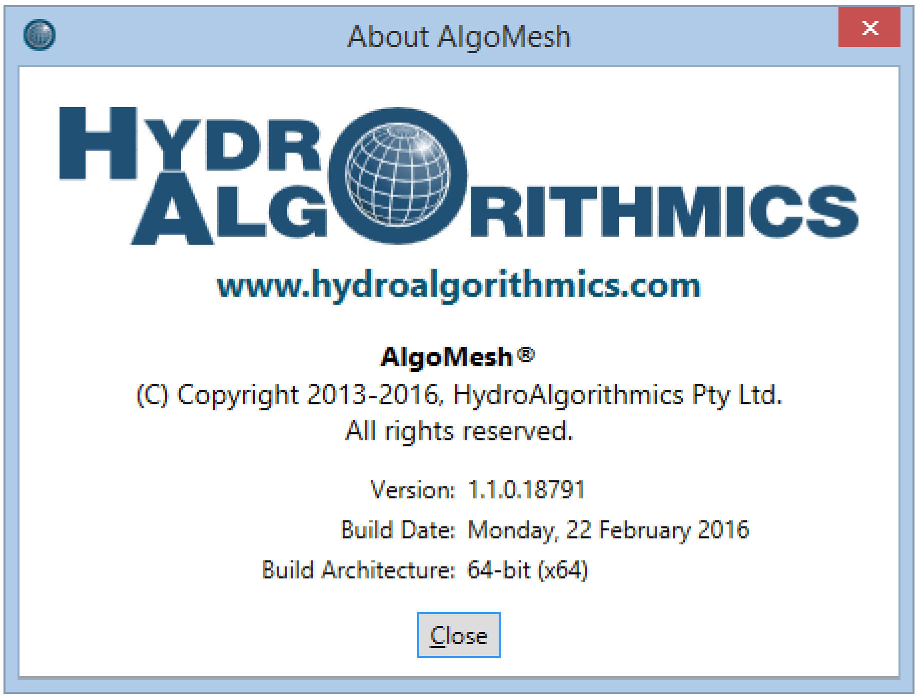
Figure 114: The Help menu.

View Help

Launches the AlgoMesh user guide in HTML Help (.chm) format.

About AlgoMesh

Shows the About AlgoMesh window (Figure 115), detailing the version, date and architecture of the software build that is being used. This information should be provided to HydroAlgorithmics Support with any software support request.



www.hydroalgorithmics.com

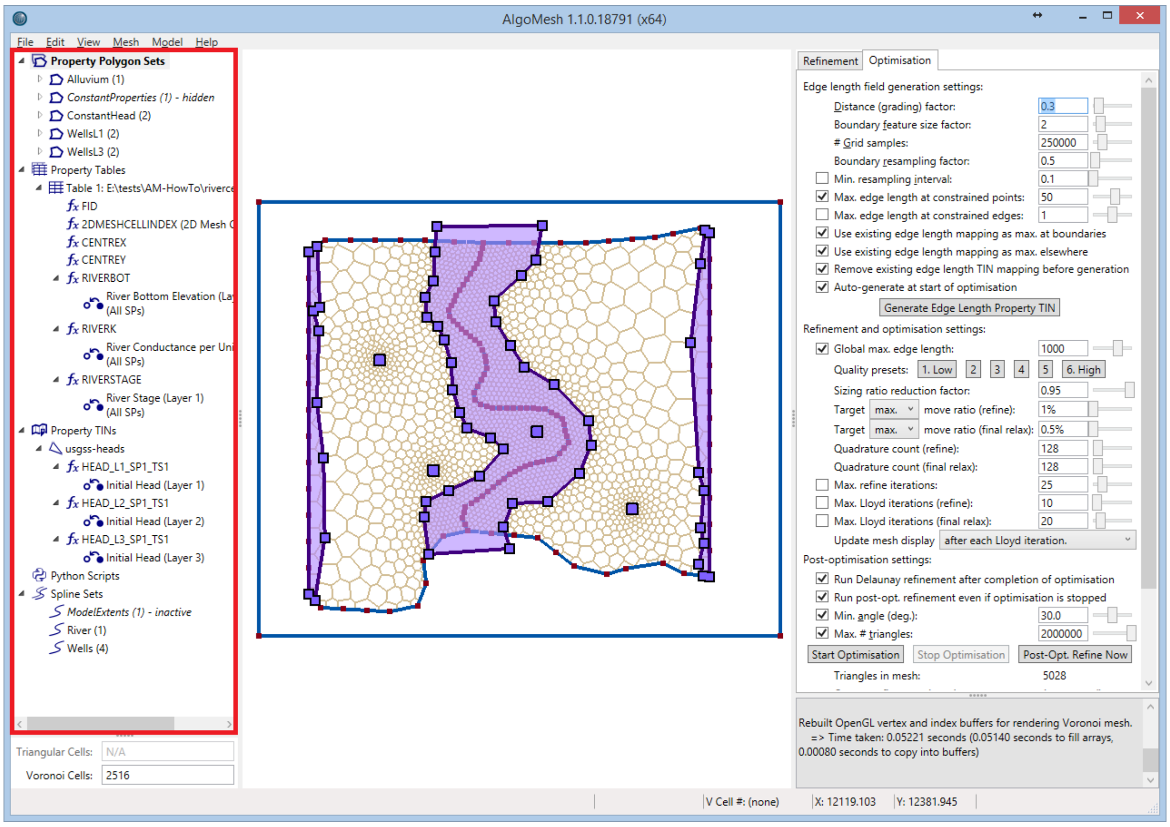
AlgoMesh®
(C) Copyright 2013-2016, HydroAlgorithmics Pty Ltd.
All rights reserved.

Version: 1.1.0.18791
Build Date: Monday, 22 February 2016
Build Architecture: 64-bit (x64)

Figure 115: The About AlgoMesh window.

4.1.3 Project Explorer

The project explorer is the left-most pane of the main AlgoMesh window, outlined in red in Figure 116. It provides access to most of the input geometry and property mapping data in an AlgoMesh project.


AlgoMesh 1.1.0.18791 (x64)
File Edit View Mesh Model Help
Refinement
Optimisation
Generate Edge Length Property TIN
Quality presets: 1. Low 2 3 4 5 6. High
Start Optimisation
Stop Optimisation
Post-Opt. Refine Now

- Property Polygon Sets
  - Alluvium (1)
  - *ConstantProperties (1) - hidden*
  - ConstantHead (2)
  - WellsL1 (2)
  - WellsL3 (2)
- Property Tables
  - Table 1: E:\tests\AM-HowTo\riverce
    - FID
    - 2DMESHCELLINDEX (2D Mesh C
    - CENTREX
    - CENTREY
    - RIVERBOT
      - River Bottom Elevation (La (All SPs)
    - RIVERK
      - River Conductance per Uni (All SPs)
    - RIVERSTAGE
      - River Stage (Layer 1) (All SPs)
- Property TINs
  - usgss-heads
    - HEAD_L1_SP1_TS1
      - Initial Head (Layer 1)
    - HEAD_L2_SP1_TS1
      - Initial Head (Layer 2)
    - HEAD_L3_SP1_TS1
      - Initial Head (Layer 3)
- Python Scripts
- Spline Sets
  - *ModelExtents (1) - inactive*
  - River (1)
  - Wells (4)

Triangular Cells: N/A
Voronoi Cells: 2516

Edge length field generation settings:
Distance (grading) factor: 0.3
Boundary feature size factor: 2
Grid samples: 250000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
✓ Max. edge length at constrained points: 50
☐ Max. edge length at constrained edges: 1
✓ Use existing edge length mapping as max. at boundaries
✓ Use existing edge length mapping as max. elsewhere
✓ Remove existing edge length TIN mapping before generation
✓ Auto-generate at start of optimisation

Refinement and optimisation settings:
✓ Global max. edge length: 1000
Sizing ratio reduction factor: 0.95
Target max. move ratio (refine): 1%
Target max. move ratio (final relax): 0.5%
Quadrature count (refine): 128
Quadrature count (final relax): 128
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:
✓ Run Delaunay refinement after completion of optimisation
✓ Run post-opt. refinement even if optimisation is stopped
✓ Min. angle (deg.): 30.0
✓ Max. # triangles: 2000000

Triangles in mesh: 5028

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
   => Time taken: 0.05221 seconds (0.05140 seconds to fill arrays,
0.00080 seconds to copy into buffers)
```

V Cell #: (none) | X: 12119.103 | Y: 12381.945

Figure 116: The project explorer is the pane on the left-hand side of the main AlgoMesh window, outlined here in red.

The data presented in the project explorer is divided into categories under five headers. See the individual sections for details on each:

- Property Polygon Sets.
- Property Tables.
- Property TINs.
- Python Scripts.
- Spline Sets.

Properties of polygon sets, tables, TINs and Python scripts may be mapped to mesh generation and model variables, as described in the Mesh Variable Mapping section.

4.1.3.1 Property Polygon Sets

The Property Polygon Sets section of the project explorer (Figure 117) lists all property polygon sets in the AlgoMesh project. Each property polygon set is akin to a polygon feature layer in GIS: it contains a collection of polygons and attribute data (properties) associated with them.

Each polygon set is listed underneath the Property Polygon Sets header in the project explorer, with the number of polygons it contains shown in parentheses; for example, in the figure shown, there is a set called Alluvium containing one polygon, a hidden set called ConstantProperties containing one polygon, a set called ConstantHead containing two polygons, and so on. Underneath each polygon set are the properties that may have values assigned for each polygon in the set. In the figure shown, the Alluvium set contains properties CellSize, Id, Kx1, Kz1, Ss1 and Sy1. All property values are converted to real numbers for use in AlgoMesh. Underneath each property are listed any mappings that have been made from that property to AlgoMesh variables; for example, the Ss1 property is mapped here to Specific Storage (Layer 1).

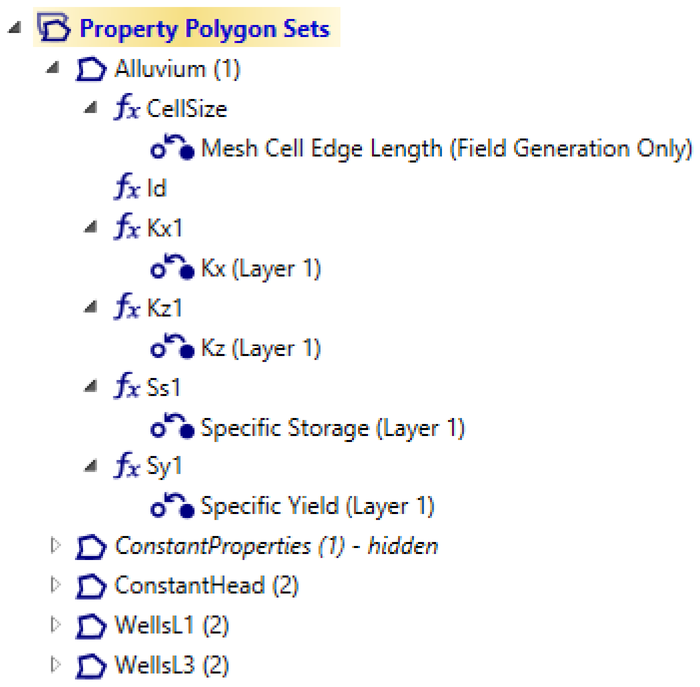

Property Polygon Sets
Alluvium (1)
CellSize
Mesh Cell Edge Length (Field Generation Only)
Id
Kx1
Kx (Layer 1)
Kz1
Kz (Layer 1)
Ss1
Specific Storage (Layer 1)
Sy1
Specific Yield (Layer 1)
ConstantProperties (1) - hidden
ConstantHead (2)
WellsL1 (2)
WellsL3 (2)

Figure 117: The Property Polygon Sets section of the project explorer.

The order of polygon sets in the list is important; when there are multiple mappings for the same property present, AlgoMesh evaluates them from top to bottom through the list, i.e. the top-most property mapping takes precedence.

Property Polygon Sets Header Context Menu

Right-clicking on the Property Polygon Sets header brings up the context menu shown in Figure 118.

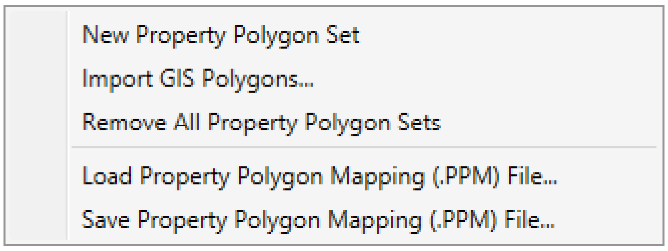

New Property Polygon Set
Import GIS Polygons...
Remove All Property Polygon Sets
Load Property Polygon Mapping (.PPM) File...
Save Property Polygon Mapping (.PPM) File...

Figure 118: The context menu for the Property Polygon Sets header.

New Property Polygon Set

Creates a new, empty, property polygon set.

Import GIS Polygons

Reads polygons and associated attributes from any of a number of common file formats

(.shp, .dxf, .mid/.mif, and others). A property polygon set is created with the name of the imported layer, or if a set with that name already exists, the imported polygons are added to the existing set.

Remove All Property Polygon Sets

Removes all property polygon sets, including all polygons, properties and property mappings they contain, from the AlgoMesh project.

Load Property Polygon Mapping (.PPM) File

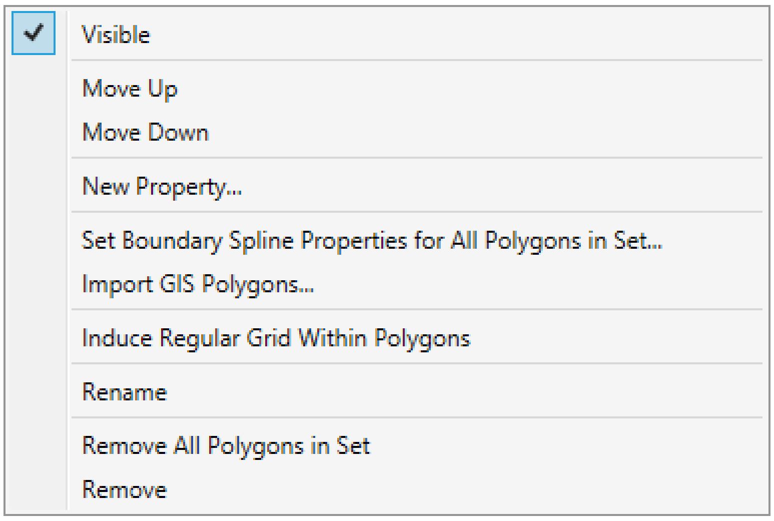
Reads property polygon sets containing polygons and associated attributes from a Property Polygon Mapping (.ppm) file.

Save Property Polygon Mapping (.PPM) File

Saves all property polygon sets, including all polygons, properties and property mappings they contain, to a Property Polygon Mapping (.ppm) file.

Property Polygon Set Context Menu

Right-clicking on a property polygon set entry in the project explorer brings up the context menu shown in Figure 119.


✓ Visible
Move Up
Move Down
New Property...
Set Boundary Spline Properties for All Polygons in Set...
Import GIS Polygons...
Induce Regular Grid Within Polygons
Rename
Remove All Polygons in Set
Remove
Figure 119: The context menu for a property polygon set.

Visible

This flag determines whether the polygons in this polygon set are shown or hidden in the main view when View->Property Polygons is turned on. When View->Property Polygons is turned off, no polygons are shown in the main view, regardless of their set's visible status.

Move Up

Moves this polygon set up one spot in the list of polygon sets. If it is already the top-most set in the list, this does nothing.

Move Down

Moves this polygon set down one spot in the list of polygon sets. If it is already the bottom-most set in the list, this does nothing.

New Property

Creates a new property for this polygon set, for which values for each polygon in the set may be defined. A window is brought up allowing the new property to be named and given a default value.

A polygon's property values may be inspected by right-clicking on it in the main view with Edit->Edit Property Polygons mode enabled.

Set Boundary Spline Properties for All Polygons in Set

Brings up the Apply Spline Properties to Property Polygon Set window, allowing spline properties to be changed for the boundaries of all polygons in the set simultaneously.

Import GIS Polygons

Reads polygons and associated attributes from any of a number of common file formats (.shp, .dxf, .mid/.mif, and others). The imported polygons and any new properties are all added to this polygon set.

Induce Regular Grid Within Polygons

Brings up the Generate Splines to Induce Regular Grid within Polygons window allowing structured sub-grids to be generated within the polygons in this polygon set. See the advanced topic on Structured Sub-Grid Generation for more information.

Rename

Allows the name of the polygon set to be edited.

Remove All Polygons in Set

Removes all polygons contained in this polygon set, without removing the polygon set itself. The list of properties for the polygon set and any mappings are not removed.

Remove

Removes this polygon set from the AlgoMesh project, including any polygons, properties and property mappings it contains.

Polygon Property Context Menu

Right-clicking on a polygon set property entry in the project explorer brings up the context menu shown in Figure 120.

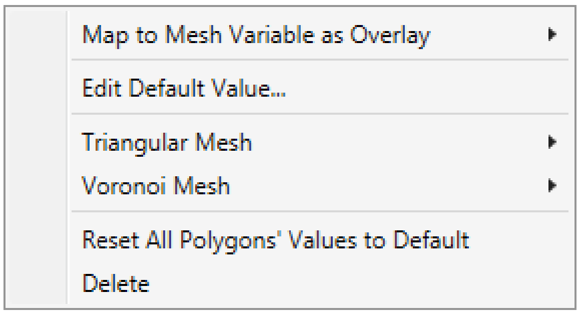

Map to Mesh Variable as Overlay
Edit Default Value...
Triangular Mesh
Voronoi Mesh
Reset All Polygons' Values to Default
Delete

Figure 120: The context menu for a polygon property.

Map to Mesh Variable as Overlay

Allows mapping this property to mesh generation or model variables. See the Mesh Variable Mapping section for more details.

Edit Default Value

Allows the default value for this property to be edited. The default value is given to any polygons that do not have their own value explicitly specified.

Triangular Mesh and Voronoi Mesh

The Triangular Mesh and Voronoi Mesh options bring up a sub-menu allowing export of the property values to CSV at cell centres or vertices. This export utilises either the triangular elements in the triangular mesh or the polygonal Voronoi cells in the Voronoi mesh, depending on whether the Triangular Mesh or Voronoi Mesh sub-menu is used.

Reset All Polygons' Values to Default

Removes explicit specifications of values for this property from all polygons in the set, so that they all take the property's default value.

Delete

Removes this property and values of the property specified for individual polygons from the polygon set.

Property Mapping Context Menu

Right-clicking on a property mapping in the project explorer brings up the context menu shown in Figure 121.

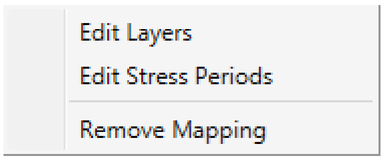


Figure 121: The context menu for a property mapping.

Edit Layers

Makes the list of layers for this property mapping editable. A collection of layers may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive layer ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The layer list may be left empty or the word **none** specified to signify no layers. The word **all** or an asterisk * may be used to signify all layers in the model.

This is disabled for mappings to mesh generation-only properties such as Mesh Cell Edge Length.

Edit Stress Periods

Makes the list of stress periods for this property mapping editable. A collection of stress periods may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive stress period ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The stress period list may be left empty or the word **none** specified to signify no stress periods. The word **all** or an asterisk * may be used to signify all stress periods in the simulation.

This is disabled for mappings to non-transient properties.

Remove Mapping

Unmaps the property from this mesh variable, removing the mapping entry.

4.1.3.2 Property Tables

The Property Tables section of the project explorer (Figure 122) lists all property tables in the AlgoMesh project. Each property table is a link to a comma-separated-values (CSV) file containing a table of data entries, each column of which may be mapped to an AlgoMesh variable.

Each table is listed underneath the Property Tables header in the project explorer, alongside the file path to its linked CSV file; for example, in the figure shown, there is a table called Table 1 that is loaded from the CSV file E:\tests\AM-HowTo\rivercells.csv. Underneath each table is the list of columns (properties) in the CSV table. Underneath each property are listed any mappings that have been made from that property to AlgoMesh variables; for example, the RIVERBOT property is mapped here to River Bottom Elevation (Layer 1) (All SPs).

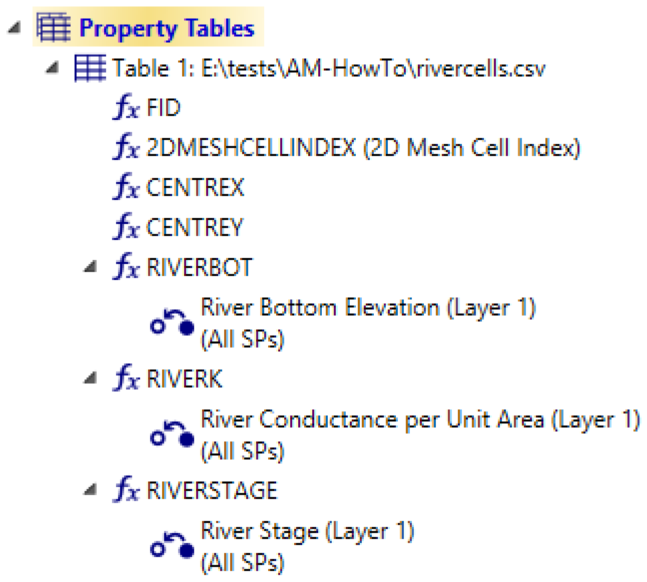

Property Tables
Table 1: E:\tests\AM-HowTo\rivercells.csv
FID
2DMESHCELLINDEX (2D Mesh Cell Index)
CENTREX
CENTREY
RIVERBOT
River Bottom Elevation (Layer 1) (All SPs)
RIVERK
River Conductance per Unit Area (Layer 1) (All SPs)
RIVERSTAGE
River Stage (Layer 1) (All SPs)

Figure 122: The Property Tables section of the project explorer.

Each table must have a column that identifies the model cell to which each data row pertains; this is configured in the CSV Property Table Settings window. If a 2D mesh cell index column is specified alone for this purpose, layers and stress periods must be assigned to any property mappings made as they would be with polygon or TIN property mappings. However, it is also possible to specify a layer column or to use a global cell index column instead of a 2D mesh cell index column; in either of these cases, the layer is determined by the column's value for each row and is not set on the mapping itself. Similarly, a stress period column may be specified, in which case the stress period is determined by that column's value for each row, and is not set on the mapping itself.

Property mappings made to table columns do not overlay other mappings like polygon property mappings, and only one such mapping per property may exist. In the same way, these mappings are mutually exclusive with property TIN mappings; a given property may be mapped to a TIN property or a table property, but not both - unless each is specified for an entirely different combination of layers and stress periods.

Property Tables Header Context Menu

Right-clicking on the Property Tables header brings up the context menu shown in Figure 123.

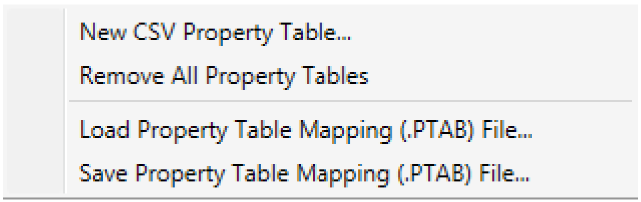

New CSV Property Table...
Remove All Property Tables
Load Property Table Mapping (.PTAB) File...
Save Property Table Mapping (.PTAB) File...

Figure 123: The context menu for the Property Tables header.

New CSV Property Table

Loads a selected CSV file from disk, and opens the CSV Property Table Settings window to configure how its data is to be interpreted by AlgoMesh. When the settings are accepted, a new property table is created linked to the specified CSV file.

Remove All Property Tables

Removes all property tables, including all of their settings and property mappings, from the AlgoMesh project.

Load Property Table Mapping (.PTAB) File

Loads all property tables, their settings and associated mappings from a Property Table Mapping (.PTAB) file.

Save Property Table Mapping (.PTAB) File

Reads property tables, their settings and associated mappings to a Property Table Mapping (.PTAB) file.

Property Table Context Menu

Right-clicking on a property table entry in the project explorer brings up the context menu shown in Figure 124.

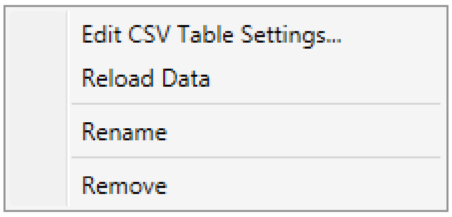


Figure 124: The context menu for a CSV property table.

Edit CSV Table Settings

Brings up the CSV Property Table Settings window for this property table.

Reload Data

Forces AlgoMesh to reload the linked CSV file now, and updates the list of columns accordingly.

Rename

Allows the name of this property table to be edited.

Remove

Removes this property table from the AlgoMesh project, including its settings and associated mappings.

Table Property Context Menu

Right-clicking on a table property entry in the project explorer brings up the context menu shown in Figure 125.

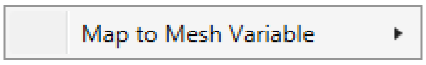


Figure 125: The context menu for a CSV property table field.

Map to Mesh Variable

Allows mapping this property to mesh generation or model variables. See the Mesh Variable Mapping section for more details.

Property Mapping Context Menu

Right-clicking on a property mapping in the project explorer brings up the context menu shown in Figure 126.

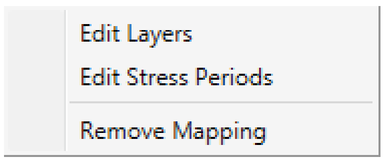


Figure 126: The context menu for a property mapping.

Edit Layers

Makes the list of layers for this property mapping editable. A collection of layers may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive layer ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The layer list may be left empty or the word **none** specified to signify no layers. The word **all** or an asterisk * may be used to signify all layers in the model.

This is disabled for mappings to mesh generation-only properties such as Mesh Cell Edge Length, and for all mappings on a table that has a Layer column specified.

Edit Stress Periods

Makes the list of stress periods for this property mapping editable. A collection of stress periods may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive stress period ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The stress period list may be left empty or the word **none** specified to signify no stress periods. The word **all** or an asterisk * may be used to signify all stress periods in the simulation.

This is disabled for mappings to non-transient properties, and for all mappings on a table that has a Stress Period column specified.

Remove Mapping

Unmaps the property from this mesh variable, removing the mapping entry.

4.1.3.3 Property TINs

The Property TINs section of the project explorer (Figure 127) lists all property TINs (*Triangulated Irregular Networks*) in the AlgoMesh project. Each property TIN is a triangulated surface of sample points (e.g. grid points), each of which may be associated with a value of one or more properties to be interpolated over the surface.

Each TIN is listed by name underneath the Property TINs header in the project explorer, and beneath it, each of its properties are listed and available for mapping to AlgoMesh variables. For example, in the figure shown, the HEAD_L1_SP1_TS1 property of the TIN usgss-heads is mapped to Initial Head (Layer 1).

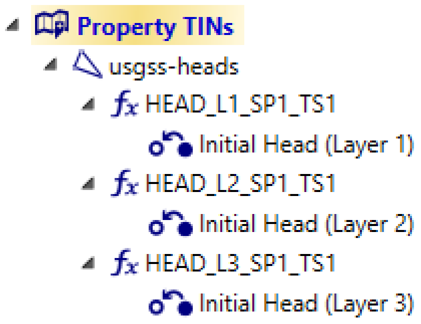


Figure 127: The Property TINs section of the project explorer.

Property TINs are useful for representing spatially-varying data, and may be created from point shapefiles, textual XYZ files, or from AlgoMesh Mesh Domain Description (.mdd) files.

Property TINs Header Context Menu

Right-clicking on the Property TINs header brings up the context menu shown in Figure 128.

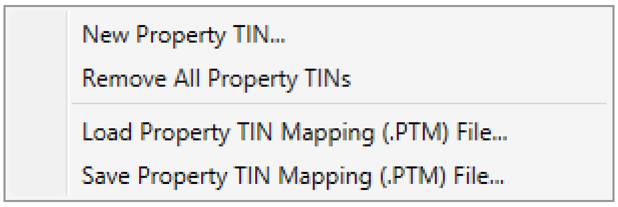

New Property TIN...
Remove All Property TINs
Load Property TIN Mapping (.PTM) File...
Save Property TIN Mapping (.PTM) File...

Figure 128: The context menu for the Property TINs header.

New Property TIN

Reads points, constrained line segments and any point attributes from a supported format (.shp, .xyz, .mdd, and others) and creates a new property TIN from them.

Remove All Property TINs

Removes all property TINs and all associated property mappings from the AlgoMesh project.

Load Property TIN Mapping (.PTM) File

Loads all property TINs and associated mappings from a Property TIN Mapping (.ptm) file. Note that .ptm files link to external .mdd files, and these .mdd files must also be present in the right locations so that AlgoMesh can read them in.

Save Property TIN Mapping (.PTM) File

Saves all property TINs and associated mappings to a Property TIN Mapping (.ptm) file. Any property TINs that are not present as .mdd files will be saved to .mdd format prior to writing the .ptm file.

Property TIN Context Menu

Right-clicking on a property TIN entry in the project explorer brings up the context menu shown in Figure 129.

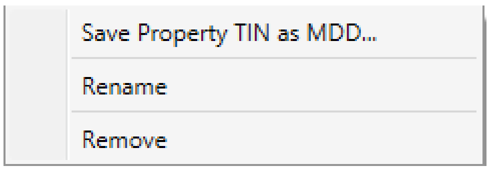


Figure 129: The context menu for a property TIN.

Save Property TIN as MDD

Exports this property TIN to a Mesh Domain Description (.mdd) file on disk.

Rename

Allows the name of the property TIN to be edited.

Remove

Removes the property TIN and any associated mappings from the AlgoMesh project.

TIN Property Context Menu

Right-clicking on a TIN property entry in the project explorer brings up the context menu shown in Figure 130.

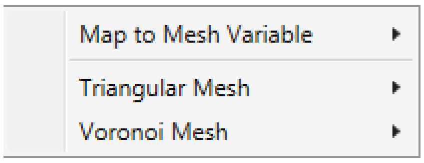


Figure 130: The context menu for a TIN property.

Map to Mesh Variable

Allows mapping this property to mesh generation or model variables. See the Mesh Variable Mapping section for more details.

Triangular Mesh and Voronoi Mesh

The Triangular Mesh and Voronoi Mesh options bring up a sub-menu allowing export of the property values to CSV at cell centres or vertices. This export utilises either the triangular elements in the triangular mesh or the polygonal Voronoi cells in the Voronoi mesh, depending on whether the Triangular Mesh or Voronoi Mesh sub-menu is used.

Property Mapping Context Menu

Right-clicking on a property mapping in the project explorer brings up the context menu shown in Figure 131.

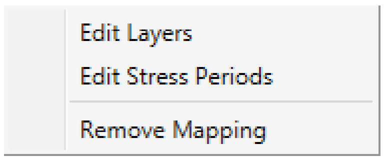


Figure 131: The context menu for a property mapping.

Edit Layers

Makes the list of layers for this property mapping editable. A collection of layers may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive layer ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The layer list may be left empty or the word **none** specified to signify no layers. The word **all** or an asterisk * may be used to signify all layers in the model.

This is disabled for mappings to mesh generation-only properties such as Mesh Cell Edge Length.

Edit Stress Periods

Makes the list of stress periods for this property mapping editable. A collection of stress periods may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive stress period ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The stress period list may be left empty or the word **none** specified to signify no stress periods. The word **all** or an asterisk * may be used to signify all stress periods in the simulation.

This is disabled for mappings to non-transient properties.

Remove Mapping

Unmaps the property from this mesh variable, removing the mapping entry.

4.1.3.4 Python Scripts

The Python Scripts section of the project explorer (Figure 132) lists all embedded Python scripts in the AlgoMesh project.

Note that a Python 3.4 installation must be present on your system for Python script support; otherwise, the Python Scripts header will be disabled.

See the advanced topic on Python Scripts for more details.

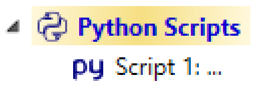


Figure 132: The Python Scripts section of the project explorer.

Python Scripts Header Context Menu

Right-clicking on the Python Scripts header brings up the context menu shown in Figure 133.

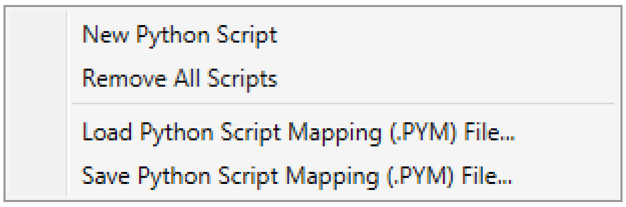

New Python Script
Remove All Scripts
Load Python Script Mapping (.PYM) File...
Save Python Script Mapping (.PYM) File...

Figure 133: The context menu for the Python Scripts header.

New Python Script

Creates a new, empty, Python script.

Remove All Scripts

Removes all Python scripts and any associated mappings from the AlgoMesh project.

Load Python Script Mapping (.PYM) File

Loads all Python scripts and associated mappings from a Python Script Mapping (.pym) file.

Save Python Script Mapping (.PYM) File

Saves all Python scripts and associated mappings to a Python Script Mapping (.pym) file. All script code is embedded into the .pym file.

Python Script Context Menu

Right-clicking on a Python script entry in the project explorer brings up the context menu shown in Figure 134.

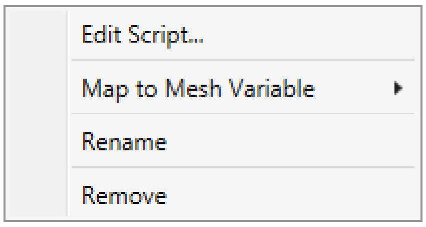


Figure 134: The context menu for a Python script.

Edit Script

Brings up the Python Script editor for this script.

Map to Mesh Variable

Allows mapping this script to mesh generation or model variables. See the Mesh Variable Mapping section for more details.

Rename

Allows the name of this Python script to be edited.

Remove

Removes this Python script and any associated mappings from the AlgoMesh project.

Property Mapping Context Menu

Right-clicking on a property mapping in the project explorer brings up the context menu shown in Figure 135.

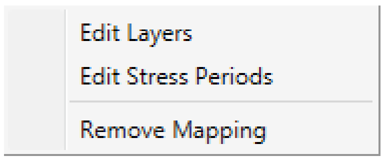


Figure 135: The context menu for a property mapping.

Edit Layers

Makes the list of layers for this property mapping editable. A collection of layers may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive layer ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The layer list may be left empty or the word **none** specified to signify no layers. The word **all** or an asterisk * may be used to signify all layers in the model.

This is disabled for mappings to mesh generation-only properties such as Mesh Cell Edge Length.

Edit Stress Periods

Makes the list of stress periods for this property mapping editable. A collection of stress periods may be specified, with values separated by commas, e.g. **1, 2, 4, 6**. Inclusive stress period ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The stress period list may be left empty or the word **none** specified to signify no stress periods. The word **all** or an asterisk * may be used to signify all stress periods in the simulation.

This is disabled for mappings to non-transient properties.

Remove Mapping

Unmaps the property from this mesh variable, removing the mapping entry.

4.1.3.5 Spline Sets

The Spline Sets section of the project explorer (Figure 136) lists all spline sets in the AlgoMesh project. Each spline set contains a collection of polylines and points, with the ability to represent smoothly curving features as well as linear polylines. All active spline sets have their contents sampled and added to the mesh when mesh generation takes place. No properties are available for mapping within spline sets - they are purely used to insert geometry into the mesh.

Each spline set is listed underneath the Spline Sets header in the project explorer, with the number of features it contains shown in parentheses; for example, in the figure shown, there is an inactive set called ModelExtents containing one spline, a set called River containing one spline, and a set called Wells containing 4 points. Note that when spline sets are used to contain point features, each point is considered a "spline" in its own right, even though it will generate only a single point and no line segments in the mesh. Inactive spline sets are not shown in the main view and are not added to the mesh when mesh generation is initiated.

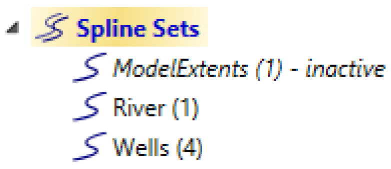


Figure 136: The Spline Sets section of the project explorer.

Spline Sets Header Context Menu

Right-clicking on the Spline Sets header brings up the context menu shown in Figure 137.

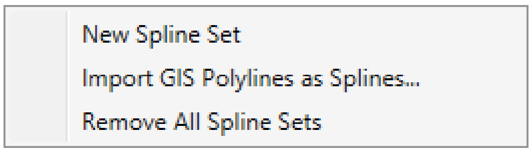


Figure 137: The context menu for the Spline Sets header.

New Spline Set

Creates a new, empty, spline set.

Import GIS Polylines as Splines

Reads points, polylines or polygons and associated attributes from any of a number of common file formats (.shp, .dxf, .mid/.mif, and others). A spline set is created with the name of the imported layer, or if a set with that name already exists, the imported features are added to the existing set.

Remove All Spline Sets

Removes all spline sets and the features they contain from the AlgoMesh project.

Spline Set Context Menu

Right-clicking on a spline set entry in the project explorer brings up the context menu shown in Figure 138.

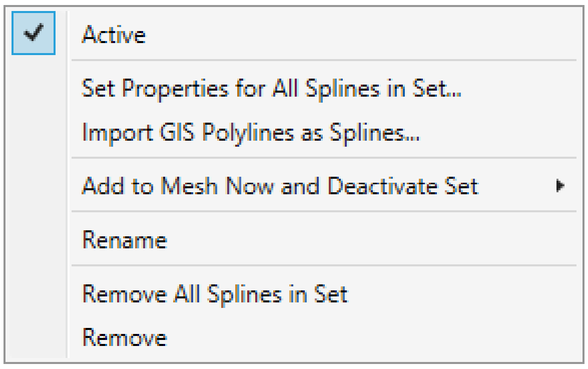

✓ Active
Set Properties for All Splines in Set...
Import GIS Polylines as Splines...
Add to Mesh Now and Deactivate Set
Rename
Remove All Splines in Set
Remove

Figure 138: The context menu for a spline set.

Active

This flag determines whether the features in this spline set are added to the mesh domain when mesh generation is initiated (active, default) or not (inactive). Features contained within inactive spline sets are never displayed in the main view, regardless of whether the View->Splines flag is turned on or off.

Set Properties for All Splines in Set

Brings up the Apply Properties to Spline Set window, allowing spline properties to be changed for all features in the set simultaneously.

Import GIS Polylines as Splines

Reads points, polylines or polygons and associated attributes from any of a number of common file formats (.shp, .dxf, .mid/.mif, and others). The imported features are all added to this spline set.

Add to Mesh Now and Deactivate Set

Immediately adds the features in this spline set into the mesh and makes the spline set inactive. A sub-menu allows the choice of whether the features are added to both the original mesh domain and the current generated mesh (recommended) or only the current generated mesh, the latter leaving the original mesh domain untouched.

This feature is useful for setting complex extent boundaries in a model and subsequently setting areas external to these as outside the mesh, prior to mesh generation. See the Import Geometry from GIS how-to section for a step-by-step introduction to this process.

Rename

Allows the name of this spline set to be edited.

Remove All Splines in Set

Removes all features in the spline set, without removing the spline set itself.

Remove

Removes this spline set from the AlgoMesh project.

4.1.3.6 Mesh Variable Mapping

Properties contained in Property Polygon Sets, Property Tables and Property TINs, in addition to Python Scripts, may all be mapped to AlgoMesh *mesh variables* by using the context menu options Map to Mesh Variable as Overlay (for polygon properties) or Map to Mesh Variable (for everything else). A mesh variable in AlgoMesh is something that provides a needed value either at a location in the mesh domain for the mesh generation process, or at a cell centre when building MODFLOW-USG model files.

The list of available mesh variables may be seen in the mesh variable mapping menu shown in Figure 139. Model property variables are continuous over the domain, whereas model boundary condition variables should only be mapped in specific locations where the corresponding boundary condition should be assigned to model cells. Note that EVT and RCH package boundary conditions, if mapped at all, may only be mapped to layer 1, and must be mapped across the entire mesh domain.

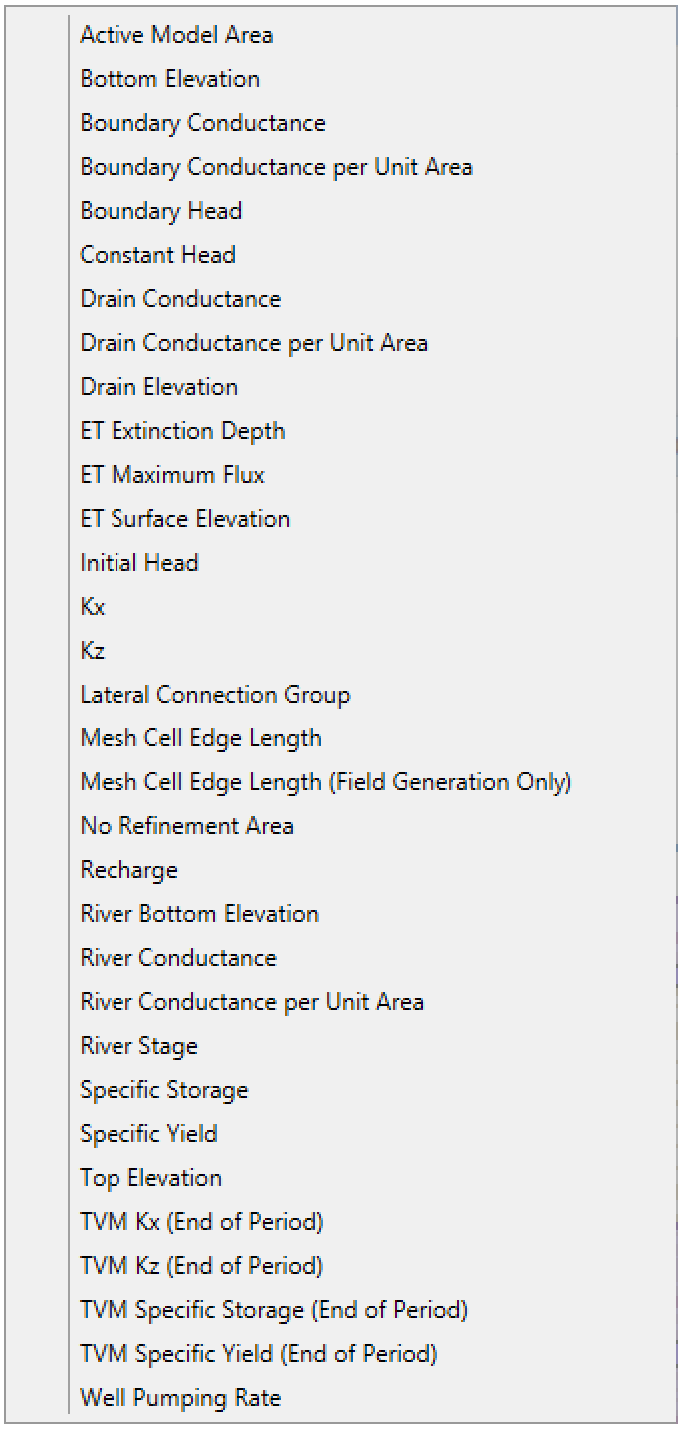


Figure 139: The mesh variable mapping menu.

Active Model Area *(Model Property Variable)*

Defines which cells should be included in a model (value 1) and which should be removed entirely (value 0). The default value for this property is 1, i.e. all cells are included by default.

Bottom Elevation *(Model Property Variable)*

The elevation (L) at the bottom of each cell.

Boundary Conductance *(Model Boundary Condition Variable)*

Conductance (L^2 / T) for General Head Boundary (GHB) cells.

Boundary Conductance per Unit Area *(Model Boundary Condition Variable)*

A factor to be multiplied by cell area by AlgoMesh to produce a conductance for General Head Boundary (GHB) cells.

Boundary Head *(Model Boundary Condition Variable)*

Specified head (L) for General Head Boundary (GHB) cells.

Constant Head *(Model Boundary Condition Variable)*

Constant head (L) for Constant Head (CHD) cells.

Drain Conductance *(Model Boundary Condition Variable)*

Conductance (L^2 / T) for Drain (DRN) cells.

Drain Conductance per Unit Area *(Model Boundary Condition Variable)*

A factor to be multiplied by cell area by AlgoMesh to produce a conductance for Drain (DRN) cells.

Drain Elevation *(Model Boundary Condition Variable)*

Elevation (L) for Drain (DRN) cells.

ET Extinction Depth *(Model Boundary Condition Variable)*

Extinction depth (L) for the Evapotranspiration (EVT) package. If any ET properties are mapped, they must all be mapped over the entirety of layer 1.

ET Maximum Flux *(Model Boundary Condition Variable)*

Maximum volumetric flow rate per unit area (L / T) for the Evapotranspiration (EVT) package. If any ET properties are mapped, they must all be mapped over the entirety of layer 1.

ET Surface Elevation *(Model Boundary Condition Variable)*

Surface elevation (L) for the Evapotranspiration (EVT) package. If any ET properties are mapped, they must all be mapped over the entirety of layer 1.

Initial Head *(Model Property Variable)*

Head (L) to be assigned at the start of the simulation, specified in the Basic (BAS6) package.

Kx *(Model Property Variable)*

Lateral hydraulic conductivity (L / T), specified in the Layer-Property Flow (LPF) package. Note that

AlgoMesh does not support anisotropy, so Ky = Kx.

Kz *(Model Property Variable)*

Vertical hydraulic conductivity (L / T), specified in the Layer-Property Flow (LPF) package.

Lateral Connection Group *(Model Property Variable)*

A value used to determine groups of cells to connect laterally by different means. Cells with the same LCG value are connected laterally only if they are in the same layer. Cells of different LCG values are normally not connected laterally, but this may be controlled in the Model Setup window. The default LCG value is 0.

See the advanced topic on Lateral Connection Groups for more information.

Mesh Cell Edge Length *(Mesh Generation Variable)*

The desired maximum triangle edge length (L) over the mesh domain. This property is used by the edge length field generation process, the Delaunay refinement process and the multi-level optimisation process to influence the cell size distribution in the mesh. It may be used to override an edge length field property TIN during mesh generation.

Mesh Cell Edge Length (Field Generation Only) *(Mesh Generation Variable)*

The desired maximum triangle edge length (L) over the mesh domain. This property is only used by the edge length field generation process to provide maximum limits on cell sizing in the edge length field, and thus will not override an edge length field property TIN during mesh generation.

No Refinement Area *(Mesh Generation Variable)*

Defines regions in the mesh domain within which triangles should not be split to create new triangles during mesh generation. A value of 1 indicates a no refinement area. The default value for this property is 0, i.e. refinement is permitted in all areas of the model.

See the advanced topic on Edge Constraint Flags and No Refinement Areas for more information.

Recharge *(Model Boundary Condition Variable)*

Vertical recharge volumetric flux per unit area (L / T) for the Recharge (RCH) package.

River Bottom Elevation *(Model Boundary Condition Variable)*

Elevation (L) of the bottom of the river bed for River (RIV) cells.

River Conductance *(Model Boundary Condition Variable)*

Conductance (L^2 / T) for River (RIV) cells.

River Conductance per Unit Area *(Model Boundary Condition Variable)*

A factor to be multiplied by cell area by AlgoMesh to produce a conductance for River (RIV) cells.

River Stage *(Model Boundary Condition Variable)*

Stage (L) for River (RIV) cells.

Specific Storage *(Model Property Variable)*

Specific storage (1 / L) for transient simulations.

Specific Yield *(Model Property Variable)*

Specific yield (dimensionless) for transient simulations.

Top Elevation *(Model Property Variable)*

The elevation (L) at the top of each cell.

TVM Kx (End of Period) *(Model Boundary Condition Variable)*

Modified lateral hydraulic conductivity (L / T) to be applied during a transient simulation. Note that this requires a special version of MODFLOW-USG with the TVM package built in.

TVM Kz (End of Period) *(Model Boundary Condition Variable)*

Modified vertical hydraulic conductivity (L / T) to be applied during a transient simulation. Note that this requires a special version of MODFLOW-USG with the TVM package built in.

TVM Specific Storage (End of Period) *(Model Boundary Condition Variable)*

Modified specific storage (1 / L) to be applied during a transient simulation. Note that this requires a special version of MODFLOW-USG with the TVM package built in.

TVM Specific Yield (End of Period) *(Model Boundary Condition Variable)*

Modified specific storage (dimensionless) to be applied during a transient simulation. Note that this requires a special version of MODFLOW-USG with the TVM package built in.

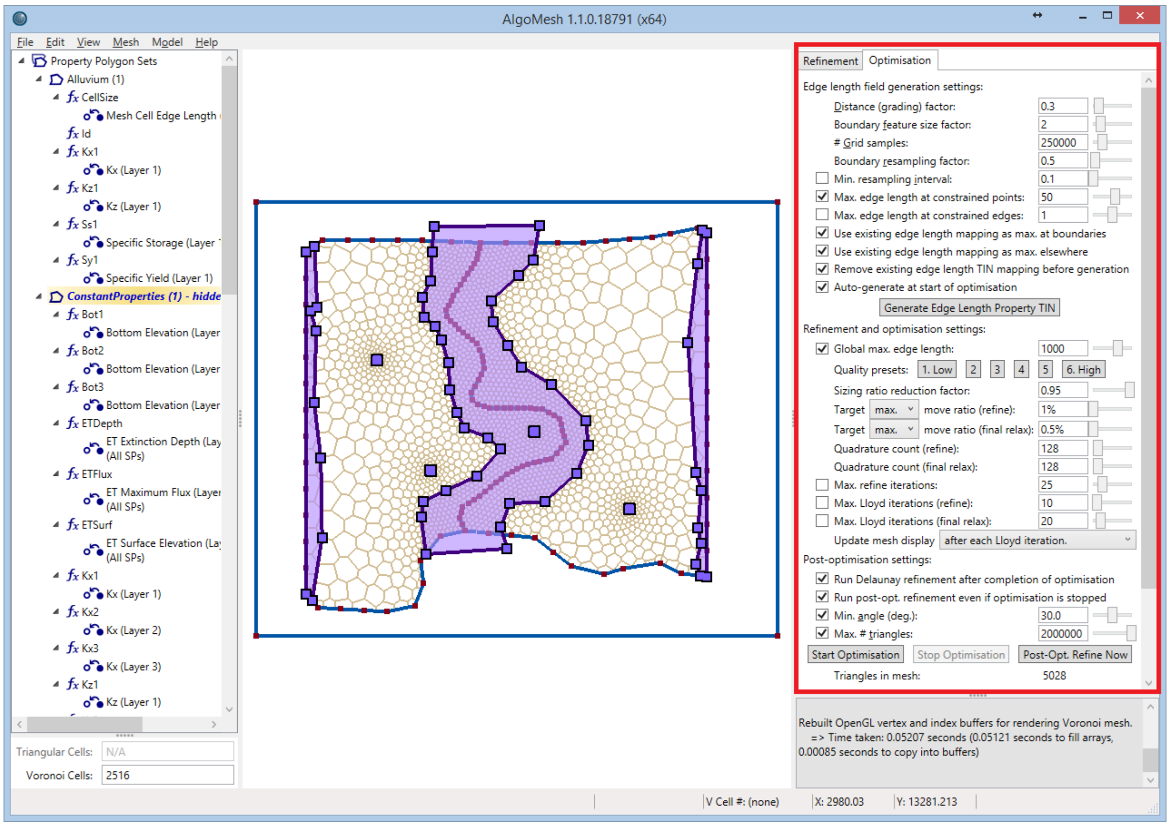
Well Pumping Rate *(Model Boundary Condition Variable)*

Volumetric flow rate (L^3 / T) for Well (WEL) cells. Positive values indicates recharge, and negative values indicate discharge (pumping).

4.1.4 Mesh Generation Panels

The mesh generation panels appear on the right-hand side of the main AlgoMesh window, as outlined in red in Figure 140.

Two panels are available: one for the Delaunay refinement method (the *Refinement* tab), and the other for the multi-level optimisation method (the *Optimisation* tab). Each panel allows configuration of the relevant settings for the corresponding method, and provides buttons to initiate and control mesh generation.



Triangular Cells: N/A
Voronoi Cells: 2516

Edge length field generation settings:

- Distance (grading) factor: 0.3
- Boundary feature size factor: 2
- # Grid samples: 250000
- Boundary resampling factor: 0.5
- ☐ Min. resampling interval: 0.1
- ✓ Max. edge length at constrained points: 50
- ☐ Max. edge length at constrained edges: 1
- ✓ Use existing edge length mapping as max. at boundaries
- ✓ Use existing edge length mapping as max. elsewhere
- ✓ Remove existing edge length TIN mapping before generation
- ✓ Auto-generate at start of optimisation

Generate Edge Length Property TIN

Refinement and optimisation settings:

- ✓ Global max. edge length: 1000
- Quality presets: 1. Low | 2 | 3 | 4 | 5 | 6. High
- Sizing ratio reduction factor: 0.95
- Target max. move ratio (refine): 1%
- Target max. move ratio (final relax): 0.5%
- Quadrature count (refine): 128
- Quadrature count (final relax): 128
- ☐ Max. refine iterations: 25
- ☐ Max. Lloyd iterations (refine): 10
- ☐ Max. Lloyd iterations (final relax): 20
- Update mesh display after each Lloyd iteration.

Post-optimisation settings:

- ✓ Run Delaunay refinement after completion of optimisation
- ✓ Run post-opt. refinement even if optimisation is stopped
- ✓ Min. angle (deg.): 30.0
- ✓ Max. # triangles: 2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 5028

```
Rebuilt OpenGL vertex and index buffers for rendering Voronoi mesh.
  => Time taken: 0.05207 seconds (0.05121 seconds to fill arrays,
0.00085 seconds to copy into buffers)
```

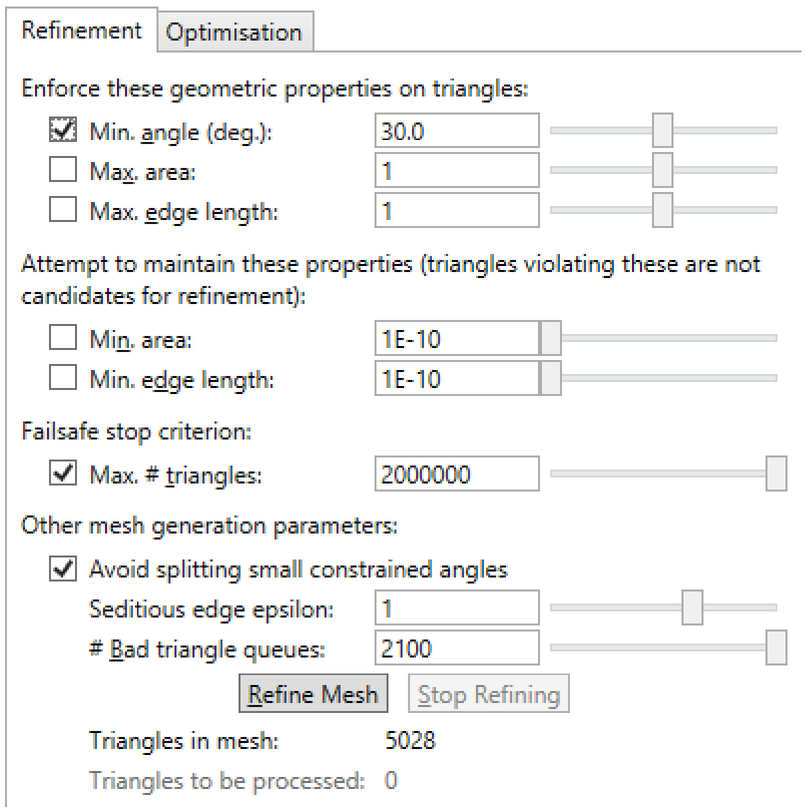
V Cell #: (none) | X: 2980.03 | Y: 13281.213

Figure 140: The mesh generation panels for initiating the Delaunay refinement and multi-level optimisation methods are in the right-hand pane of the main AlgoMesh window, outlined here in red.

4.1.4.1 Refinement

The Refinement panel, shown in Figure 141, provides access to the Delaunay refinement mesh generation algorithm and its settings.

See the how-to guides Create a Simple Triangular Mesh Using Delaunay Refinement and Create a Simple Voronoi Mesh Using Delaunay Refinement for an introduction to using the Delaunay refinement algorithm.



Enforce these geometric properties on triangles:

✓ Min. angle (deg.):	30.0
Max. area:	1
Max. edge length:	1

Attempt to maintain these properties (triangles violating these are not candidates for refinement):

Min. area:	1E-10
Min. edge length:	1E-10

Failsafe stop criterion:

✓ Max. # triangles:	2000000

Other mesh generation parameters:

✓ Avoid splitting small constrained angles	
Seditious edge epsilon:	1
# Bad triangle queues:	2100
Triangles in mesh:	5028
Triangles to be processed:	0

Figure 141: The Refinement tab, used for controlling the Delaunay refinement mesh generation process.

Settings

Min. Angle (Deg.)

Applies a constraint on the minimum angle between two edges at any vertex of a triangle in the triangular mesh. Typical values are between 20 and 34 degrees.

Max. Area

Applies a constraint on the maximum area of any triangle in the triangular mesh. This has the effect of globally limiting cell size over the generated mesh.

Max. Edge Length

Applies a constraint on the maximum length of any triangle edge in the triangular mesh. This has the effect of globally limiting cell size over the generated mesh (node to node distance for triangular meshes, or centre-to-centre distance for Voronoi meshes).

Min. Area

When active, stops AlgoMesh from considering any triangle with less than the given area as a candidate for refinement. This setting is not recommended for general use, as it may reduce the quality of the generated mesh.

Min. Edge Length

When active, stops AlgoMesh from considering any triangle whose longest edge is shorter than the given length as a candidate for refinement. This setting is not recommended for general use, as it may reduce the quality of the generated mesh.

Max. # Triangles

Stops the mesh generation process early when the triangular mesh contains given number of triangles. If this option is disabled, mesh generation will continue until there are no more "bad" triangles to be split. It is recommended that this option is left turned on, particularly if a high minimum angle value is selected, to stop AlgoMesh from continuing to refine the mesh indefinitely.

Avoid Splitting Small Constrained Angles

When active, stops AlgoMesh from creating triangles between two constrained edges where the angle is lower than the minimum angle constraint. It is recommended that this option is left turned on.

Seditious Edge Epsilon

The maximum difference in length between two constrained edges meeting at an angle of less than 60 degrees for AlgoMesh to consider an edge *seditious*. Seditious edges may lead to infinite refinement. Upon detecting a seditious edge, AlgoMesh stops splitting the edge further to allow mesh generation to complete. The default value of 1 should be sufficient for practical applications.

# Bad Triangle Queues

The number of discrete queues used to form an approximate ordering of bad triangles in the Delaunay refinement process. The default value of 2100 should be sufficient for practical applications.

Execution Control Buttons

Refine Mesh

Starts execution of the Delaunay refinement process.

Stop Refining

Stops execution of the Delaunay refinement process early. This is only enabled while the refinement algorithm is executing. Note that the Delaunay refinement process will normally stop executing by itself, without needing to click this button.

Progress Information

Triangles in Mesh

Displays the current number of triangles in the underlying triangulation. Note that this includes not only mesh elements, but also inactive triangles and "ghost" triangles that AlgoMesh stores internally around the outer edges of a mesh's convex hull.

For an accurate accounting of the number of cells or elements in a mesh, see the cell count display in the main AlgoMesh window.

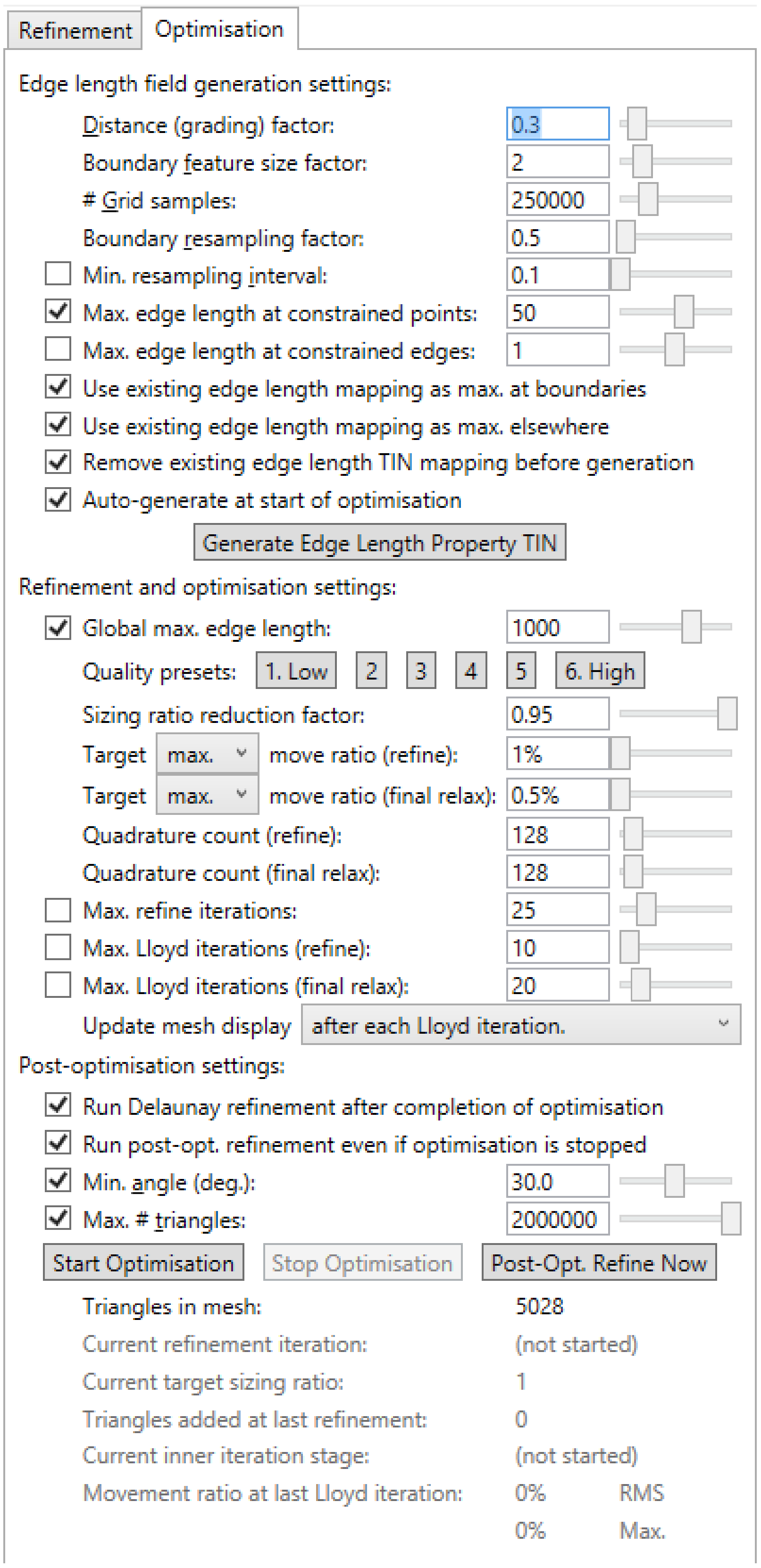
Triangles to be Processed

The number of triangles yet to be processed by the Delaunay refinement method. This number is updated dynamically as the algorithm executes. It increases as bad triangles are identified, and decreases as they are processed.

4.1.4.2 Optimisation

The Optimisation panel, shown in Figure 142, provides access to the multi-level optimisation mesh generation algorithm, including the edge length field generation process, and associated settings.

See the how-to guides Generate a High-Quality Triangular Mesh using Multi-Level Optimisation and Generate a High-Quality Voronoi Mesh using Multi-level Optimisation for an introduction to using the multi-level optimisation algorithm.



Refinement | Optimisation

Edge length field generation settings:

Distance (grading) factor: 0.3
Boundary feature size factor: 2
Grid samples: 250000
Boundary resampling factor: 0.5
☐ Min. resampling interval: 0.1
☑ Max. edge length at constrained points: 50
☐ Max. edge length at constrained edges: 1
☑ Use existing edge length mapping as max. at boundaries
☑ Use existing edge length mapping as max. elsewhere
☑ Remove existing edge length TIN mapping before generation
☑ Auto-generate at start of optimisation

Generate Edge Length Property TIN

Refinement and optimisation settings:

☑ Global max. edge length: 1000
Quality presets: 1. Low | 2 | 3 | 4 | 5 | 6. High
Sizing ratio reduction factor: 0.95
Target max. move ratio (refine): 1%
Target max. move ratio (final relax): 0.5%
Quadrature count (refine): 128
Quadrature count (final relax): 128
☐ Max. refine iterations: 25
☐ Max. Lloyd iterations (refine): 10
☐ Max. Lloyd iterations (final relax): 20
Update mesh display after each Lloyd iteration.

Post-optimisation settings:

☑ Run Delaunay refinement after completion of optimisation
☑ Run post-opt. refinement even if optimisation is stopped
☑ Min. angle (deg.): 30.0
☑ Max. # triangles: 2000000

Start Optimisation | Stop Optimisation | Post-Opt. Refine Now

Triangles in mesh: 5028
Current refinement iteration: (not started)
Current target sizing ratio: 1
Triangles added at last refinement: 0
Current inner iteration stage: (not started)
Movement ratio at last Lloyd iteration: 0% RMS
0% Max.

Figure 142: The Optimisation tab, used for controlling the multi-level optimisation mesh generation process.

Edge Length Field Generation Settings

Distance (Grading) Factor

Controls how rapidly mesh cells change in size from boundaries to empty areas of the mesh domain. The recommended range is from 0.05 (almost uniform, many cells) to 1 (rapid grading from small to large, few cells). The default value is 0.4.

Boundary Feature Size Factor

Controls how many cells are generated along the boundaries of a mesh domain. The sizing of cells along the boundaries is always dependent on a combination of the spacing of input points and the distance from one boundary to other nearby boundaries, but the boundary feature size factor acts as a multiplier on this. The recommended range is from 0.5 (many smaller cells along boundaries) to 3 (fewer, more elongated cells along boundaries). The default value is 2.

# Grid Samples

The edge length field generation process produces a TIN with a combination of resampled points along mesh domain boundaries, intermediate points between boundaries, and regular grid points. Target mesh cell edge length values are calculated at each of these points. This setting controls the number of regular grid points that are used in this process. The default value is 25000. Recommended values are 25000 to 5000000. Higher values for this setting will result in more resolution in the resulting property TIN, but will also increase the memory and space requirements for the property TIN. For example, if two constrained cell size areas are quite close to one another, the default resolution may not be fine enough to allow cell size to increase in the empty space between these areas, and may instead result in a strip of minor over-refinement connecting the two areas. Increasing the number of grid samples may eliminate this problem.

Boundary Resampling Factor

Controls how finely the boundaries are resampled for computation of local feature size in the edge length field generation process, as a factor of boundary line segment length. Smaller values will result in over-refinement along boundaries, particularly near tighter angles or in areas of congested geometry. Higher values may mitigate over-refinement along boundaries, but the resulting mesh should be checked to ensure that cell quality is not compromised. The recommended range is from 0.25 to 2. The default value is 0.5.

Min. Resampling Interval

If enabled, places a constraint on how finely the boundaries are resampled for computation of local feature size in the edge length field generation process, as an absolute minimum distance between resampled points. This has a similar effect to the boundary resampling factor, but acts as an absolute minimum distance instead of a factor on segment length. The value used is dependent on the spatial scale of the mesh domain. By default, this setting is turned off.

Max. Edge Length at Constrained Points

If enabled, constrains the edge length value assigned at lone constrained points in the mesh domain to the given maximum value or less. This setting applies globally to all constrained points to which no line segments are connected throughout the mesh domain. By default, this setting is turned off.

Max. Edge Length at Constrained Edges

If enabled, constrains the edge length value assigned at constrained edges in the mesh domain to the given maximum value or less. This setting applies globally to all constrained line segments throughout the mesh domain. By default, this setting is turned off.

Use Existing Edge Length Mapping as Max. at Boundaries

If enabled, uses any existing Mesh Cell Edge Length or Mesh Cell Edge Length (Field Generation Only) mappings as maximum constrained edge length values at boundary points (constrained points or points lying on constrained edges) in the mesh domain. By default, this setting is turned on.

Use Existing Edge Length Mapping as Max. Elsewhere

If enabled, uses any existing Mesh Cell Edge Length or Mesh Cell Edge Length (Field Generation Only) mappings as maximum constrained edge length values at points that do not lie on boundaries in the mesh domain. By default, this setting is turned on.

Remove Existing Edge Length TIN Mapping before Generation

If enabled, removes any existing mapping to Mesh Cell Edge Length before starting the edge length field generation process. This ensures that any previous edge length field generated by the process will not influence the next generation. By default, this setting is turned on.

Auto-Generate at Start of Optimisation

If enabled, AlgoMesh will first run the edge length field generation process to produce a property TIN with a Mesh Cell Edge Length mapping when the Start Optimisation button is clicked. If turned off, the optimisation process will instead use whatever Mesh Cell Edge Length TIN mapping is already present, instead of re-running the edge length field generation process. By default, this setting is turned on. It turns off automatically after edge length field generation has been run, and turns back on again when any of the field generation settings are modified.

Generate Edge Length Property TIN Button

Runs the edge length field generation process immediately to produce a property TIN with a Mesh Cell Edge Length mapping.

Refinement and Optimisation Settings

Global Max. Edge Length

If enabled, constrains the maximum length of any triangle edge in the triangular mesh. This has the effect of globally limiting cell size over the generated mesh (node to node distance for triangular meshes, or centre-to-centre distance for Voronoi meshes).

Quality Presets (1. Low, 2, 3, 4, 5, 6. High)

Sets the sizing ratio reduction factor, target move ratio (refine), target move ratio (final relax), quadrature count (refine) and quadrature count (final relax) settings automatically to preset values, depending on the quality level selected:

Quality Preset

	1	2	3	4	5	6
Sizing ratio reduction factor	0.5774	0.5774	0.5774	0.75	0.9	0.95
Target move ratio (refine)	3% RMS	0.5% RMS	3% max.	3% max.	2% max.	1% max.
Target move ratio (final relax)	1% RMS	0.1% RMS	1% max.	1% max.	1% max.	0.5% max.
Quadrature count (refine)	6	24	24	24	64	128
Quadrature count (final relax)	6	12	12	12	64	128

These quality presets allow the optimisation settings to be quickly and easily set to an approximate desired quality level - from 1, being lowest, to 6, being highest. Lower quality levels will produce a mesh more quickly, but typically with more cells of lower quality. Higher quality levels tend to minimise cell count and produce cells of high quality, but take longer to generate.

Note that any individual changes made to the affected settings will be overwritten when one of the quality preset buttons is clicked.

Sizing Ratio Reduction Factor

In a single refinement iteration of the optimisation algorithm, triangles are identified as candidates for refinement depending on the ratio between their current size and the desired size from the Mesh Cell Edge Length mapping. Any triangle whose ratio is greater than the current *target sizing ratio* for the iteration is a candidate for refinement. The sizing ratio reduction factor controls how quickly the target sizing ratio is reduced with each successive iteration. Higher values tend to produce fewer additional cells per iteration, resulting in a closer to optimal global configuration with fewer cells overall, but the optimisation process takes longer to run. Lower values produce more additionals cells per iteration, speeding up the optimisation process but typically producing a less optimal global configuration with more cells overall. The recommended range for this setting is 0.5 to 0.95.

Target Move Ratio (Refine)

At each refinement iteration of the optimisation algorithm, after selected triangles are refined to produce additional mesh cells, a weighted Lloyd relaxation (smoothing) process is run to optimise the current mesh configuration according to the mapped edge length field. This smoothing process iteratively moves each vertex of the triangular mesh to the weighted centroid of its dual Voronoi cell, until the distance of movement recedes below a certain percentage of the Voronoi cell width. The target move ratio defines this percentage, and specifies whether the process continues until all vertices have moved at most this percentage (*max.*) or until the root-mean-square movement of all vertices is below this percentage (*RMS*).

Target Move Ratio (Final Relax)

This setting is the same as the target move ratio (refine) setting, but applied to the final relaxation process, once no more refinement is to take place.

Quadrature Count (Refine)

The number of quadratures used in the Lloyd relaxation process to divide the space contained in each Voronoi cell in order to approximately integrate values over the edge length field. Higher values result in the optimisation acting on a more accurate representation of the edge length field, but slow down the optimisation process. The recommended range is from 6 to 128.

Quadrature Count (Final Relax)

This setting is the same as the quadrature count (refine) setting, but applied to the final relaxation process, once no more refinement is to take place.

Max Refine Iterations

If enabled, stops the optimisation process early once the given number of refinement iterations have been run. By default, this setting is turned off.

Max. Lloyd Iterations (Refine)

If enabled, stops each refinement iteration of the optimisation process early once the given number of Lloyd relaxation iterations have been run. By default, this setting is turned off. In most circumstances, this works without any issues, but if you notice the optimisation getting stuck moving the same points back and forth and the process does not complete, turning this setting on may be useful to force the process to continue.

Max. Lloyd Iterations (Final Relax)

This setting is the same as the max. Lloyd iterations (refine) setting, but applied to the final relaxation process, once no more refinement is to take place.

Update Mesh Display

Controls how frequently AlgoMesh's mesh display is updated during the optimisation process: *after each Lloyd iteration* (most frequently), *after each refinement iteration* (occasionally) or *after entire optimisation* (only upon completion). AlgoMesh spends some time building the visual representation of the mesh for display, so making the visual updates less frequent may speed up the overall process somewhat.

Post-Optimisation Settings

Run Delaunay Refinement after Completion of Optimisation

If enabled, runs the Delaunay refinement mesh generation process incrementally on the mesh after the optimisation is complete. The optimisation process tends to produce very high quality cells globally, but may leave a few poor quality cells near some boundaries; following it with a Delaunay refinement process helps to eliminate these bad cells, usually without adding too many cells to the mesh overall. In particular, this process eliminates any concave Voronoi cells that may otherwise be generated along some boundaries. This setting is turned on by default.

Run Post-Opt. Refinement even if Optimisation is Stopped

Runs the post-optimisation Delaunay refinement process even if the optimisation process is stopped early by clicking the Stop Optimisation button. This setting is turned on by default.

Min. Angle (Deg.)

The minimum angle constraint used for the post-optimisation Delaunay refinement process. The default value is 20 degrees. See the Delaunay refinement settings documentation for more information.

Max. # Triangles

The maximum triangle count stop criterion for the Delaunay refinement process. The default value is 2000000. Typically this can be left at the default; it should only be increased if more than two million triangles are expected in the final triangular mesh after optimisation. See the Delaunay refinement settings documentation for more information.

Execution Control Buttons

Start Optimisation

Starts execution of the multi-level optimisation process.

Stop Optimisation

Stops execution of the multi-level optimisation process early. This is only enabled while the edge length field generation or optimisation algorithms are executing. The current step or iteration of the process will be completed before stopping.

Post-Opt. Refine Now

Runs the Delaunay refinement process immediately, incrementally on the existing generated mesh, using the post-optimisation settings specified in the Optimisation tab (does not use the settings in the Refinement tab).

Progress Information

Triangles in Mesh

Displays the current number of triangles in the underlying triangulation. Note that this includes not only mesh elements, but also inactive triangles and "ghost" triangles that AlgoMesh stores internally around the outer edges of a mesh's convex hull.

For an accurate accounting of the number of cells or elements in a mesh, see the cell count display in the main AlgoMesh window.

Current Refinement Iteration

Reports the current refinement iteration number or the current step in the edge length field generation or optimisation process.

Current Target Sizing Ratio

Displays the target sizing ratio of the current refinement iteration. This gradually decreases to 1 with each refinement iteration, after which all refinements that occur are attempting to bring the mesh to its ultimate resolution. Once the target sizing ratio is 1 and no more triangles are found as candidates for refinement (i.e. all cells meet the mesh cell edge length mapping), the final relaxation step takes place and then the optimisation completes.

Triangles Added at last Refinement

Reports the number of triangles that were added to the underlying triangulation at the last refinement step. When this value reaches 0, and the target sizing ratio is 1, the optimisation process moves into the final relaxation step.

Current Inner Iteration Stage

The stage of the optimisation process now being executed in the current refinement iteration: *Refinement* (adding new triangles) or *Lloyd #X* (the *X*th iteration of Lloyd relaxation).

Movement Ratio at last Lloyd Iteration

The root-mean square and maximum vertex movement ratios in the last Lloyd relaxation iteration, as a percentage of respective Voronoi cell width.

4.2 Other Windows

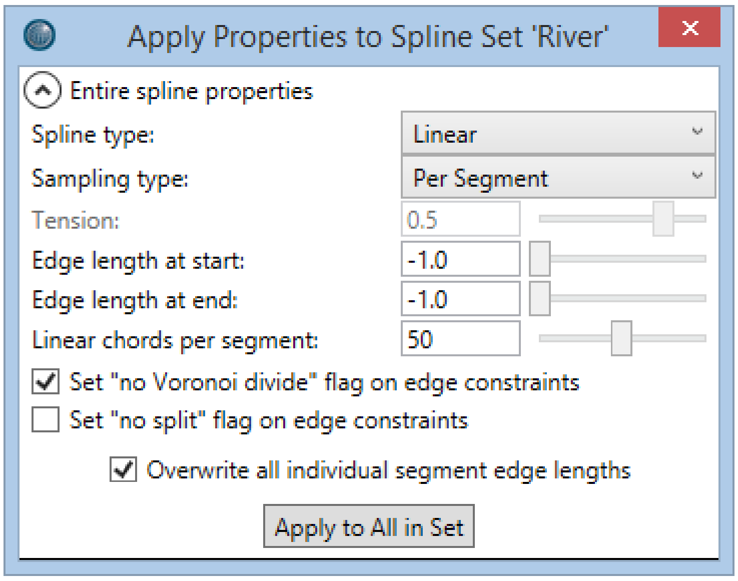
There are a number of user interface windows in AlgoMesh that appear when particular menu options are selected. The following sections detail each of these windows.

4.2.1 Apply Properties to Spline Set

The Apply Properties to Spline Set window, shown in Figure 143, allows spline geometric properties and edge constraint flags to be set for all splines in a given spline set at once.

Unlike the Spline Properties window, settings changed in this window do not take immediate effect; instead they take effect only when the Apply to All in Set button is clicked.

See the Manually Digitise and Edit Splines how-to section for a tutorial on manipulating splines and their properties.


Apply Properties to Spline Set 'River'

Entire spline properties

Spline type:	Linear
Sampling type:	Per Segment
Tension:	0.5
Edge length at start:	-1.0
Edge length at end:	-1.0
Linear chords per segment:	50

[✓] Set "no Voronoi divide" flag on edge constraints

[ ] Set "no split" flag on edge constraints

[✓] Overwrite all individual segment edge lengths

Figure 143: The Apply Properties to Spline Set window.

Spline Type

Specifies whether the spline is sampled as a linear polyline (*Linear*), a natural cubic spline (*Natural*) or a cardinal cubic spline (*Cardinal*). Linear is recommended for most modelling purposes, or Cardinal if representing a smoothly curving feature. Linear is the default.

Sampling Type

Determines whether the spline is resampled from start to end along its length, ignoring its generating points (*Entire Spline*), or resampled at each generating point and at intervals along the segments between the generating points (*Per Segment*). In the latter case, resampling intervals may be controlled independently along each segment by modifying the individual segment properties.

Tension

Controls how "tightly" a cardinal cubic spline curve follows its sequence of generating points. A tension of 1.0 is equivalent to a linear polyline. A tension of 0.5 is the default. This setting does not apply for linear or natural splines.

Edge Length at Start

Sets the desired distance between sampled points along the spline at its starting point. A positive number indicates an absolute distance, whereas a negative number indicates that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number (e.g. -10 would equate to whatever distance would divide the spline into (10 times number of segments) equal parts). When start and end edge length values differ, AlgoMesh will grade the resampling interval between the two values in a geometric progression. Which point of the spline is the start and which is the end is determined by the order in which the spline's polyline was defined.

Edge Length at End

Sets the desired distance between sampled points along the spline at its ending point. A positive number indicates an absolute distance, whereas a negative number indicates that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number (e.g. -10 would equate to whatever distance would divide the spline into (10 times number of segments) equal parts). When start and end edge length values differ, AlgoMesh will grade the resampling interval between the two values in a geometric progression. Which point of the spline is the start and which is the end is determined by the order in which the spline's polyline was defined.

Linear Chords per Segment

For natural and cardinal cubic splines, AlgoMesh generates a linear approximation to the spline curve for the purposes of display and sampling. This setting controls the number of linear chords used internally to approximate each segment of the spline curve. This setting is not used for splines with Spline Type set to *Linear*.

Set "No Voronoi Divide" Flag on Edge Constraints

Sets the "no Voronoi divide" edge constraint flag on all edge constraints added into the mesh by this spline. See the advanced topic on Edge Constraint Flags and No Refinement Areas for details on edge constraint flags.

Set "No Split" Flag on Edge Constraints

Sets the "no split" edge constraint flag on all edge constraints added into the mesh by this spline. See the advanced topic on Edge Constraint Flags and No Refinement Areas for details on edge constraint flags.

Overwrite All Individual Segment Edge Lengths

If enabled, causes the *Edge Length at Start* and *Edge Length at End* values to be propagated to the individual segment properties for all spline segments, so that they may be used with a Sampling Type of *Per Segment*. By default, this setting is turned on.

Apply to All in Set Button

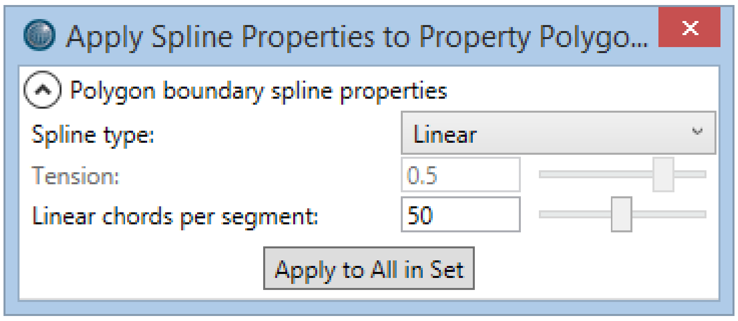
Overwrites the spline properties for all splines in the spline set to those specified.

4.2.2 Apply Spline Properties to Property Polygon Set

The Apply Spline Properties to Property Polygon Set window, shown in Figure 144, allows the geometric properties of polygon boundaries to be set for all polygons in a given property polygon set at once. These properties are similar to those in the Spline Properties window, but resampling settings and edge constraint flags are not present, as property polygons are used only for mesh variable mapping and do not explicitly add any geometry to the mesh.

Unlike the Polygon Properties window, settings changed in this window do not take immediate effect; instead they take effect only when the Apply to All in Set button is clicked.

See the Manually Digitise and Edit Splines how-to section for a tutorial on manipulating splines and their properties; property polygons use splines for their boundaries and many of the same properties apply.


Apply Spline Properties to Property Polygo...
Polygon boundary spline properties
Apply to All in Set

Spline type: Linear
Tension: 0.5
Linear chords per segment: 50

Figure 144: The Apply Spline Properties to Property Polygon Set window.

Spline Type

Specifies whether the spline is sampled as a linear polyline (*Linear*), a natural cubic spline (*Natural*) or a cardinal cubic spline (*Cardinal*). Linear is recommended for most modelling purposes, or Cardinal if representing a smoothly curving feature. Linear is the default.

Tension

Controls how "tightly" a cardinal cubic spline curve follows its sequence of generating points. A tension of 1.0 is equivalent to a linear polyline. A tension of 0.5 is the default. This setting does not apply for linear or natural splines.

Linear Chords per Segment

For natural and cardinal cubic splines, AlgoMesh generates a linear approximation to the spline curve for the purposes of display and sampling. This setting controls the number of linear chords used internally to approximate each segment of the spline curve. This setting is not used for splines with Spline Type set to *Linear*.

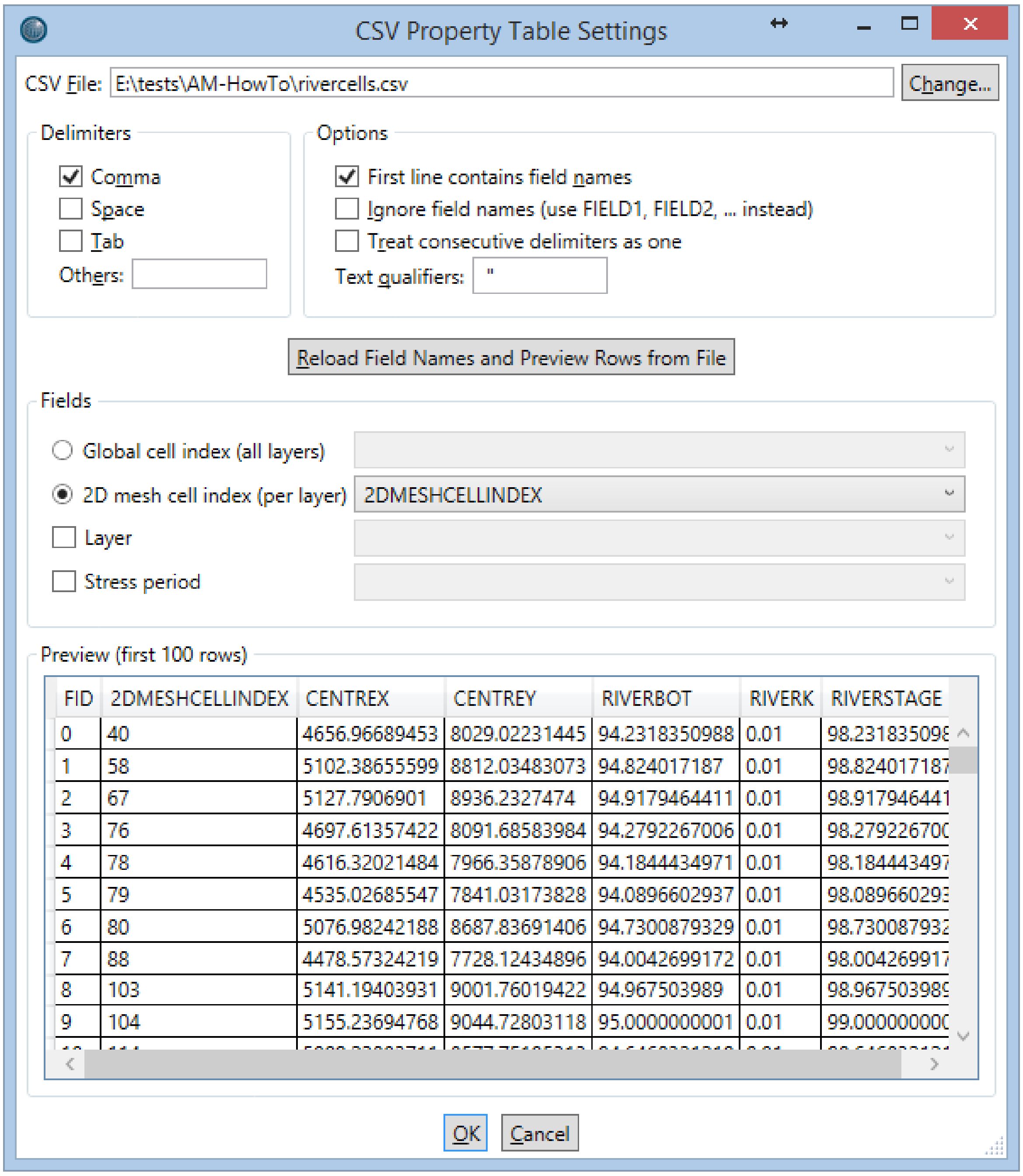
Apply to All in Set Button

Overwrites the spline properties for the boundaries of all polygons in the property polygon set to those specified.

4.2.3 CSV Property Table Settings

The CSV Property Table Settings window, shown in Figure 145, allows you specify how AlgoMesh should read a given property table text file.

See the how-to guide Set up a Steady-State MODFLOW-USG Model for a tutorial on mapping model properties using CSV property tables and other mechanisms.


CSV Property Table Settings
Change...
Reload Field Names and Preview Rows from File
OK
Cancel

CSV File: E:\tests\AM-HowTo\rivercells.csv

Delimiters
- [✓] Comma
- [ ] Space
- [ ] Tab
- Others:

Options
- [✓] First line contains field names
- [ ] Ignore field names (use FIELD1, FIELD2, ... instead)
- [ ] Treat consecutive delimiters as one
- Text qualifiers: "

Fields
- ○ Global cell index (all layers)
- ● 2D mesh cell index (per layer): 2DMESHCELLINDEX
- [ ] Layer
- [ ] Stress period

Preview (first 100 rows)

FID	2DMESHCELLINDEX	CENTREX	CENTREY	RIVERBOT	RIVERK	RIVERSTAGE
0	40	4656.96689453	8029.02231445	94.2318350988	0.01	98.23183509
1	58	5102.38655599	8812.03483073	94.824017187	0.01	98.82401718
2	67	5127.7906901	8936.2327474	94.9179464411	0.01	98.91794644
3	76	4697.61357422	8091.68583984	94.2792267006	0.01	98.27922670
4	78	4616.32021484	7966.35878906	94.1844434971	0.01	98.18444349
5	79	4535.02685547	7841.03173828	94.0896602937	0.01	98.08966029
6	80	5076.98242188	8687.83691406	94.7300879329	0.01	98.73008793
7	88	4478.57324219	7728.12434896	94.0042699172	0.01	98.00426991
8	103	5141.19403931	9001.76019422	94.967503989	0.01	98.96750398
9	104	5155.23694768	9044.72803118	95.0000000001	0.01	99.00000000

Figure 145: The CSV Property Table Settings window.

CSV File

The path and name of the loaded property table file are reported here. The file may be changed by clicking the **Change** button and browsing to a new file. When the file is changed, the field names and the preview at the bottom of the window are refreshed.

Delimiters

One or more ASCII characters used as field delimiters in the table file may be specified here:

commas, spaces, tabs, or a list of other specified characters. The default is that only commas are taken to be delimiters.

Options

First Line Contains Field Names

If turned on, AlgoMesh does not attempt to read values from the first line of the file; it will only use this line to determine the names of each field. Initially, when a file is first loaded, AlgoMesh tries to detect automatically whether the first line contains numerical values or not. If it does, this setting is turned off by default; otherwise, it is turned on.

Ignore Field Names (Use FIELD1, FIELD2, ... Instead)

If turned on, this causes AlgoMesh to ignore any field names it reads from the first line of the file and instead to use generic field names FIELD1, FIELD2, etc. By default, this setting is turned off.

Treat Consecutive Delimiters as One

If turned on, multiple consecutive instances of delimiter characters in the file are considered a single break between fields. If turned off, each consecutive delimiter character encountered is assumed to indicate an additional (empty) field. By default, this setting is turned off.

Text Qualifiers

A list of ASCII characters used to denote text strings in the table file. Delimiter characters after a text qualifier are ignored until a matching text qualifier is encountered to close the text string. By default, the double quotation mark character **"** is used as a text qualifier.

Reload Field Names and Preview Rows from File Button

Clicking this button causes AlgoMesh to re-read the table file using the currently specified delimiters and options. Field names and the preview at the bottom of the window are refreshed.

Fields

Either a *Global cell index* or a *2D mesh cell index* field must be specified so that AlgoMesh knows how to assign each row of the property table to a particular cell in the model.

Global Cell Index (All Layers)

If the global cell index radio button is selected, the field assigned to it is used to index cell-by-cell mappings as a global cell index over the entire 3D model. Note that a global cell index may not be used to map Top Elevation or Bottom Elevation if Clean Layer Elevations is turned on in Model Setup, and may not be used to map Active Model Area at all, as these properties affect the way cells are index globally. To map to these properties, use the 2D mesh cell index option instead.

2D Mesh Cell Index (Per Layer)

If the 2D mesh cell index radio button is selected, the field assigned to it is used to index cell-by-cell mappings as the 2D mesh cell index within a layer of the 3D model. The 2D mesh cell indexing scheme is identical for each layer and does not change when cells are removed from the 3D model by means of pinch-outs or Active Model Area mappings.

Layer

If turned on, the field assigned to Layer is used to specify the layer of the cell to which each row of the property table applies. This setting is disabled when a *Global cell index* field is selected. If *2D mesh cell index* is selected and the Layer field option is turned off, the layer will instead be specified by each mesh variable mapping in the project explorer.

Stress Period

If turned on, the field assign to Stress Period is used to specify the stress period to which each row of the property table applies, for any transient mappings made to the table fields. If the Stress Period field option is turned off, the stress period will instead be specified by each mesh variable mapping in the project explorer.

Preview

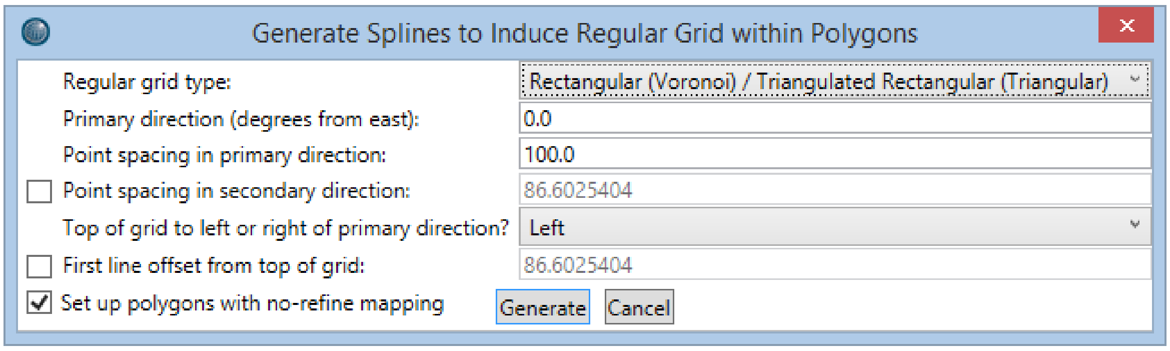
This section of the window shows the first 100 rows of the table, as interpreted by AlgoMesh using the current delimiters, options and field settings. Values in any fields specified as cell, layer or stress period indices are interpreted as integers. Values in all other fields are interpreted as real numbers. Any field data that cannot be converted to a number will be shown as ??? (including text strings), and may cause problems if mapped to a mesh variable in AlgoMesh.

4.2.4 Generate Splines to Induce Regular Grid within Polygons

The Generate Splines to Induce Regular Grid within Polygons window, shown in Figure 146, facilitates the building of sampled spline input geometry that induces a structured sub-grid.

The spline generation process induces a structured sub-grid by generating a series of parallel, "no split" splines along a given direction within all polygons in a polygon set. In the rectangular Voronoi grid case, points are sampled along these splines at a specified point spacing to produce rectangular cells. In the hexagonal Voronoi grid case, the same is done but the starting position of the resampled points on every second spline is offset by half the specified point spacing, such that equilateral triangles are produced in the triangular grid (equating to perfect hexagons in the Voronoi grid).

See the advanced topic on Structured Sub-Grid Generation for further discussion and examples of structured sub-grids.



Regular grid type:	Rectangular (Voronoi) / Triangulated Rectangular (Triangular)
Primary direction (degrees from east):	0.0
Point spacing in primary direction:	100.0
☐ Point spacing in secondary direction:	86.6025404
Top of grid to left or right of primary direction?	Left
☐ First line offset from top of grid:	86.6025404
☑ Set up polygons with no-refine mapping	

Figure 146: The Generate Splines to Induce Regular Grid within Polygons window.

Regular Grid Type

Specifies whether a rectangular Voronoi grid (triangulated rectangular triangular grid) or hexagonal Voronoi grid (equilateral triangular grid) is to be generated.

Primary Direction (Degrees from East)

The direction along which the generated splines should run, in degrees counter-clockwise from east.

Point Spacing in Primary Direction

The spacing between sample points to be generated along each spline.

Point Spacing in Secondary Direction

The spacing between generated splines, perpendicular to the primary direction. If turned off, this value is calculated automatically to generate a perfect square or hexagonal Voronoi grid, depending on the selected *Regular grid type*. Turning this on and specifying an appropriate value allows the cells to be elongated from their default square or hexagonal shape.

Top of Grid to Left or Right of Primary Direction?

Determines which side of each polygon the series of generated splines should start. For a primary direction between 0 and 90 degrees, *Left* here would approximately indicate the northern side of each polygon, and *Right* would approximately indicate the southern side.

First Line Offset from Top of Grid

The distance from the "top" of each polygon at which the first spline should be placed. If turned off, this will be automatically set equal to the *Point spacing in secondary direction*.

Set up Polygons with No-Refine Mapping

If turned on, adds to the polygon set a property called NO_REFINE with a default value of 1, and maps it to No Refinement Area, to ensure that the mesh generator does not add extra mesh cells within the structured sub-grid region. See the advanced topic on Edge Constraint Flags and No Refinement Areas for more information.

Generate Button

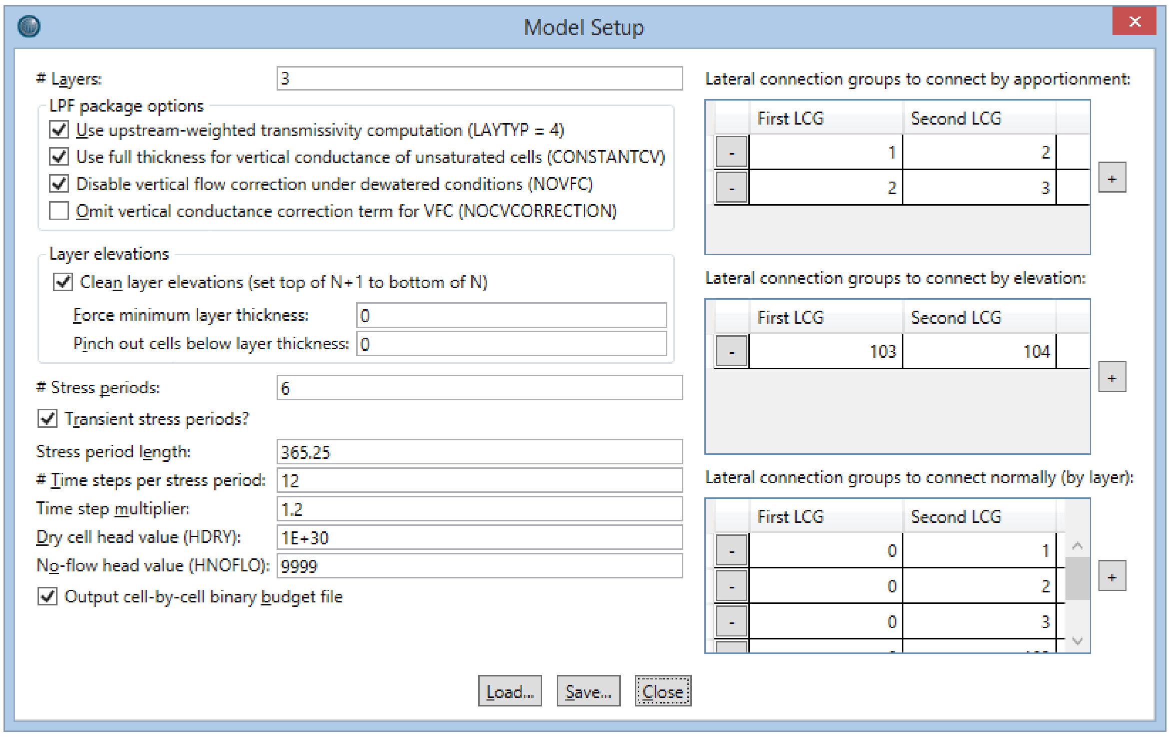
Generates splines according to the specified settings, and closes the window.

Cancel Button

Closes the window without generating any splines.

4.2.5 Model Setup

The Model Setup window, shown in Figure 147, allows specification of various settings that define the make-up of a MODFLOW-USG model produced by AlgoMesh. Several of these settings correspond directly to options in the MODFLOW-USG input files; see the input/output reference documentation that comes with MODFLOW-USG for more information.



Layers: 3

LPF package options

- ☑ Use upstream-weighted transmissivity computation (LAYTYP = 4)
- ☑ Use full thickness for vertical conductance of unsaturated cells (CONSTANTCV)
- ☑ Disable vertical flow correction under dewatered conditions (NOVFC)
- ☐ Omit vertical conductance correction term for VFC (NOCVCORRECTION)

Layer elevations

- ☑ Clean layer elevations (set top of N+1 to bottom of N)

Force minimum layer thickness: 0

Pinch out cells below layer thickness: 0

Stress periods: 6

☑ Transient stress periods?

Stress period length:	365.25
# Time steps per stress period:	12
Time step multiplier:	1.2
Dry cell head value (HDRY):	1E+30
No-flow head value (HNOFLO):	9999

☑ Output cell-by-cell binary budget file

Lateral connection groups to connect by apportionment:

First LCG	Second LCG
1	2
2	3

Lateral connection groups to connect by elevation:

First LCG	Second LCG
103	104

Lateral connection groups to connect normally (by layer):

First LCG	Second LCG
0	1
0	2
0	3

Figure 147: The Model Setup window.

General Settings

# Layers

Specifies the number of MODFLOW-USG model layers (NLAY).

Use Upstream-Weighted Transmissivity Computation (LAYTYP = 4)

If turned on, the LPF package LAYTYP flags for each layer will be set to 4 (upstream weighting). If turned off, the LAYTYP flags will be set to 3 (convertible). By default, this setting is turned on.

Use Full Thickness for Vertical Conductance of Unsaturated Cells (CONSTANTCV)

If turned on, the keyword CONSTANTCV will be added to the LPF package header. By default, this setting is turned on.

Disable Vertical Flow Correction under Dewatered Conditions (NOVFC)

If turned on, the keyword NOVFC will be added to the LPF package header. By default, this setting is turned on.

Omit Vertical Conductance Correction Term for VFC (NOCVCORRECTION)

If turned on, the keyword NOCVCORRECTION will be added to the LPF package header. By default, this setting is turned off - but note from the MODFLOW-USG documentation that the vertical conductance correction term is omitted anyway if the NOVFC keyword is used.

Clean Layer Elevations (Set Top of N+1 to Bottom of N)

If turned on, AlgoMesh will set the top elevation of each cell to the bottom elevation of the first active cell above it. If turned off, top and bottom elevations will be used precisely as they are given, without modification. In either case, any layer with a completely missing Top Elevation mapping will have its top elevations mapped from the Bottom Elevation mapping of the layer immediately above it, if it exists, and conversely, any layer with a completely missing Bottom Elevation mapping will have its bottom elevations mapped from the Top Elevation mapping of the layer immediately below it, if it exists.

Force Minimum Layer Thickness

Lowers the bottom elevation of any cell with less than the given minimum thickness such that the minimum thickness is met. This setting may override the *Pinch out cells below layer thickness* setting.

Pinch Out Cells Below Layer Thickness

Removes from the model any cells with less than the given thickness. Note that the *Force minimum layer thickness* setting takes precedence over this if set to a larger value. See the Remove or Pinch Out Unnecessary MODFLOW-USG Model Cells how-to guide for a tutorial on pinching out model cells.

# Stress Periods

Specifies the number of stress periods (NPER) in the MODFLOW-USG simulation.

Transient Stress Periods?

If turned on, all stress periods output to the MODFLOW-USG DISU package will be set as transient (TR) periods. If turned off, all stress periods will be set as steady-state (SS) periods.

Stress Period Length

Specifies the length of time (PERLEN) to be assigned to each stress period in the simulation. If stress periods of varying lengths are required, the stress period setup near the end of DISU file produced by AlgoMesh should be modified manually after exporting MODFLOW-USG files.

# Time Steps per Stress Period

Specifies the number of time steps (NSTP) to be assigned to each stress period in the simulation. If varying numbers of time steps per stress period are required, the stress period setup near the end of DISU file produced by AlgoMesh should be modified manually after exporting MODFLOW-USG files.

Time Step Multiplier

Specifies the time step multiplier (TSMULT) to be assigned to each stress period in the simulation. If varying numbers of time steps per stress period are required, the stress period setup near the end of DISU file produced by AlgoMesh should be modified manually after exporting MODFLOW-USG files.

Dry Cell Head Value (HDRY)

Specifies the head to be assigned to cells that are converted to dry during a MODFLOW-USG simulation (HDRY), written to the LPF package header.

No-Flow Head Value (HNOFLO)

Specifies the head to be assigned to no-flow cells (HNOFLO), written to the BAS package.

Output Cell-by-Cell Binary Budget File

If turned on, AlgoMesh will specify a single .CBB binary output file in the MODFLOW-USG .NAM file, and direct the cell-by-cell flow outputs for all packages to this file. If turned off, cell-by-cell flow outputs will be omitted. By default, this setting is turned on.

Lateral Connection Group Connection Settings

The Lateral Connection Group (LCG) settings on the right-hand side of the Model Setup window allows advanced control over how AlgoMesh creates lateral connections between cells in the MODFLOW-USG model. See the Lateral Connection Groups advanced topic for a discussion of these settings.

Buttons

Load

Loads model setup values from a Model Setup File (.msf), overwriting all existing values in the window.

Save

Saves all model setup values to a Model Setup File (.msf). Note that this is not necessary to save changes to model setup values within the AlgoMesh project; this is only necessary to export a given model setup for import into a different project.

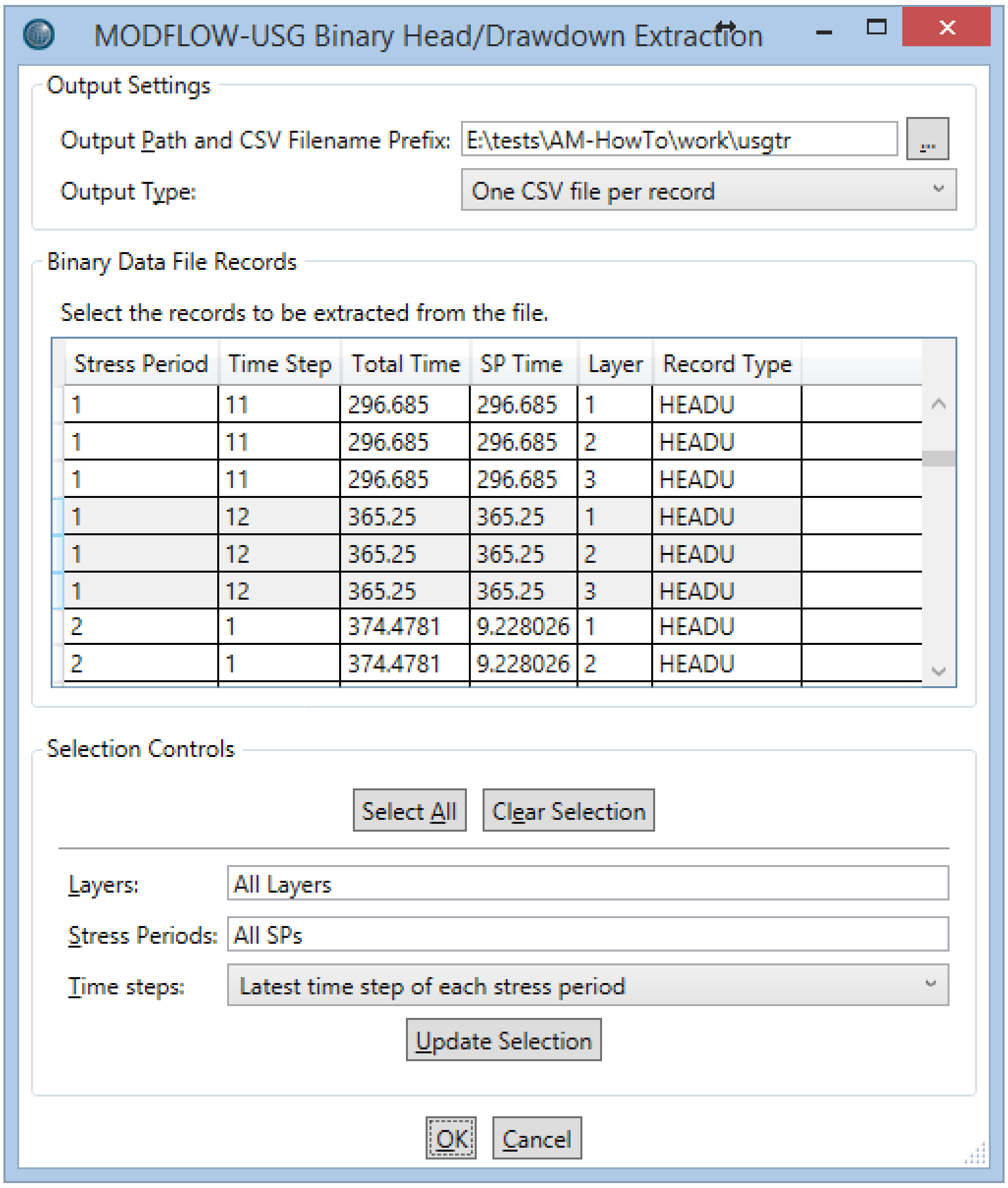
Close

Closes the Model Setup window. Any changes made are retained (i.e. there is no Cancel button).

4.2.6 MODFLOW-USG Binary Head/Drawdown Extraction

The MODFLOW-USG Binary Head/Drawdown Extraction window, shown in Figure 148, allows any number of result entries to be extracted from a binary MODFLOW-USG binary head or drawdown file and exported to one or many CSV files. The CSV files produced may be easily read into other software tools for post-processing.

The Load MODFLOW-USG Results how-to section features a brief introduction to this tool.



Output Path and CSV Filename Prefix: E:\tests\AM-HowTo\work\usgtr

Output Type: One CSV file per record

Select the records to be extracted from the file.

Stress Period	Time Step	Total Time	SP Time	Layer	Record Type
1	11	296.685	296.685	1	HEADU
1	11	296.685	296.685	2	HEADU
1	11	296.685	296.685	3	HEADU
1	12	365.25	365.25	1	HEADU
1	12	365.25	365.25	2	HEADU
1	12	365.25	365.25	3	HEADU
2	1	374.4781	9.228026	1	HEADU
2	1	374.4781	9.228026	2	HEADU

Layers: All Layers

Stress Periods: All SPs

Time steps: Latest time step of each stress period

Figure 148: The MODFLOW-USG Binary Head/Drawdown Extraction window.

Output Settings

Output Path and CSV Filename Prefix

A path and start of an output filename to be written by the extraction process. If Output Type is *All records in a single CSV file*, a single CSV file is generated by appending **.csv** to this filename prefix. If Output Type is *One CSV file per record*, multiple files of the form **<Prefix>_SP*x*_TS*y*_L*z*.csv** will be generated, where ***x*** is the corresponding record's stress period, ***y*** is the time step, and ***z*** is the layer.

Output Type

Specifies whether a single CSV file for all results or multiple CSV files (one per data file record) should be produced.

Binary Data File Records

This section presents a table listing the stress period, time step, total time, stress period time, layer and record type of all of the result entries present in the binary data file. A single entry to be exported may be selected manually by clicking on its row in the table. Multiple entries may be selected manually by holding Ctrl and clicking to add an entry to the selection, and/or by holding Shift while clicking to add a range of entries to the selection. Alternatively, the selection controls section of the window provide an automated mechanism for selecting all records pertaining to particular layers, stress periods and time steps.

By default, when the window first comes up, records are selected for all layers, in the latest time step of every stress period.

Selection Controls

The Selection Controls section provides a mechanism for automatically selecting records to export from the binary data file based on a particular choice of layers, stress periods and time steps.

Select All Button

Clicking the Select All button selects all records in the data file.

Clear Selection Button

Clicking the Clear Selection button deselects all records.

Layers

Specifies a collection of layers with which to filter the set of records selected when the Update Selection button is clicked. Multiple values may be separated by commas, e.g. **1, 2, 4, 6**. Inclusive layer ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The layer list may be left empty or the word **none** specified to signify no layers. The word **all** or an asterisk ***** may be used to signify all layers in the model.

Stress Periods

Specifies a collection of stress periods with which to filter the set of records selected when the Update Selection button is clicked. Multiple values may be separated by commas, e.g. **1, 2, 4, 6**. Inclusive stress period ranges may also be specified, using a hyphen character to separate the low and high values in the range, e.g. **1-3, 4, 5-8**. The stress period list may be left empty or the word **none** specified to signify no layers. The word **all** or an asterisk ***** may be used to signify all stress periods in the model.

Time Steps

Specifies which time steps should be included in the record selection when the Update Selection button is clicked: *Earliest time step of each stress period*, *Latest time step of each stress period*, or *All time steps*. The *Earliest* and *Latest* options are evaluated on a per-stress period basis, so that for example, the final time step in each stress period may be selected even when the number of time steps differs for each stress period.

Update Selection Button

Selects the set of data file records that pertain to the given layers in the given stress periods, and for the indicated time steps within those stress periods.

Buttons

OK

Exports all selected result records to CSV in accordance with the specified settings, and closes the window. Once extraction is done, a message box appears confirming completion of the process.

Cancel

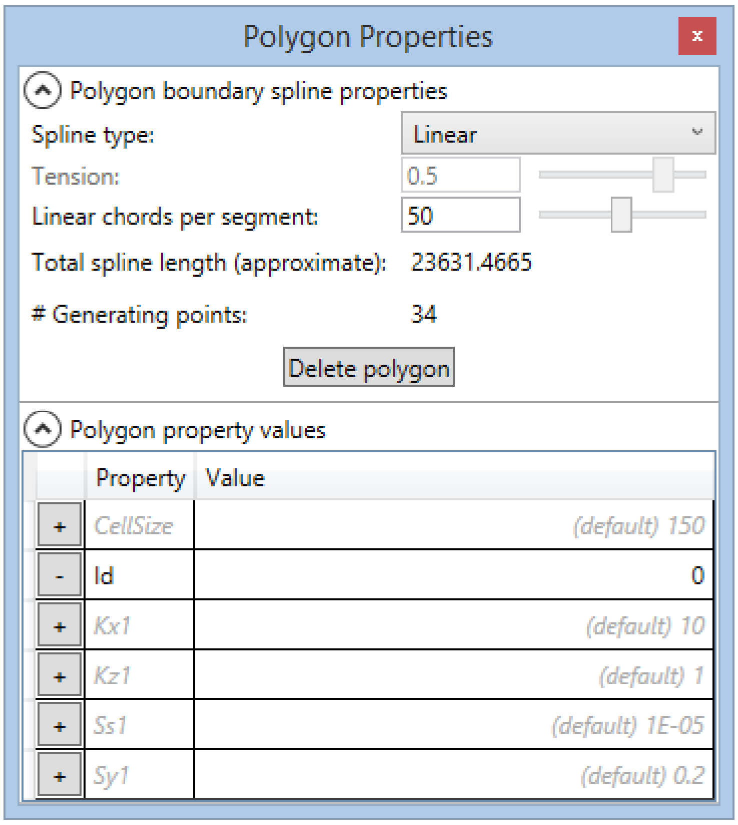
Closes the window without extracting any results.

4.2.7 Polygon Properties

The Polygon Properties window, shown in Figure 149, allows the geometric properties of a polygon's boundary to be set, and the property values associated with the polygon to be modified.

Properties modified in this window take immediate effect, i.e. there is no Apply button or similar to confirm any changes made.

See the Manually Digitise and Edit Splines how-to section for a tutorial on manipulating splines and their properties; property polygons use splines for their boundaries and many of the same properties apply.



Polygon Properties

Polygon boundary spline properties

Spline type:	Linear
Tension:	0.5
Linear chords per segment:	50
Total spline length (approximate):	23631.4665
# Generating points:	34

Polygon property values

	Property	Value
+	CellSize	(default) 150
-	Id	0
+	Kx1	(default) 10
+	Kz1	(default) 1
+	Ss1	(default) 1E-05
+	Sy1	(default) 0.2

Figure 149: The Polygon Properties window.

Polygon Boundary Spline Properties

Spline Type

Specifies whether the spline is sampled as a linear polyline (*Linear*), a natural cubic spline (*Natural*) or a cardinal cubic spline (*Cardinal*). Linear is recommended for most modelling purposes, or Cardinal if representing a smoothly curving feature. Linear is the default.

Tension

Controls how "tightly" a cardinal cubic spline curve follows its sequence of generating points. A tension of 1.0 is equivalent to a linear polyline. A tension of 0.5 is the default. This setting does not apply for linear or natural splines.

Linear Chords per Segment

For natural and cardinal cubic splines, AlgoMesh generates a linear approximation to the spline curve for the purposes of display and sampling. This setting controls the number of linear chords used internally to approximate each segment of the spline curve. This setting is not used for splines with Spline Type set to *Linear*.

Total Spline Length (Approximate)

Reports the length of the entire spline, as represented by AlgoMesh's displayed linear approximation to the spline curve.

# Generating Points

Reports the number of generating points in the spline (i.e. polyline vertices, shown as yellow squares in the main view).

Delete Polygon Button

Immediately removes this polygon from the AlgoMesh project.

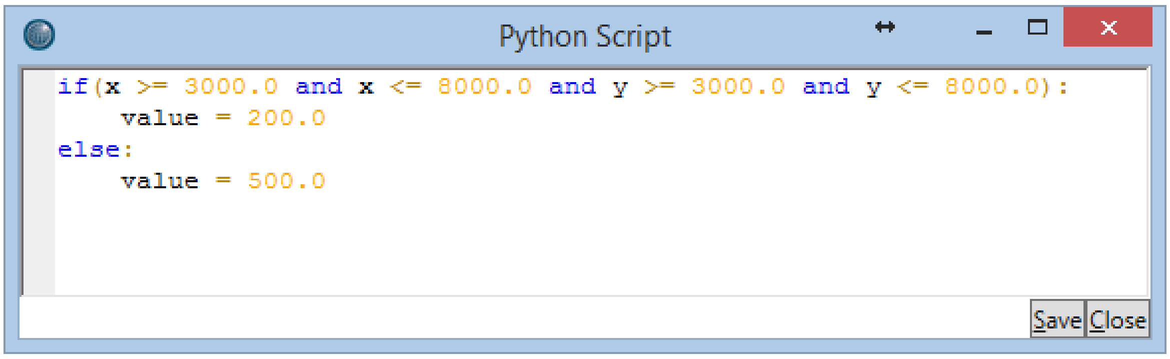
Polygon Property Values

This section contains a table listing all properties associated with the polygon's property polygon set. Each row in the table refers to a single property, giving its name and any value that is assigned to the property for this polygon. If no value is explicitly assigned to the property, the property's default value is shown in grey. Clicking the plus '**+**' button for a property with a default value causes the default value to be overridden by a specified value for this polygon. Clicking the minus '**-**' button for a property with a specified value removes the specified value from this polygon, replacing it with the default value for that property. When a specified value is present, it may be edited by clicking on the existing value and typing a new value.

4.2.8 Python Script

The Python Script window, shown in Figure 150, allows the code for an embedded Python script to be edited. A default syntax colouring scheme is provided which highlights keywords, operators, literals, and so on, to make the code more readable.

See the Python Scripts advanced topic for more information.



```
if(x >= 3000.0 and x <= 8000.0 and y >= 3000.0 and y <= 8000.0):
    value = 200.0
else:
    value = 500.0
```

Figure 150: The Python Script editor window.

Save

Saves the current code in the Python Script window, overwriting the embedded Python script. A message box appears acknowledging that the code was saved.

Close

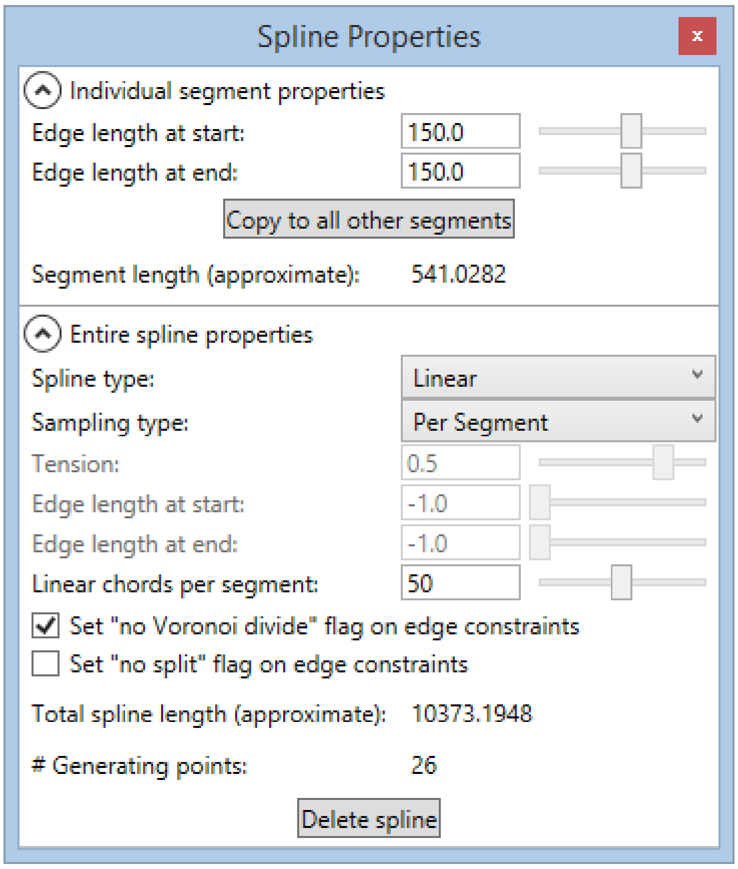
Closes the Python Script window. If the code has been modified since the last time it was saved, a message box will appear asking whether you want to save before closing.

4.2.9 Spline Properties

The Spline Properties window, shown in Figure 151, allows geometric and edge constraint properties to be set on an individual spline, and resampling intervals to be set on an individual spline segment.

Spline properties modified in this window take immediate effect, i.e. there is no Apply button or similar to confirm any changes made.

See the Manually Digitise and Edit Splines how-to section for a tutorial on manipulating splines and their properties.


Spline Properties

Individual segment properties

Edge length at start: 150.0

Edge length at end: 150.0

Segment length (approximate): 541.0282

Entire spline properties

Spline type: Linear

Sampling type: Per Segment

Tension: 0.5

Edge length at start: -1.0

Edge length at end: -1.0

Linear chords per segment: 50

[✓] Set "no Voronoi divide" flag on edge constraints

[ ] Set "no split" flag on edge constraints

Total spline length (approximate): 10373.1948

Generating points: 26

Figure 151: The Spline Properties window.

Individual Segment Properties

Individual segment properties are applied when the Sampling Type of a spline is set to *Per Segment*. If the Sampling Type is set to *Entire Spline*, the properties in this section have no effect.

Edge Length at Start

Sets the desired distance between sampled points along the spline at the starting point of the segment. A positive number indicates an absolute distance, whereas a negative number indicates that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number (e.g. -10 would equate to whatever distance would divide the segment into 10 equal parts). When start and end edge length values differ, AlgoMesh will grade the resampling interval between the two values in a geometric progression. Which point of the segment is the start and which is the end is determined by the order in which the spline's polyline was defined.

Edge Length at End

Sets the desired distance between sampled points along the spline at the ending point of the segment. A positive number indicates an absolute distance, whereas a negative number indicates that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number (e.g. -10 would equate to whatever distance would divide the segment into 10 equal parts). When start and end edge length values differ, AlgoMesh will grade the resampling interval between the two values in a geometric progression. Which point of the segment is the start and which is the end is determined by the order in which the spline's polyline was defined.

Copy to All Other Segments Button

Sets the *Edge length at start* and *Edge length at end* values for all segments in the entire spline to those specified for this segment.

Segment Length (Approximate)

Reports the length of this spline segment, as represented by AlgoMesh's displayed linear approximation to the spline curve.

Entire Spline Properties

Spline Type

Specifies whether the spline is sampled as a linear polyline (*Linear*), a natural cubic spline (*Natural*) or a cardinal cubic spline (*Cardinal*). Linear is recommended for most modelling purposes, or Cardinal if representing a smoothly curving feature. Linear is the default.

Sampling Type

Determines whether the spline is resampled from start to end along its length, ignoring its generating points (*Entire Spline*), or resampled at each generating point and at intervals along the segments between the generating points (*Per Segment*). In the latter case, resampling intervals may be controlled independently along each segment by modifying the individual segment properties.

Tension

Controls how "tightly" a cardinal cubic spline curve follows its sequence of generating points. A tension of 1.0 is equivalent to a linear polyline. A tension of 0.5 is the default. This setting does not apply for linear or natural splines.

Edge Length at Start

Sets the desired distance between sampled points along the spline at its starting point. A positive number indicates an absolute distance, whereas a negative number indicates that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number (e.g. -10 would equate to whatever distance would divide the spline into (10 times number of segments) equal parts). When start and end edge length values differ, AlgoMesh will grade the resampling interval between the two values in a geometric progression. Which point of the spline is the start and which is the end is determined by the order in which the spline's polyline was defined. This setting only applies if the Sampling Type is set to *Entire Spline*.

Edge Length at End

Sets the desired distance between sampled points along the spline at its ending point. A positive number indicates an absolute distance, whereas a negative number indicates that AlgoMesh should calculate and use the distance that would result in a number of intervals per segment equal to the magnitude of the number (e.g. -10 would equate to whatever distance would divide the spline into (10 times number of segments) equal parts). When start and end edge length values differ, AlgoMesh will grade the resampling interval between the two values in a geometric progression. Which point of the spline is the start and which is the end is determined by the order in which the spline's polyline was defined. This setting only applies if the Sampling Type is set to *Entire Spline*.

Linear Chords per Segment

For natural and cardinal cubic splines, AlgoMesh generates a linear approximation to the spline curve for the purposes of display and sampling. This setting controls the number of linear chords used internally to approximate each segment of the spline curve. This setting is not used for splines with Spline Type set to *Linear*.

Set "No Voronoi Divide" Flag on Edge Constraints

Sets the "no Voronoi divide" edge constraint flag on all edge constraints added into the mesh by this spline. See the advanced topic on Edge Constraint Flags and No Refinement Areas for details on edge constraint flags.

Set "No Split" Flag on Edge Constraints

Sets the "no split" edge constraint flag on all edge constraints added into the mesh by this spline. See the advanced topic on Edge Constraint Flags and No Refinement Areas for details on edge constraint flags.

Total Spline Length (Approximate)

Reports the length of the entire spline, as represented by AlgoMesh's displayed linear approximation to the spline curve.

# Generating Points

Reports the number of generating points in the spline (i.e. polyline vertices, shown as yellow squares in the main view).

Delete Spline Button

Immediately removes this spline and all of its properties from the AlgoMesh project.

5 Advanced Topics

This section contains information on a number of advanced topics warranting further discussion than is provided in the reference material elsewhere in this user guide.

5.1 Avoiding Over-Refinement

The mesh generation process in AlgoMesh guarantees that all input point and line segment geometry is included precisely in the generated mesh. At the same time, it attempts to produce quality mesh cells in all areas of the mesh, e.g. triangles that are not too thin and narrow. As a result, it is particularly important that your input geometry is *clean*; that is, that it does not contain areas in which independent points and line segments are very close to one another. If the distance between any geometric features is too small, over-refinement may occur, and in some cases this may even cause the optimisation method to appear to take forever to generate a mesh.

For example, consider the mesh shown in Figure 152.

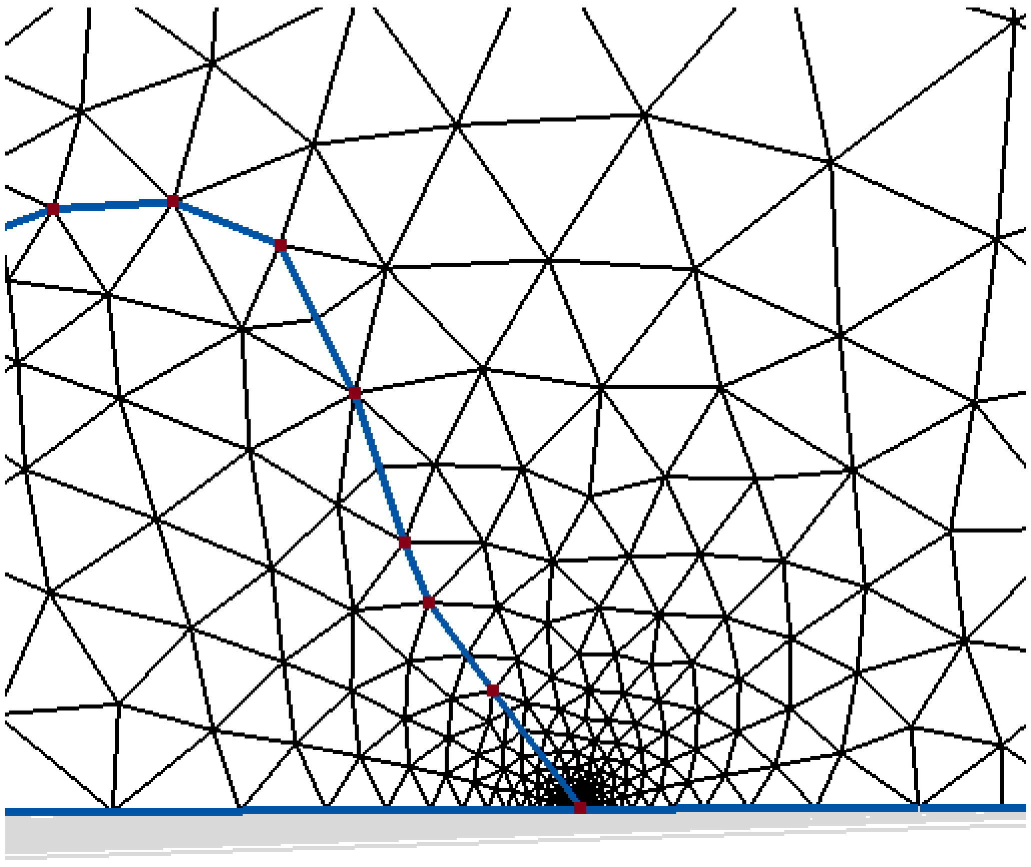


Figure 152: Over-refinement at a join between a river polyline and the model boundary.

In this mesh, there are a large number of very small triangles around the join between a river polyline and the model boundary. Zooming further into the problem region, as in Figure 153, shows the cause of the problem: the endpoint of the river polyline lies slightly apart from the boundary polygon. This forces AlgoMesh to create smaller and smaller elements around the endpoint, until each of the elements is a reasonable shape. Perhaps the endpoint of the river polyline was actually supposed to lie exactly on the polygon, but limited floating point number precision often makes it impossible to ensure this exactly - and there may be differences between software implementations (e.g. between AlgoMesh and the GIS package in which the geometry was initially created) in how floating-point numbers are rounded.

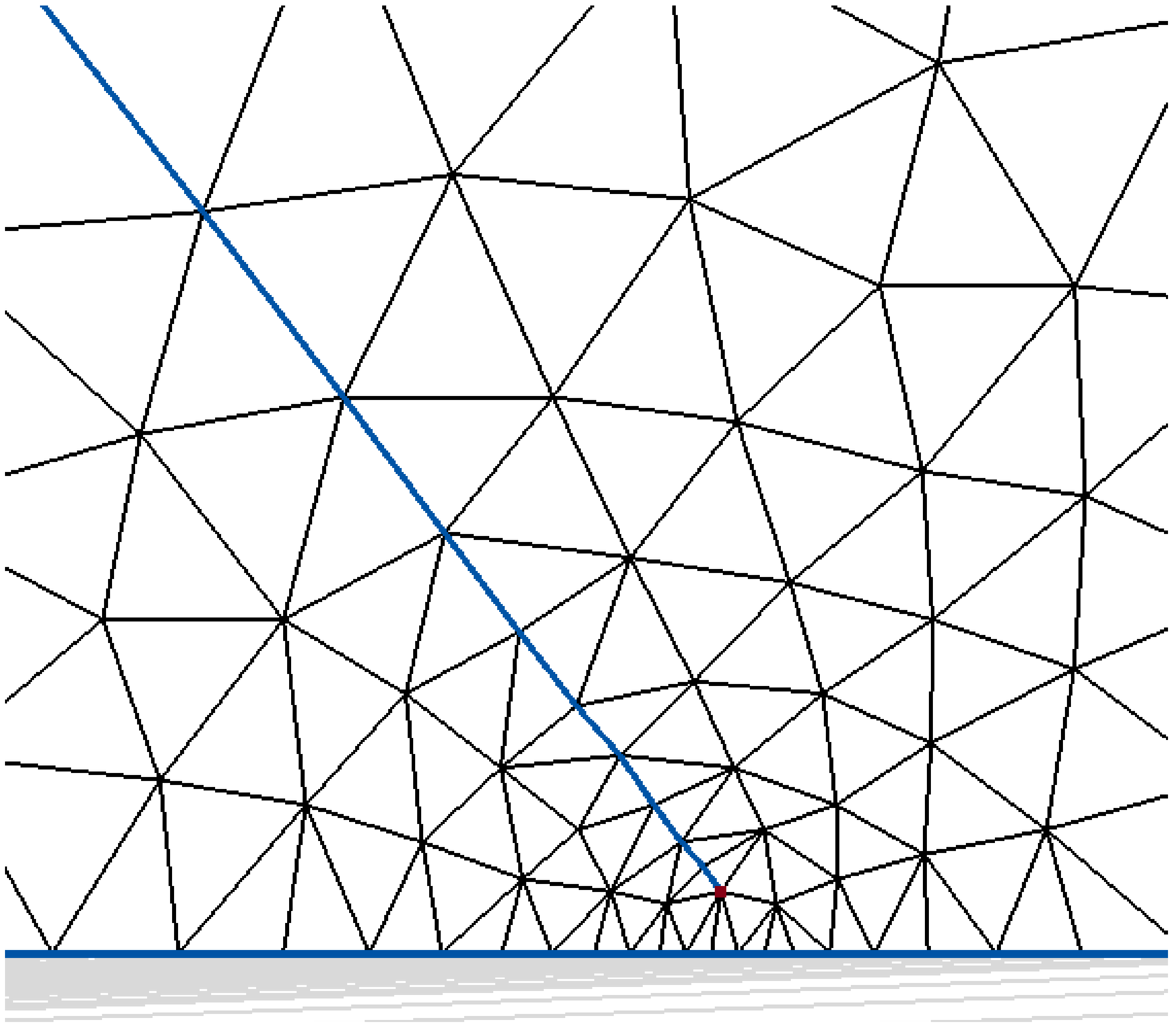


Figure 153: The endpoint of a river polyline lying on the model boundary, but not snapped to a common vertex, may cause over-refinement.

This problem is not limited to model boundaries. It may also occur at junctions between streamlines, for example, or even between two groundwater well locations that are much closer to one another than the desired cell size for that area of the model.

To avoid this problem, you should clean up your input geometry prior to generating a mesh in AlgoMesh, by doing one of the following:

1. Where two polylines (or a polyline and a polygon) meet, as in the above example, ensure that there is a vertex in exactly the same location on both polylines at their point of intersection. In the case above, you would need an additional vertex on the boundary polygon, and the endpoint of the river polyline should be snapped precisely to this vertex in your GIS software. This guarantees that both polylines contain the same point, and stops AlgoMesh from trying to create tiny triangles between the endpoint and the line segment.
2. Alternatively, where two polylines or a polyline and a polygon meet, trim one polyline (the river polyline in the above example) so that it ends a reasonable distance away from the other polyline - at least the desired triangle edge length away. This will typically create enough space to avoid tiny triangles between the point and the line.
3. Where a single constrained point (e.g. groundwater well) lies too close to a polyline, consider

removing the lone point and adding it as a vertex on the polyline instead - or, if the exact location is not critical, move the point further away from the polyline (at least the desired triangle edge length away).

4. Where two constrained points are too close together, consider amalgamating them into a single point. Any boundary conditions (e.g. pumping wells) to be associated with the points in your model can both be added to the same cell or node.

Applying option 2 to the example above eliminates the over-refinement, leading to a much more useful mesh, as seen in Figure 154.

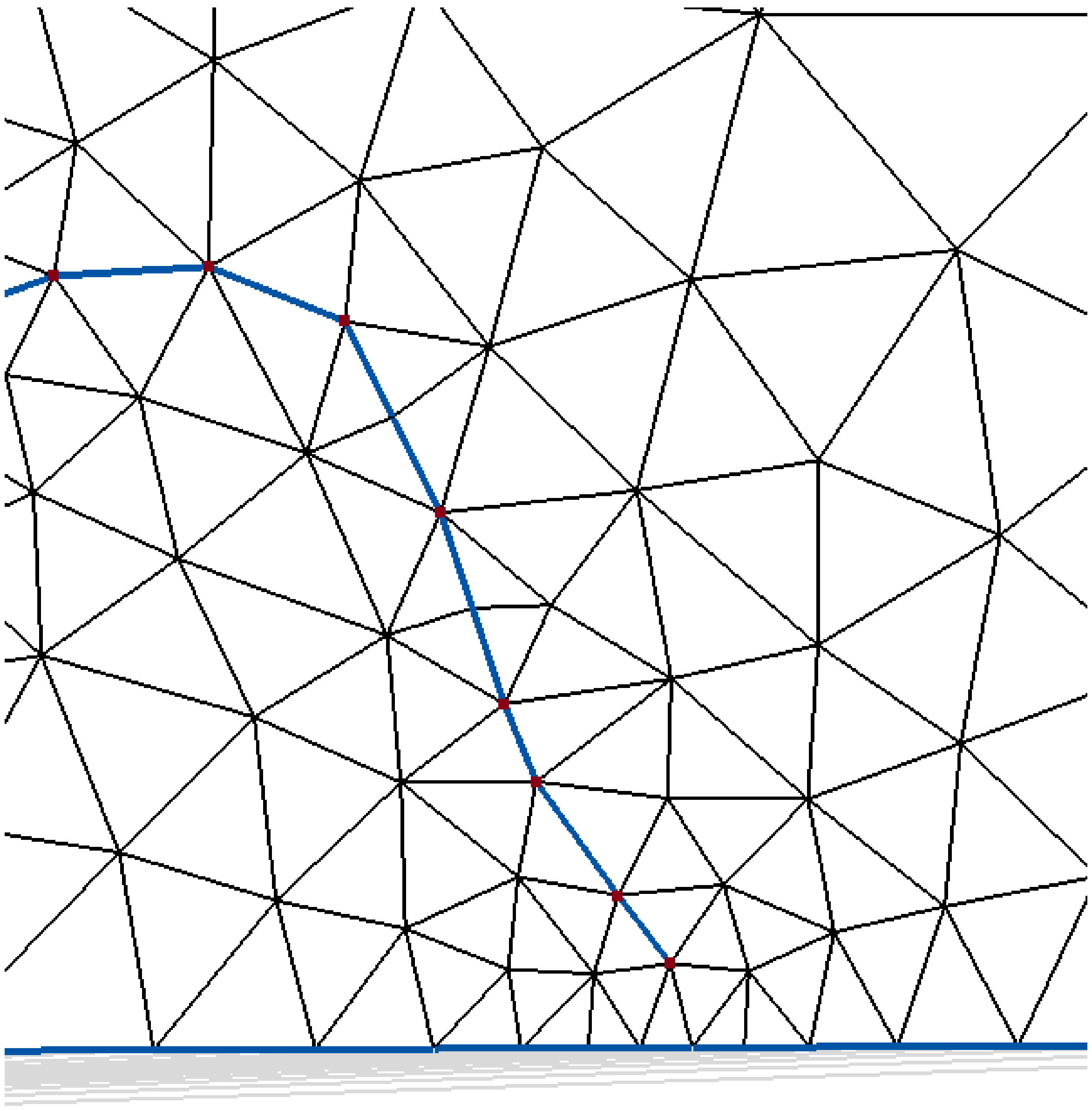


Figure 154: Trimming the endpoint of the river polyline and regenerating the mesh eliminates the over-refinement.

5.2 Edge Constraint Flags and No Refinement Areas

AlgoMesh has several advanced settings that may be used to restrict the mesh generation process in certain areas, and to control how Voronoi cells are generated along constrained edges.

Two of these settings are made available as *edge constraint flags*:

- **The "no split" edge constraint flag:** A constrained edge (line segment) in the mesh with the no split constraint flag set will never be split - have new vertices added - during the mesh generation process. By default, this flag is off. This is not recommended for use in normal meshing, as it may result in very narrow triangles (and for Voronoi grids, it may result in concave cells) but does provide a way to force AlgoMesh to leave certain input geometry untouched. It is used by the Structured Sub-Grid Generation feature. Figure 155 shows an example.

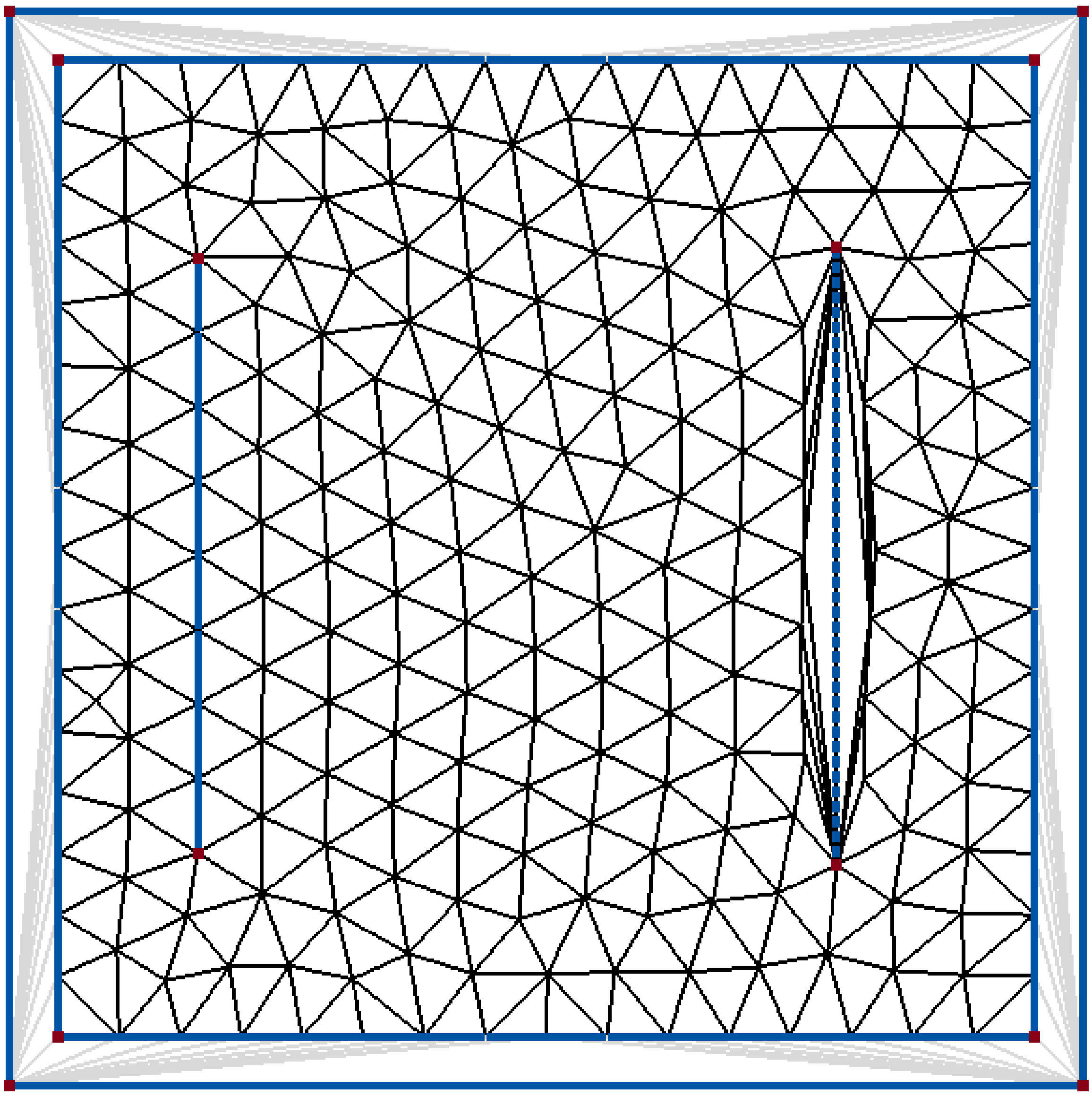


Figure 155: On the left, a regular edge constraint is shown, allowing new mesh vertices to be generated as needed for quality meshing. On the right, an edge constraint with the "no split" flag is shown - as a dashed line - which stops the mesh generator from generating vertices along its length.

- **The "no Voronoi divide" edge constraint flag**: When this flag is on for an edge constraint, Voronoi cells are generated centred at each vertex of the edge such that their shared edge is perpendicular to the edge constraint. When it is off, the generated Voronoi cells are divided along the line of the edge. By default, this flag is on. It may be useful to turn this flag off when a straight line of cell edges is needed in a Voronoi grid - for example a fault line or a man-made structure. Be aware, however, that the perpendicularity property of flow lines between adjacent Voronoi cells, as discussed in the Grid Generation with AlgoMesh section, is not guaranteed for these cells. For Voronoi cells that are cut in this way, the centroid is used as cell centre instead of the associated Delaunay triangulation vertex. Figure 156 shows an example of Voronoi cells produced around a constrained edge with the "no Voronoi divide" flag turned off. The figure has edge constraint visibility turned off to allow the cut Voronoi cells to be seen, but when the edge constraints are shown, constraints with the "no Voronoi divide" flag turned on are coloured blue, and constraints with the "no Voronoi divide" flag turned off are coloured green. Note that all Voronoi cells along the borders of a model are cut (where there is no mesh on the other side of the edge), regardless of whether this edge constraint flag is turned on or off.

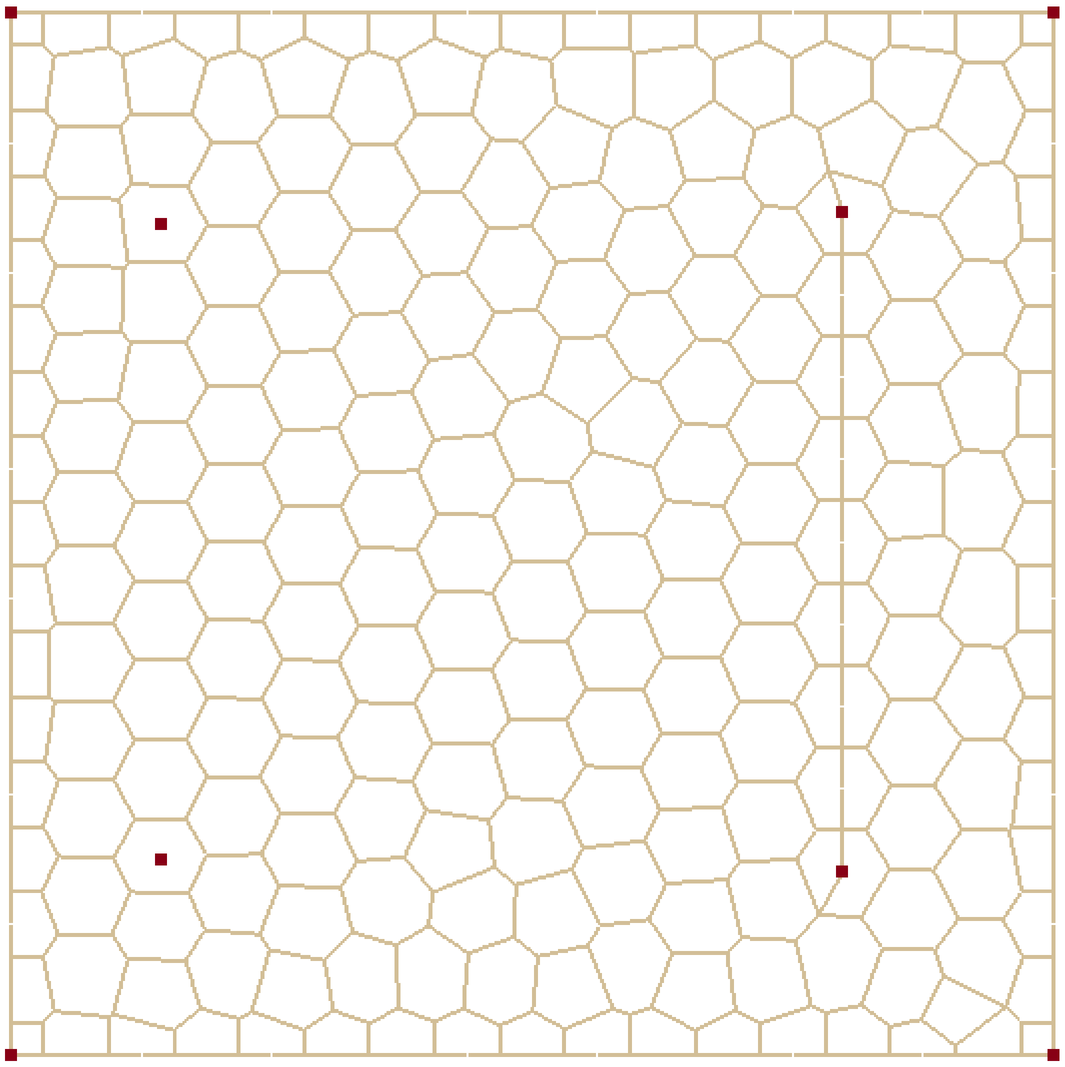


Figure 156: Between the two constrained points on the left, running vertically, a regular edge constraint is present, resulting in whole Voronoi cells aligned along the edge's direction. On the right, an edge constraint with the "no Voronoi divide" flag turned off is present, resulting in each of the Voronoi cells running along the edge being cut into two cells.

Both of these flags may be active in either state on an edge constraint, e.g. it is possible to turn on the "no split" flag and turn off the "no Voronoi divide" flag for the same edge, if desired.

The edge constraint flags may be set for new edge constraints created manually in Edit Mesh mode by using the options in the **Edit->Flags for New Edge Constraint** menu. For splines, edge constraint flags may be set in the **Spline Properties** window.

In addition to the edge constraint flags, there is a mappable mesh variable called **No Refinement Area** that stops the mesh generator from refining any mesh cells that lie inside a certain area - wherever it has a non-zero value mapped. It is used by the Structured Sub-Grid Generation feature in conjunction with the "no split" edge constraint flag to stop the mesh generator from touching an area of input geometry that has been constructed to precisely define a structured sub-grid. Like the "no split" edge constraint flag, it is not recommended for use in normal meshing, as it restricts the mesh generator's ability to produce quality mesh cells. Figure 157 shows the effect of placing a no refinement area around a series of no-split edges.

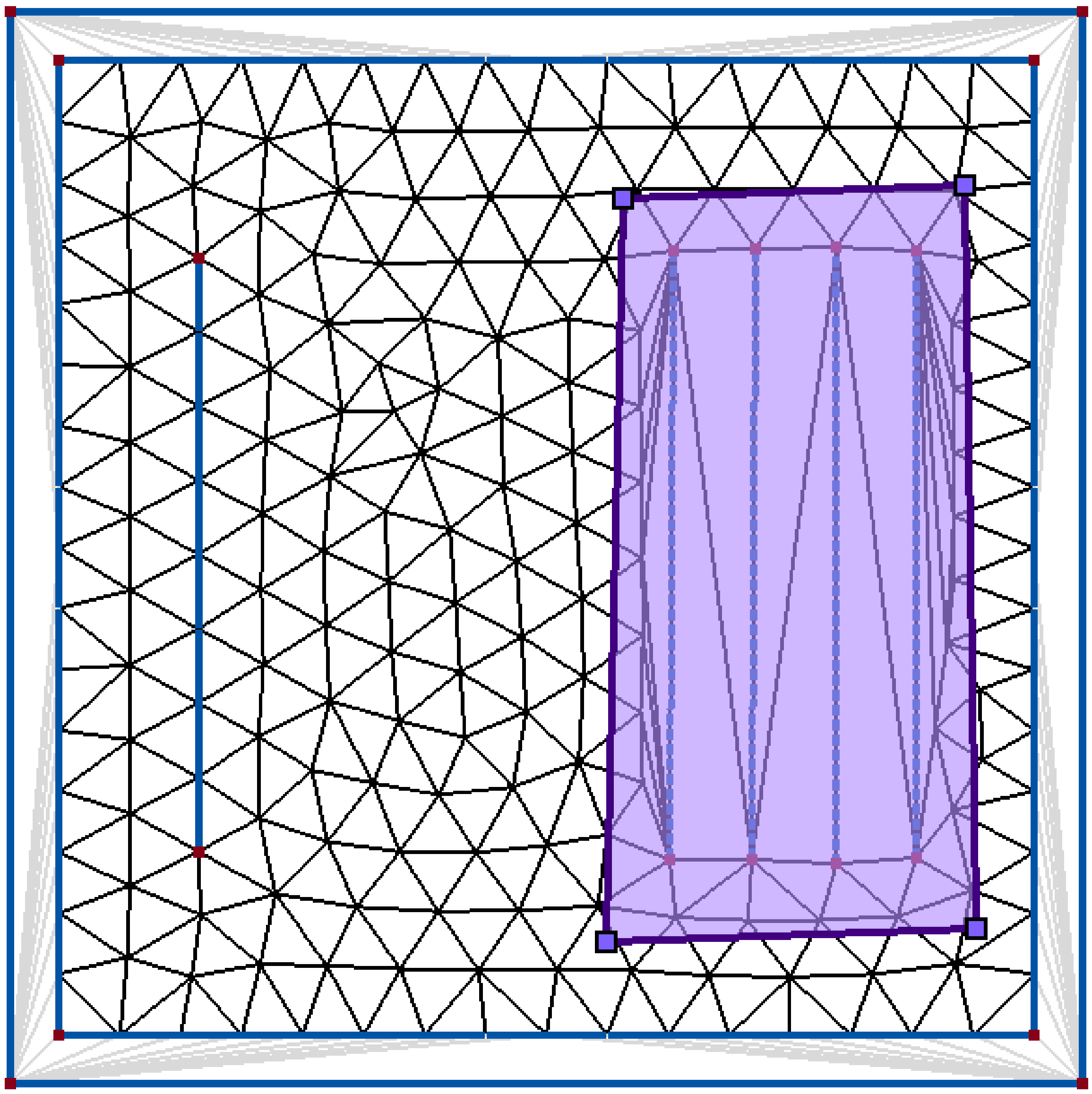


Figure 157: The effect of specifying a no refinement area, in conjunction with several "no split" edges.

5.3 Lateral Connection Groups

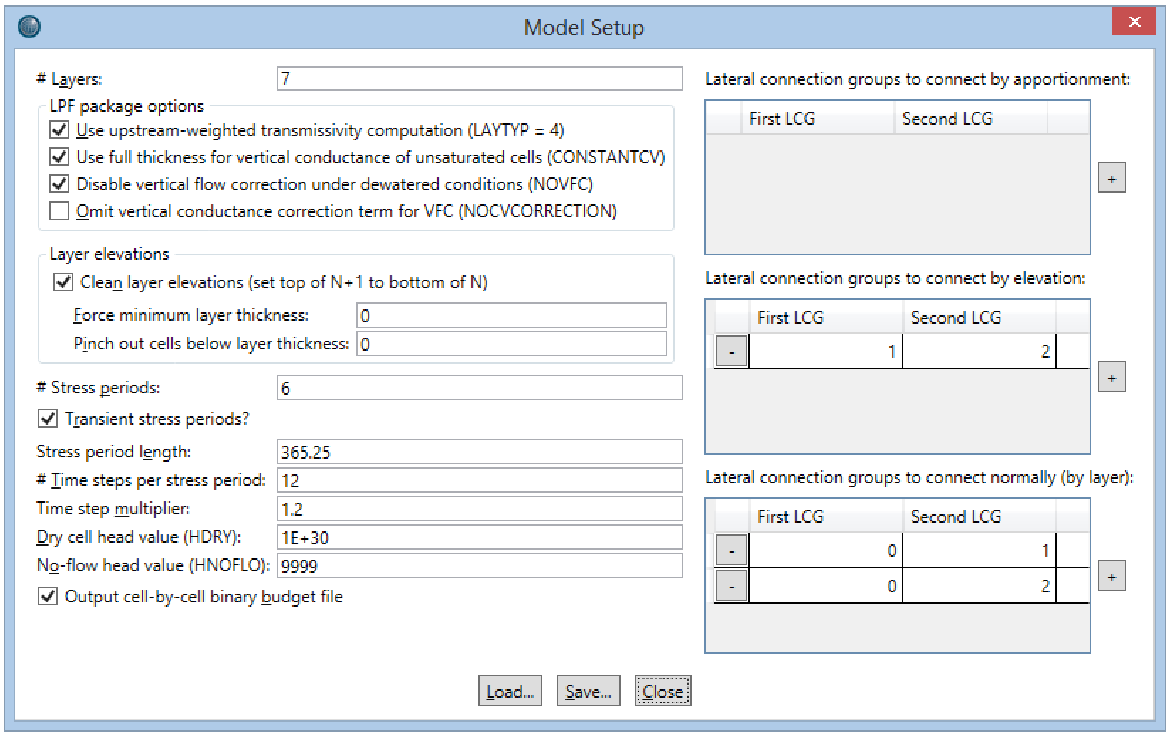
Lateral Connection Groups (LCGs) may be used to control the lateral connections that AlgoMesh creates in the MODFLOW-USG unstructured discretisation (DISU) file.

There are three main scenarios for which LCGs are useful:

1. **Vertical sub-discretisation (nesting):** This is where multiple stacked layers in a spatial area of the model are made to connect to a single regional layer at the boundary of the spatial area.
2. **Cross-strata connections at vertical faults**: Lateral connections may be created across different layers to simulate flow between different geological units that are shifted along a vertical plane from one side of the fault to the other.
3. **Flow barriers (walls)**: If you need to model a scenario where flow should be entirely blocked across cell boundaries, you can use LCGs to stop lateral connections being made between certain sets of cells.

Lateral Connection Groups are specified by mapping any property (from a TIN, polygon set or table) to AlgoMesh's **Lateral Connection Group** property. All LCG scenarios involve adjusting connectivity across the boundary between a pair of different LCG values. The default LCG value for all unmapped cells is 0. Cells with the same LCG value that are adjacent in the two-dimensional mesh will be connected by layer as normal, i.e. a lateral flow connection is made only between cells in the same layer. Cells with different LCG values will not be connected laterally at all, unless specified explicitly in the **Model Setup** window.

For example, to set up connections across a vertical fault, an LCG of 1 may be set for cells on one side of the fault, and an LCG of 2 for cells on the other side. The Model Setup window may then be used to specify that connections between cells of LCG 1 and 2 should be determined by the elevation overlap of the cells, as in Figure 158. In this scenario, we also need to tell AlgoMesh to make normal "by layer" connections between adjacent cells of LCGs 0 and 1, and between adjacent cells of LCGs 0 and 2, so that the fault region is connected as normal to the rest of the model.


Model Setup

Layers: 7

LPF package options
- [x] Use upstream-weighted transmissivity computation (LAYTYP = 4)
- [x] Use full thickness for vertical conductance of unsaturated cells (CONSTANTCV)
- [x] Disable vertical flow correction under dewatered conditions (NOVFC)
- [ ] Omit vertical conductance correction term for VFC (NOCVCORRECTION)

Layer elevations
- [x] Clean layer elevations (set top of N+1 to bottom of N)

Force minimum layer thickness: 0
Pinch out cells below layer thickness: 0

Stress periods: 6
- [x] Transient stress periods?

Stress period length: 365.25
Time steps per stress period: 12
Time step multiplier: 1.2
Dry cell head value (HDRY): 1E+30
No-flow head value (HNOFLO): 9999
- [x] Output cell-by-cell binary budget file

Lateral connection groups to connect by apportionment:

First LCG	Second LCG

Lateral connection groups to connect by elevation:

First LCG	Second LCG
1	2

Lateral connection groups to connect normally (by layer):

First LCG	Second LCG
0	1
0	2

Figure 158: The Model Setup window allows you tell AlgoMesh how to connect each pair of lateral connection groups. This example connects LCGs 1 and 2 by elevation to simulate flow across different layers at a connected vertical fault. Each of these LCGs is connected to the regular model cells normally with a "by layer" connection between LCG pair 0 and 1 and LCG pair 0 and 2.

Note in the Model Setup window that in addition to the normal "by layer" connection method, there are two connection methods available: connection by apportionment and connection by elevation.

Connection by Apportionment

Connection by apportionment is used to allow single geological units to be subdivided into multiple layers in a certain area of a model (**LCG scenario #1**).

Consider the scenario depicted in Figure 159 – a side-on view showing two vertical columns of model cells associated with two (plan view-) adjacent mesh cells.

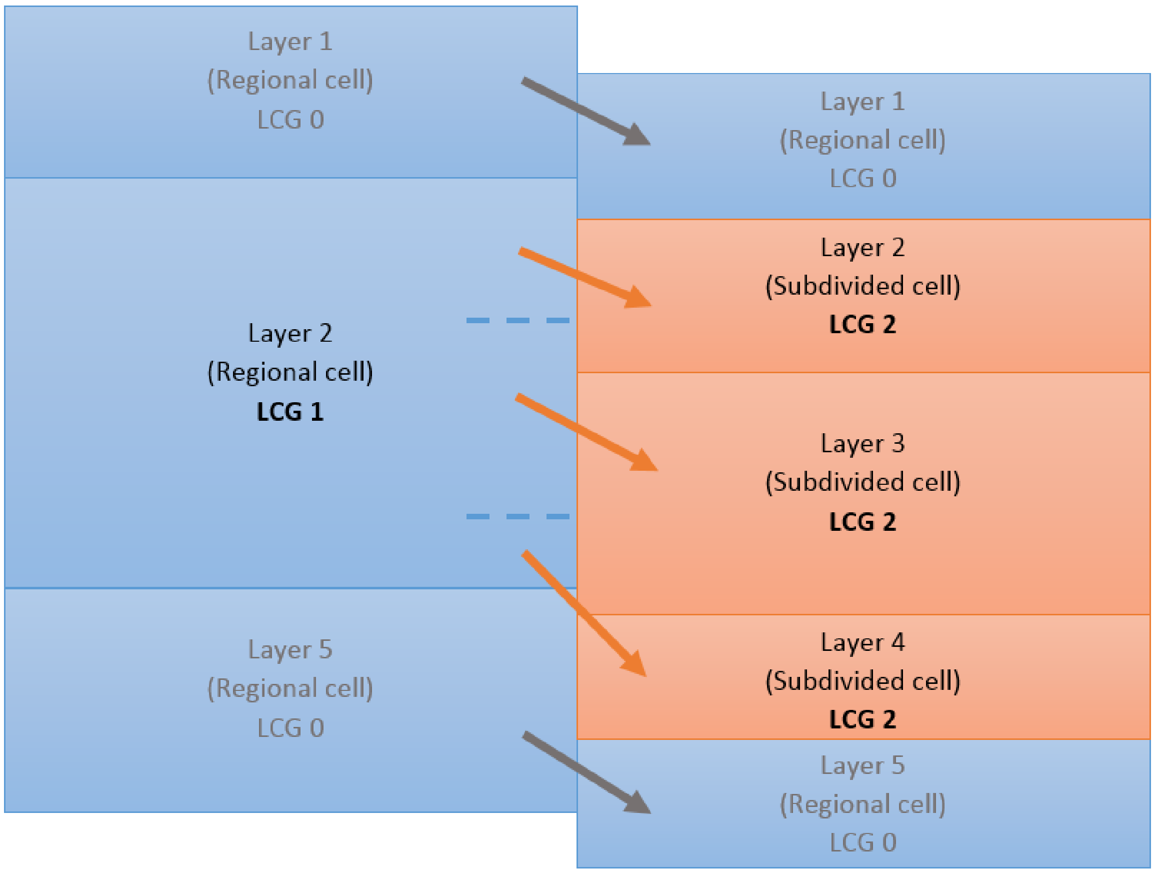
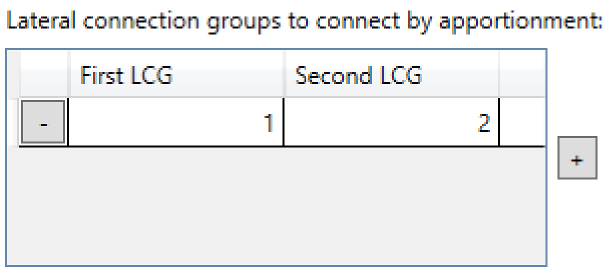


Figure 159: An example of a vertical nesting scenario, implemented using LCGs with connection by apportionment.

In this example, we have three layers regionally, but regional layer 2 is vertically subdivided into layers 2, 3 and 4 within a confined sub-area of interest in the model. Outside of the confined area (regionally), cells in layers 3 and 4 are removed from the model with an Active Cell Area value of 0. We have an LCG of 1 for the regional cells in layer 2, and an LCG of 2 for all cells in the subdivided area, layers 2, 3 and 4, and have added the LCG pair (1, 2) to the list of “Lateral connection groups to connect by apportionment” in AlgoMesh’s Model Setup window (see Figure 160). All cells in Layers 1 and 5 are left with the default LCG value of 0.


Lateral connection groups to connect by apportionment:

	First LCG	Second LCG
-	1	2

Figure 160: The LCG pair connection setup for the vertical nesting example.

The cells with an LCG of 0 are laterally connected as per traditional MODFLOW connections, i.e. they connect only with the adjacent cell in the same layer. The cell with LCG 1, however, is laterally connected to all three adjacent cells with LCG 2, because LCG pair (1, 2) is set to create connections by apportionment. The total flow interface area between the LCG 1 cell and all LCG 2 cells will be the same as if the LCG 2 cells were merged into a single cell with their combined total thickness. This total area is apportioned among each of the three connections in proportion to the fraction of total thickness contributed by the associated individual LCG 2 cell.

Note that the absolute elevations of the top and bottom of each cell using the apportionment method are irrelevant, except in calculating thicknesses. As per traditional MODFLOW lateral connections, it does not matter whether there is any elevation overlap between cells or not; all cells with the same LCG in one vertical column are connected to cells in the adjacent column.

An extension of the same process is used when there is more than one cell in both vertical columns; in this case, cells on one side are assigned connections to cells on the other side according to the fraction of total thickness at their top and bottom. Any two cells whose range of top and bottom fractional thicknesses overlap are connected, with the interface area calculated proportionally from total interface area according to the amount of fractional thickness overlap.

In other areas of the model – away from the boundary between regional and vertically-nested areas – there will be only a single LCG value (LCG 1 in the regional layer 2 and LCG 2 in the nested area). As adjacent cells in these areas have the same LCG, they will be connected only to adjacent cells in the same layer.

If adjacent cells are given different LCGs, and no pair of these LCGs appears in the “by apportionment”, “by elevation” or "by layer" connection lists, AlgoMesh will create no connections between the cells. This is useful for **LCG scenario #3** (entirely blocking flow between cells).

Connection by Elevation

Any (plan view-) adjacent cells whose LCGs appear in the "connection by elevation" list in the Model Setup window will be assessed for connection by elevation overlap. For each cell in this case, all adjacent cells whose top and bottom elevation range overlaps with the cell's own elevation range will be connected to the cell.

The elevation overlap of the cell with each of the adjacent cells satisfying these criteria are summed, and each individual elevation overlap is increased proportionally if necessary, such that the entire thickness of each cell is allocated across the adjacent elevation-overlapping cells. The cell is connected with all of these adjacent cells and the resulting overlap values are used to calculate the effective flow interface area for each.

Be aware that unlike apportionment, this method uses absolute elevations to determine connectivity, which may result in some cells not being connected laterally. For example, the Layer 4 nested cell would not be connected to the Layer 2 regional cell in the aforementioned apportionment connection example, as there is no overlap in their elevations. It would not be connected to the Layer 5 regional cell either, assuming only the LCG pair 1 and 2 appeared in the connection by elevation list, the Layer 5 cell has an LCG of 0.

This method of connection is useful for modelling **LCG scenario #2** (connections across vertical faults).

5.4 Python Scripts

AlgoMesh includes basic support for Python scripting, allowing custom processes to be written that generate mappable values used by AlgoMesh.

To use Python scripting in AlgoMesh, an installation of **Python 3.4** must be present on your system. Only versions 3.4.x of Python are supported; Python 2 and other minor versions of Python 3 (e.g. 3.5.x, 3.3.x) will not be picked up by AlgoMesh. At the time of writing, Python 3.4 is available for download at the official Python website: http://www.python.org/

When AlgoMesh starts, if it finds a Python 3.4 installation on the computer, the **Python Scripts** header in the project explorer on the left-hand side of the AlgoMesh window will be enabled for use. **Right-click** the Python Scripts header and select **New Python Script** to create a new script embedded in the AlgoMesh project. The script's code can be edited by **right-clicking on the script** and choosing **Edit Script**, and it may be mapped to a property just like a property polygon set, table or TIN field.

You may perform whatever processing you like in the script, and return a value to AlgoMesh by setting a variable named **value** to a real number, or to **None** if whatever property is being mapped should not have a value. AlgoMesh calls the script whenever it needs a value for the mapped property at a certain cell or point location. Prior to calling the script, it sets the values of a number of predefined variables, which may be used by the script to find out which cell or location AlgoMesh requires a value for, and to get information about the cells in the current mesh. The variables set by AlgoMesh are listed in the following table:

Variable name	Description
x	The x coordinate of the query point location. For a mesh cell, this will be the x coordinate of the cell centre.
y	The y coordinate of the query point location. For a mesh cell, this will be the y coordinate of the cell centre.
propertyName	An identifier specifying the mesh variable mapped to this script which AlgoMesh is currently querying.
layer	The layer of the query cell, or -1 if not applicable.
globalCellIndex	The global cell index of the query cell, or -1 if not applicable.
stressPeriod	The stress period for which a value is requested.
isContinuous	**True** if the property being queried is continuous and should be present at all areas in the model domain, or **False** if the property is permitted to have no value in some parts of the domain.
cell	An object representing the cell being queried, or **None** if the query is for a point location only.
mesh	An object representing the mesh containing the cell being queried, or **None** if the query is for a point location only.

The **cell** object has the following attributes:

Attribute name	Description
index	The 2D mesh index of the cell.
centre and **center**	A tuple containing the (x, y) location of the cell centre.
area	The 2D calculated area of the cell.
vertexIndices	A list of the mesh vertex indices of the vertices making up the cell's polygon.
vertices	A list of tuples containing the (x, y) locations of the vertices making up the

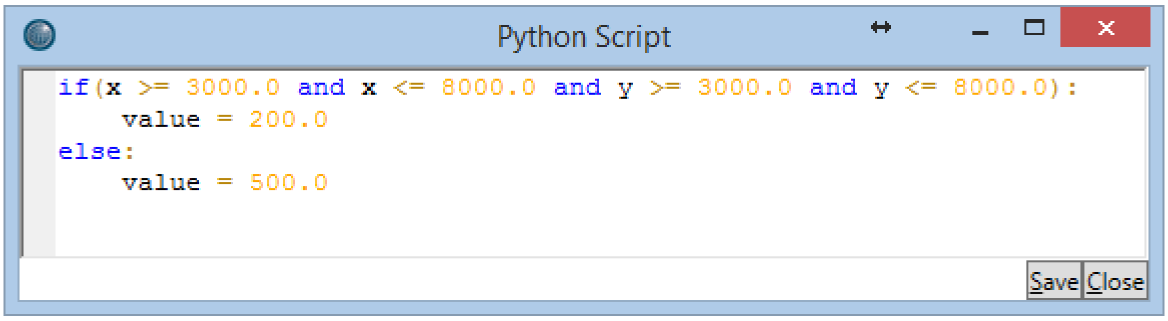
	cell's polygon.
connectedCellIndices	A list of 2D mesh cell indices of cells connected to (neighbouring) this cell.
connectedCellInterfaceLengths	A list containing the lengths of the shared edge between this cell and each neighbouring cell.
connectedCellDistancesToInterfaces	A list containing the perpendicular distance from this cell's centre to the shared edge with each neighbouring cell.

The **mesh** object has the following attributes:

Attribute name	Description
cells	A list of cell objects (as above) representing every cell in the 2D mesh.
vertices	A list of tuples containing the (x, y) locations of all vertices in the mesh.

In the current version of AlgoMesh, there is no way to query other mapped values (e.g. elevations) - but utilising this data could be done through an external data source, like reading from a file in the script.

Consider the script entered into the editor in Figure 161. This script returns a value of 200 if the query location is within the rectangular region between coordinates (3000, 3000) and (8000, 8000), or a value of 500 otherwise.



```
if(x >= 3000.0 and x <= 8000.0 and y >= 3000.0 and y <= 8000.0):
    value = 200.0
else:
    value = 500.0
```

Figure 161: A Python script giving a cell size value of 200 within a rectangular region of the model from coordinates (3000, 3000) to (8000, 8000), and 500 outside of this region.

Mapping this script to *Mesh Cell Edge Length (Field Generation Only)* and running the optimisation process on the sample project **mesh-domain-without-river.amproj** produces the mesh shown in Figure 162. Notice how the cells within the rectangular region coded into the script are refined down to a 200m size, and the cells outside this are generated at a 500m size.

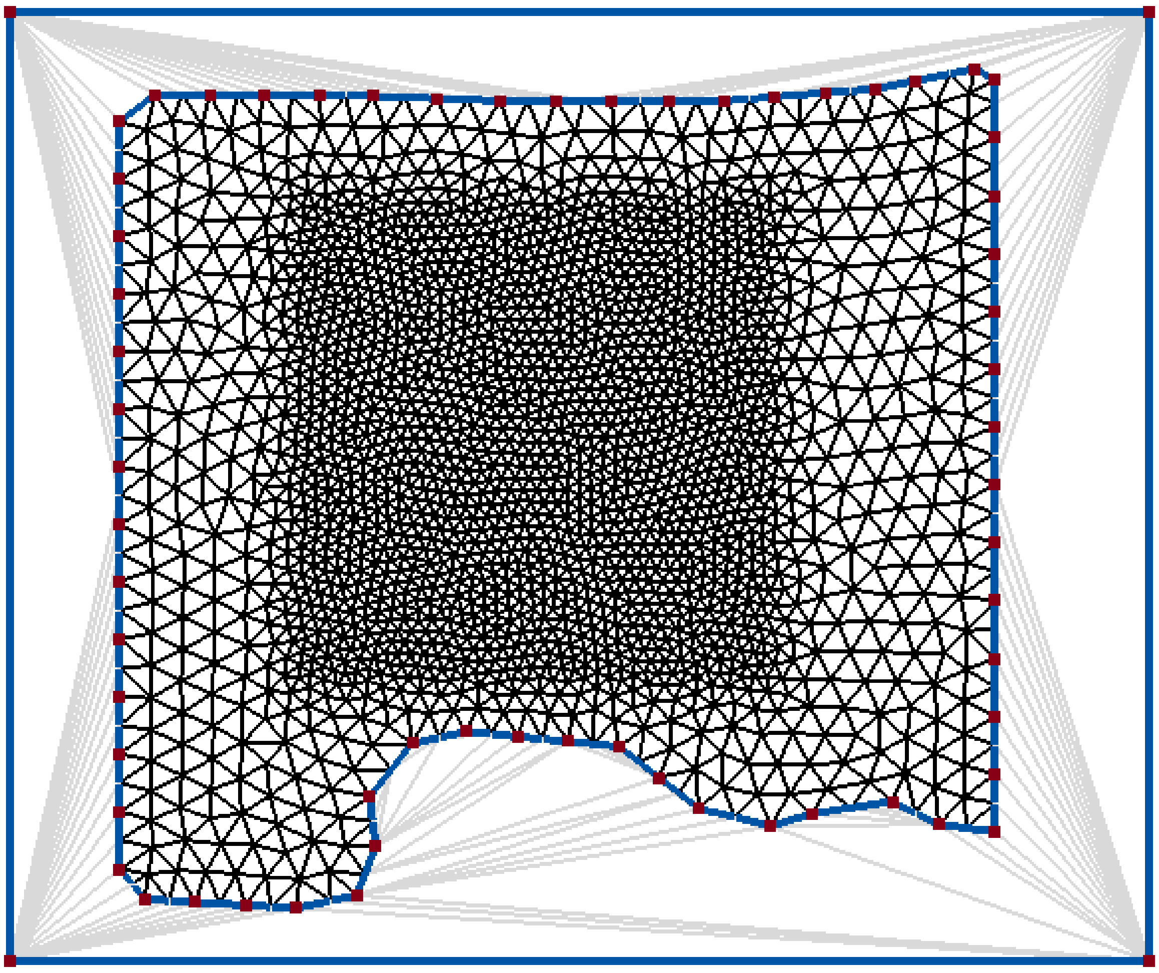


Figure 162: The generated mesh resulting from mapping the python script to Mesh Cell Edge Length (Field Generation Only).

This simple example could have been accomplished more quickly and easily using a property polygon - but the Python script is not limited to specifying a single value within a polygonal area. Any calculation that can be coded into the script may be used to produce an edge length value for AlgoMesh's mesh generator. Beyond this, all other mesh variables are also available for mapping, so you can also write scripts to generate material properties and boundary conditions too, if desired.

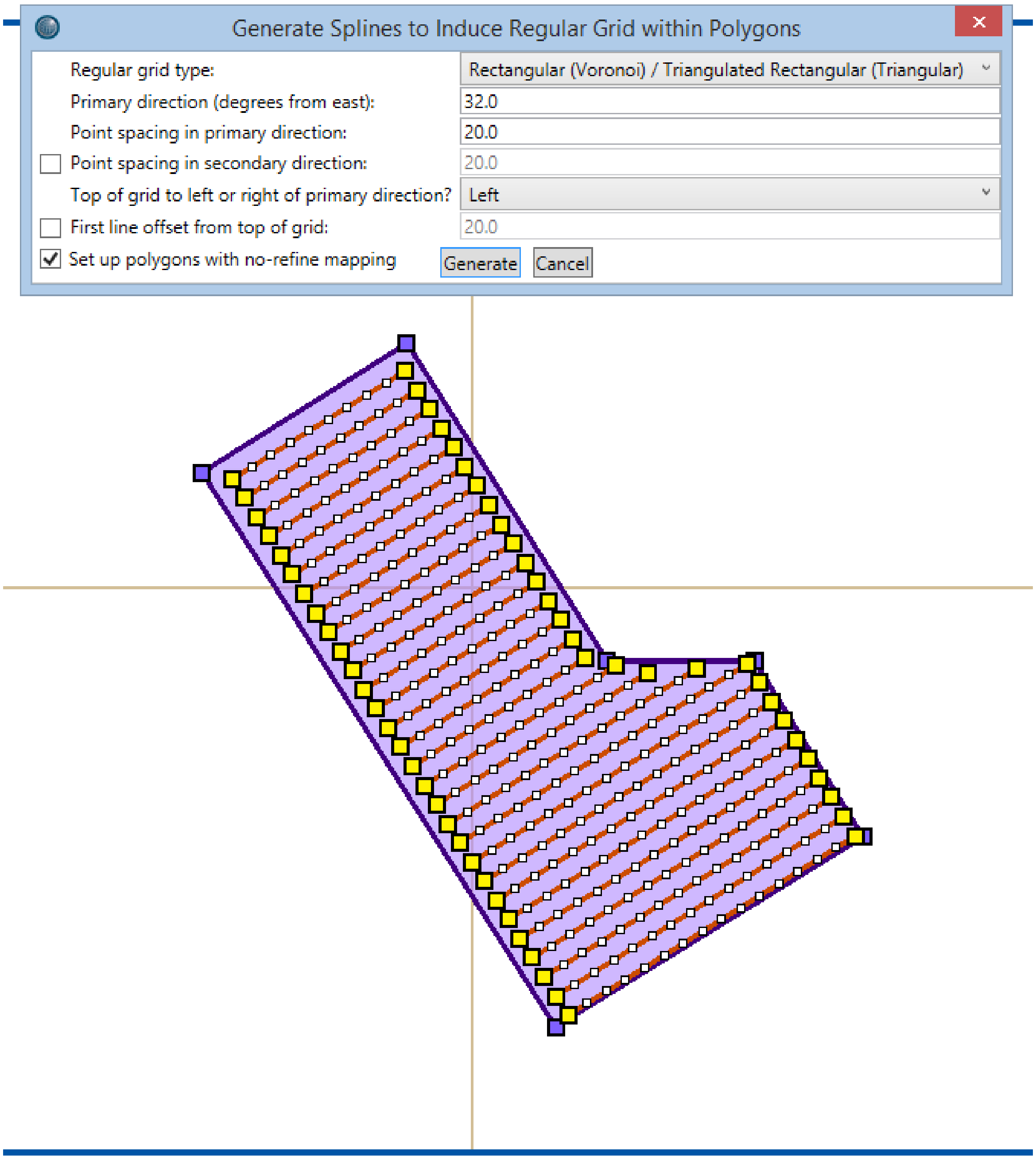
5.5 Structured Sub-Grid Generation

The *structured sub-grid generation* feature of AlgoMesh is a mechanism for generating input geometry such that cells of precise shape and size are constructed within a given polygonal region. It is typically used to construct sub-grids of square or rectangular cells within a Voronoi grid, but may also be used to produce a sub-grid of perfect equilateral triangles in a triangular grid, or perfect hexagonal cells in a Voronoi grid.

This feature is used by creating or importing a property polygon set containing the areas of interest, **right-clicking** on the **polygon set** and selecting **Induce Regular Grid Within Polygons**. This brings up the Generate Splines to Induce Regular Grid within Polygons window.

Settings for the structured grid specify its primary orientation and cell sizing, and allow some fine tuning of the positioning of the grid within the polygons. Once appropriate settings have been entered and the **Generate** button is clicked, AlgoMesh generates a spline set containing a lines within each of the polygons in the set, sampled at appropriate intervals to induce the structured grid cell shapes. These splines have the No Split property and adds a No Refinement Area mapping to the polygon set, to ensure that the constructed cell shapes are not changed by the mesh generation process (See the Edge Constraint Flags and No Refinement Areas section).

Figure 163 shows the structured sub-grid generation window with settings to produce a sub-grid with 20m square Voronoi cells whose flow faces are oriented along a 32 degree angle counter-clockwise from east, and the splines that are generated from these settings.


Generate Splines to Induce Regular Grid within Polygons

Regular grid type:	Rectangular (Voronoi) / Triangulated Rectangular (Triangular)
Primary direction (degrees from east):	32.0
Point spacing in primary direction:	20.0
☐ Point spacing in secondary direction:	20.0
Top of grid to left or right of primary direction?	Left
☐ First line offset from top of grid:	20.0
☑ Set up polygons with no-refine mapping	

Figure 163: The Generate Splines to Induce Regular Grid within Polygons window and the splines produced as a result.

The result of running the optimisation process to generate a Voronoi mesh from this new input geometry is shown in Figure 164. Note how AlgoMesh grades cells smoothly away from the edges of the sub-grid.

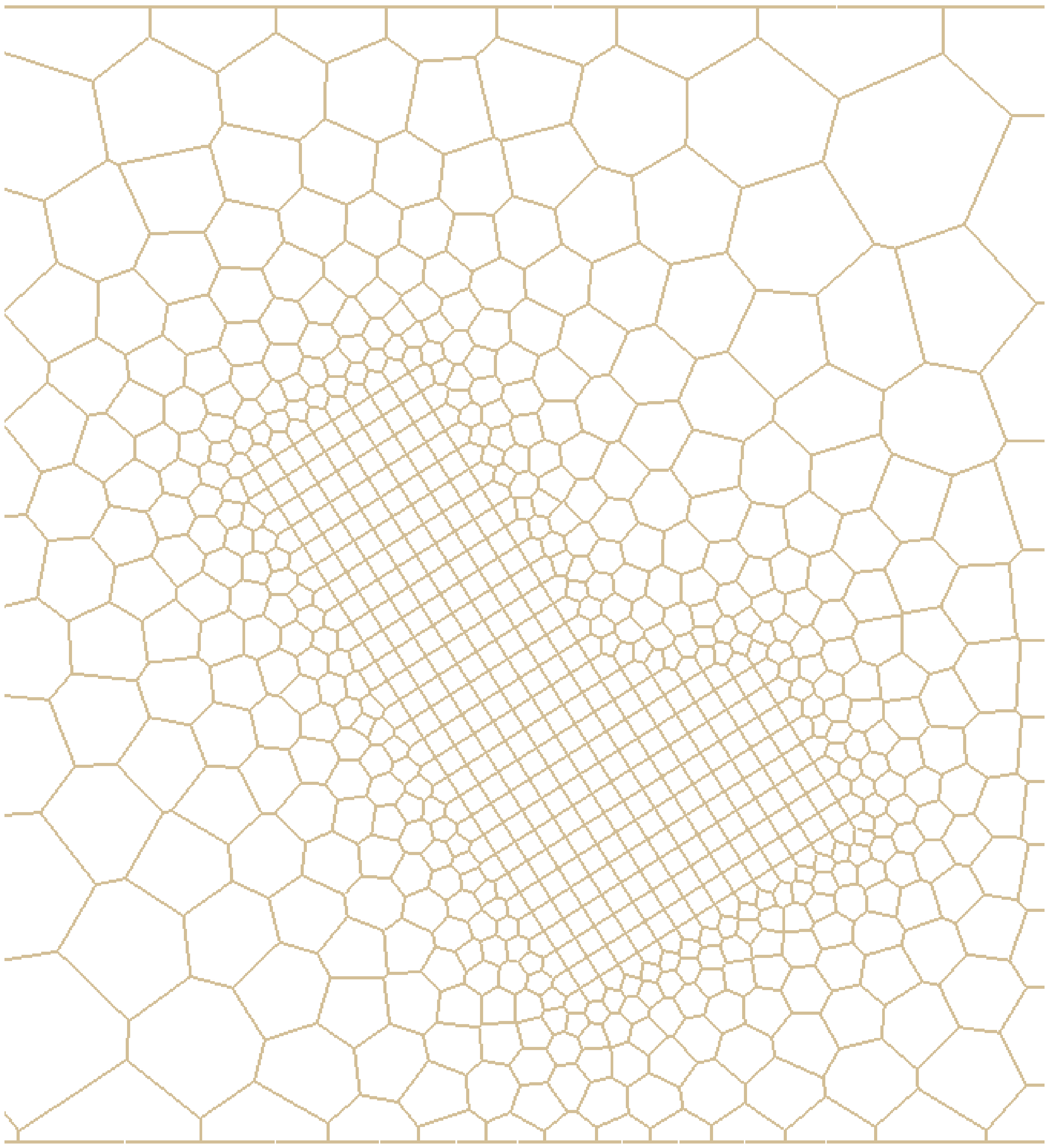


Figure 164: The Voronoi grid generated from the splines and No Refinement Area mapping produced by the structured sub-grid generation method, incorporating an area of 20m square cells within the specified polygonal area.

Rectangular Voronoi cells may optionally be generated instead of squares, but some care must be taken to avoid undesirable cell shapes at the boundaries of the sub-grid. This also applies for some square sub-grids in polygonal regions that have sharp angles. Figure 165 shows an example of this, where the "Point spacing in secondary direction" is changed to 10m to produce 20m x 10m rectangular cells. When the mesh is generated here, it produces poorly-shaped concave Voronoi cells along some of the borders of the sub-grid, as the mesh generator tries to make smaller cells but cannot, due to the no-split edges and no refinement area.

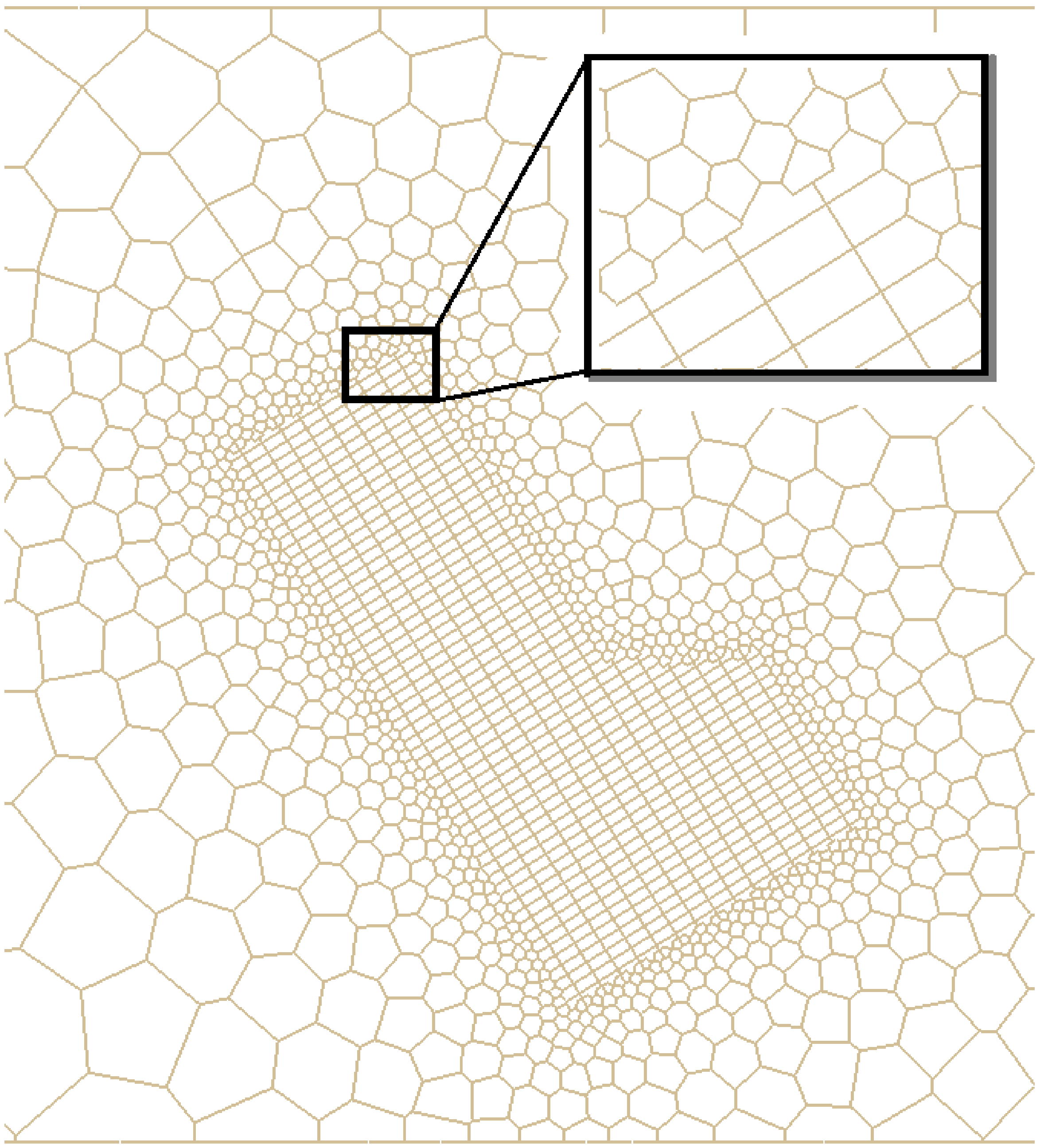
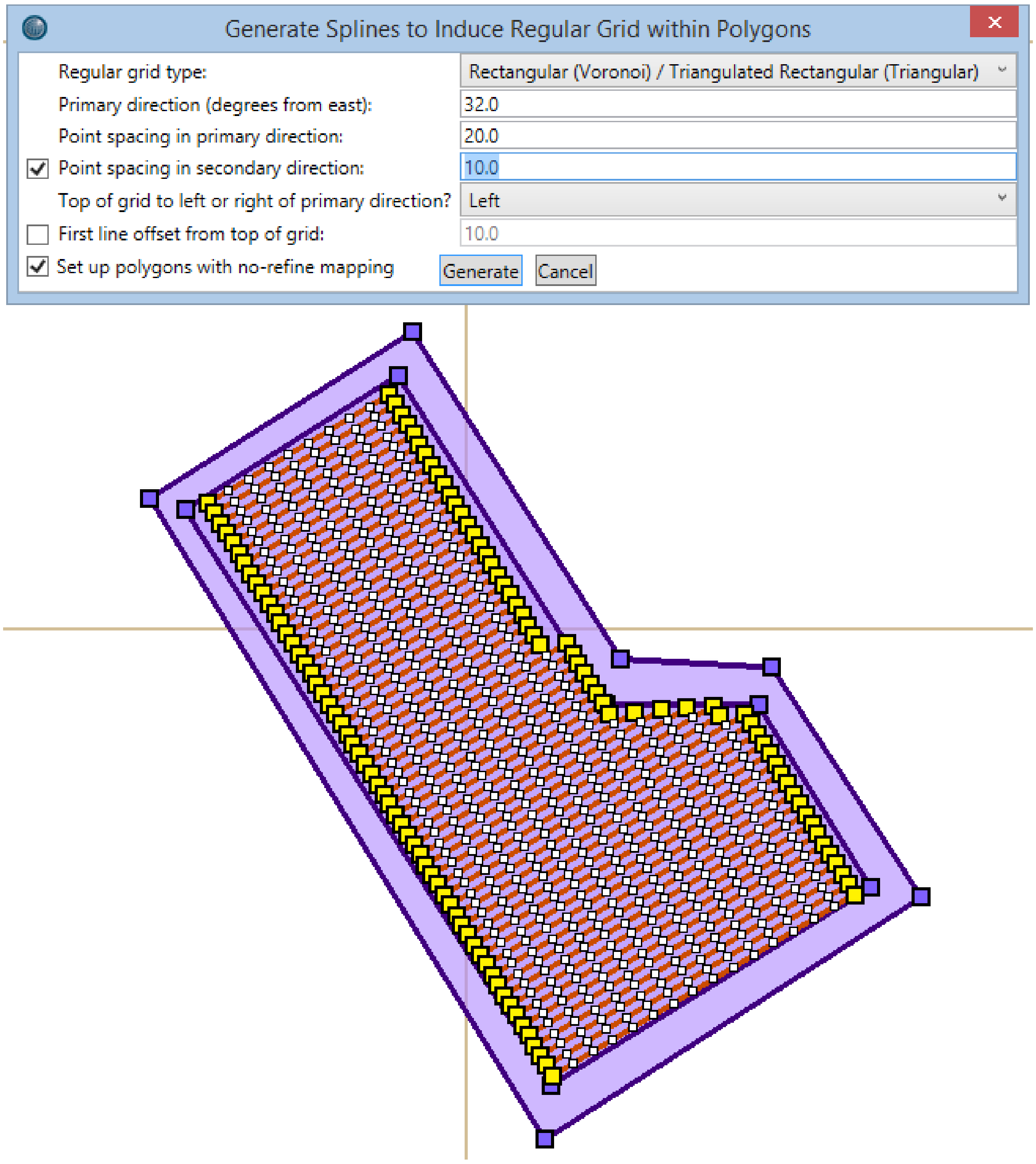


Figure 165: A 20m by 10m rectangular sub-grid. Notice the undesirable concave cells generated along the top edge (zoomed inset).

One way to deal with this is to force AlgoMesh not to produce such small cells around the borders of the polygon. Creating a buffer polygon, as shown in Figure 166, and giving it a property mapped to Mesh Cell Edge Length, allows us to more carefully control the cell sizing around the borders. In this case, a buffer polygon was manually digitised and a value of 30m was mapped to Mesh Cell Edge Length. Note that we do not map to the alternative *Mesh Cell Edge Length (Field Generation Only)* here, as this would only provide a maximum value to the edge length field generation process; instead, we need to override the value used by the optimisation process, which would normally be obtained from the edge length field. Mapping to Mesh Cell Edge Length forces the optimisation process to take our given value of 30m, instead of using the value from the generated edge length field.


Generate Splines to Induce Regular Grid within Polygons

Regular grid type:	Rectangular (Voronoi) / Triangulated Rectangular (Triangular)
Primary direction (degrees from east):	32.0
Point spacing in primary direction:	20.0
☑ Point spacing in secondary direction:	10.0
Top of grid to left or right of primary direction?	Left
☐ First line offset from top of grid:	10.0
☑ Set up polygons with no-refine mapping	

Figure 166: A buffer polygon is created around the main sub-grid polygon to allow mapping of a larger cell size around the sub-grid's border.

Applying the 30m Mesh Cell Edge Length mapping corrects the concave cell problem, as shown in Figure 167, as AlgoMesh stops trying to create cells that are too small at the boundaries of the sub-grid.

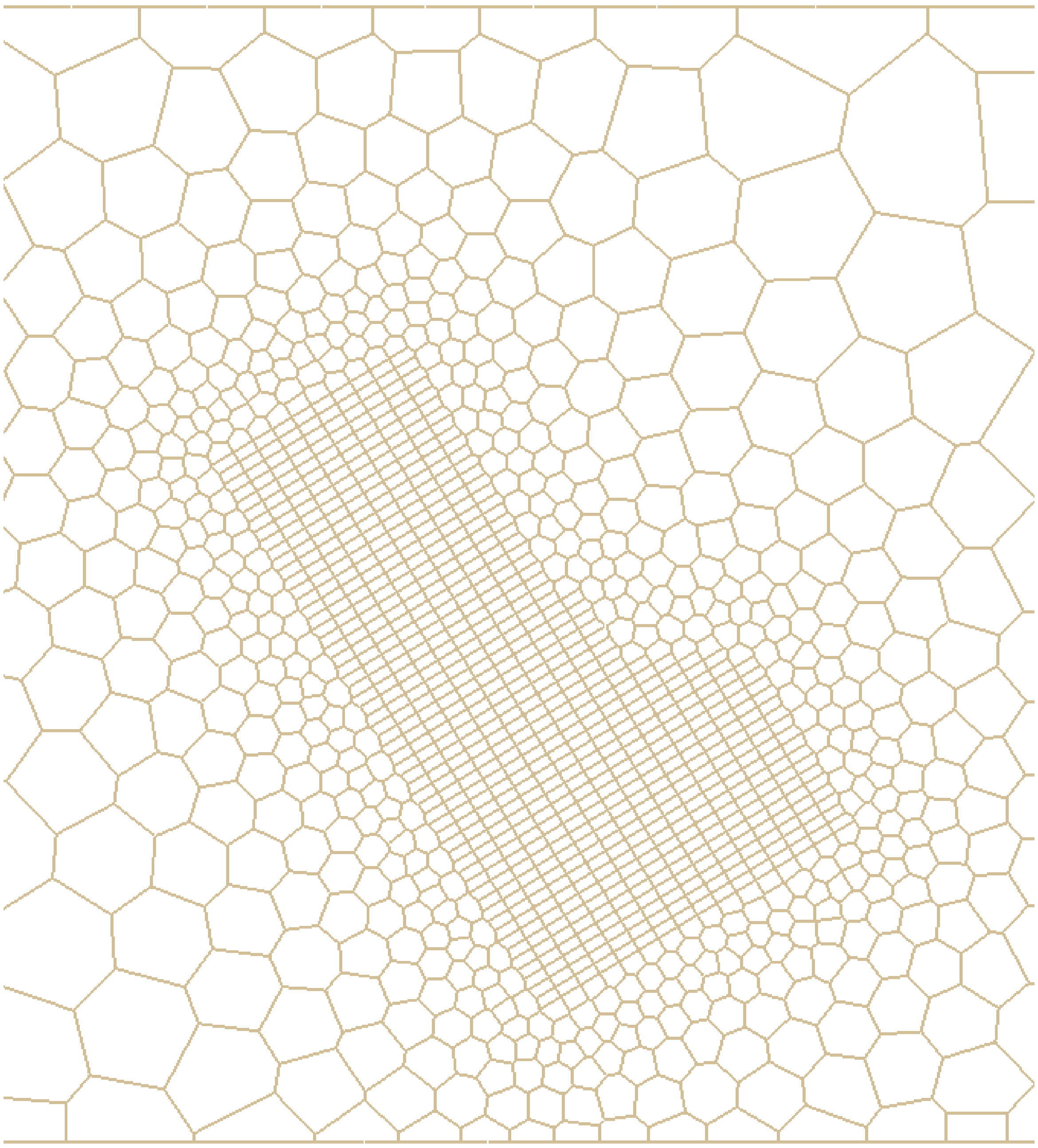


Figure 167: Forcing a larger cell size of 30m around the borders of the sub-grid fixes the concave cells.

For triangular grids, the sub-grid generation feature may instead be used to generate regions of perfect equilateral triangles, as in Figure 168.

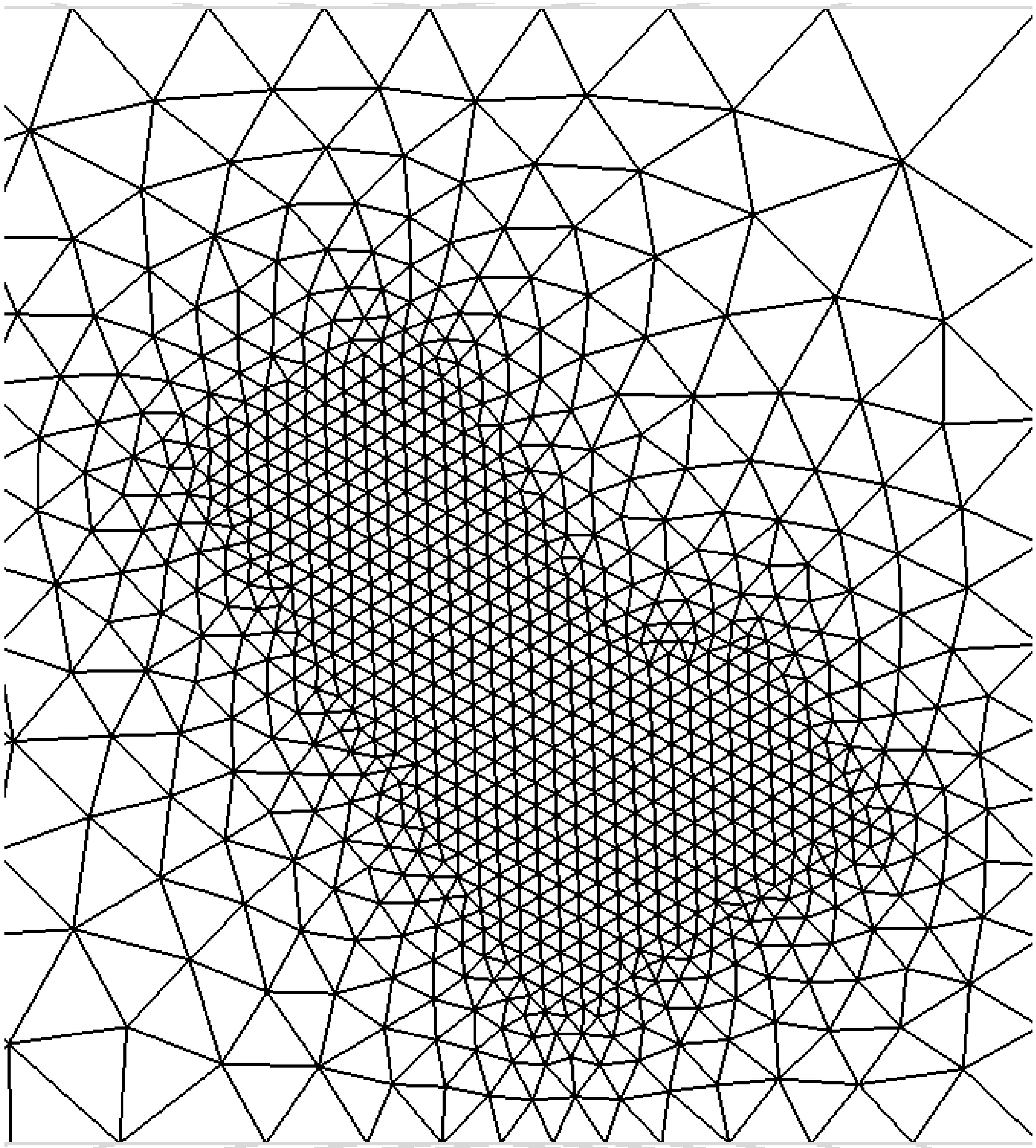


Figure 168: In triangular grids, the structured sub-grid feature can be used to generate areas containing perfect equilateral triangles.

In Voronoi grids, this equates to a perfect hexagonal sub-grid (Figure 169), which may be useful to avoid inaccuracies in MODFLOW-USG's CVFD flow solution between cells in that area.

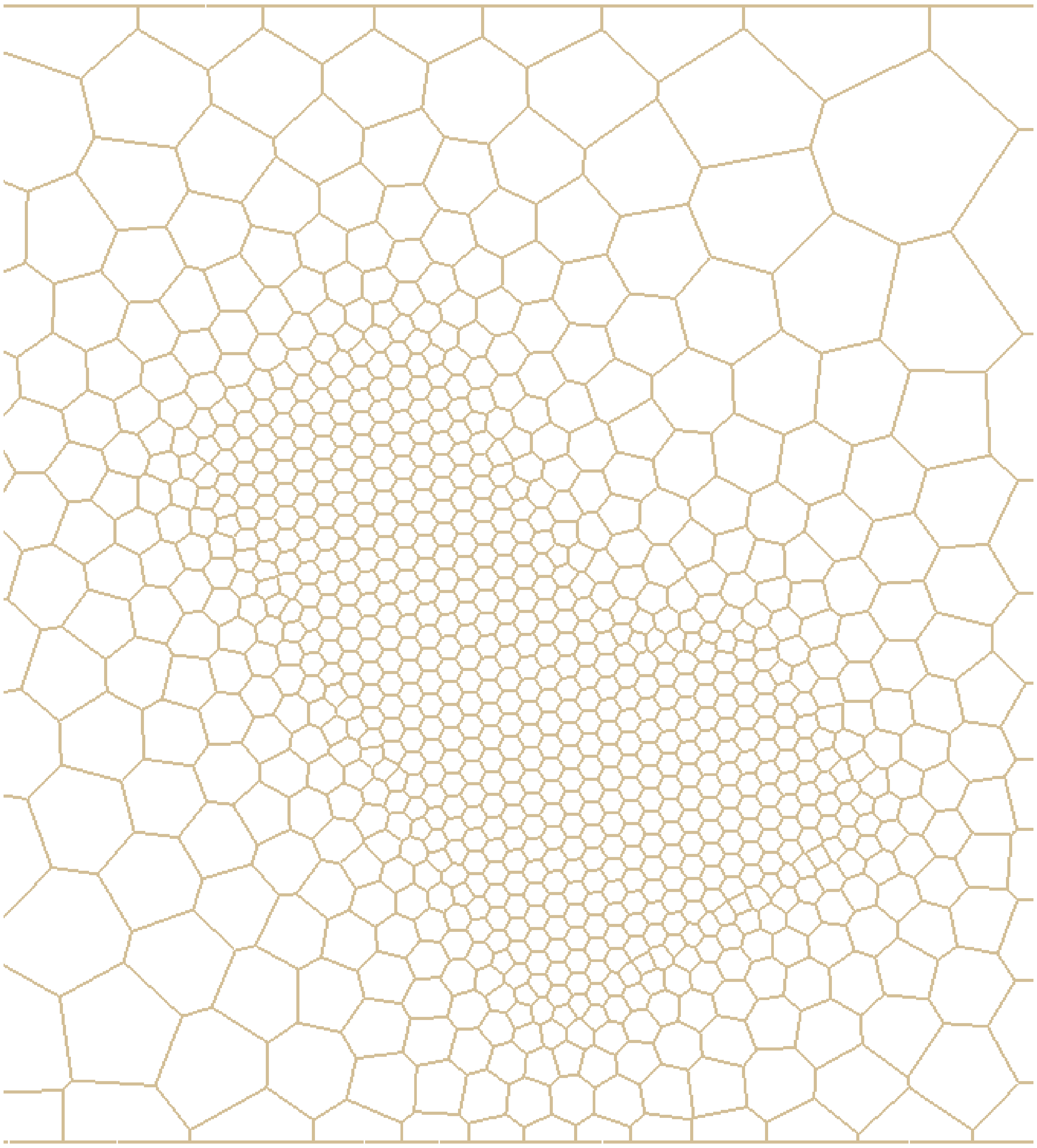


Figure 169: Perfect hexagonal sub-grids can be generated within Voronoi grids.

Note that these sub-grids are only possible in otherwise open areas of a model domain. If other point or linear features overlap the sub-grids, extraneous cells will be added and the structure of the sub-grid will be compromised.

6 File Formats

AlgoMesh makes use of a number of different file formats for storing and retrieving mesh and model information. Some of these are text-based formats which you have the option of inspecting, writing or editing yourself using a text editor or your own custom script, allowing you to customise various aspects of your model outside of AlgoMesh, if desired. Each of these textual formats is described here in its own respective section.

6.1 Mesh Domain Description (.mdd)

The MDD file specifies a two-dimensional domain to be meshed, in the form of points and line segments that must be present in the generated mesh. It may also be used without a full mesh generation process, to assign property values at specified points (elevation, head, hydraulic conductivity, etc.), for interpolation over a mesh domain.

The file is divided into various sections:

- **VERTICES**
  Points that must be included as vertices of the mesh. Each point may optionally have one or more named property values associated with it for interpolation over the domain (e.g. elevation, head, hydraulic conductivity).
- **LINE_SEGMENTS_BY_INDEX**
  Line segments that must be present as triangle edges in the generated mesh, each specified by two indices into the collection of vertices. Polylines and polygons may be formed by multiple connected line segments.
- **LINE_SEGMENTS_BY_COORDINATE**
  Line segments that must be present as triangle edges in the generated mesh, as for LINE_SEGMENTS_BY_INDEX, except each is specified by two pairs of (X, Y) coordinates instead of vertex indices.
- **FILL_ZONES**
  Zones to be marked as inside or outside the domain, specified by a single point within the zone. These may be used to mark “holes” in the mesh, or to eliminate excess mesh elements that are outside the model area. Zones outside the domain will not be subject to mesh refinement, and will not be present in the output mesh.
- **VERTEX_PROPERTY_DEFAULT_VALUES**
  Default values for named vertex properties, which are used at any vertex for which there is no value specified in the VERTICES section.
- **SPLINE_SET**
  A collection of cubic spline curves to be sampled and added as vertices and line segments (triangle edges) in the mesh.

Each section may appear once, multiple times or not at all. A section is initiated with a BEGIN marker and its name, and concluded using an END marker, as in the following example:

```
BEGIN VERTICES
  # ...
  # (Content of VERTICES section)
  # ...
END

BEGIN LINE_SEGMENTS_BY_INDEX
  # ...
  # (Content of LINE_SEGMENTS_BY_INDEX section)
  # ...
END
```

```
# ...
```

The # character is used to denote comments; all characters on a single line including and beyond the # are ignored. Any text between the END marker of one section and the BEGIN marker of another is ignored (as long as it does not start with BEGIN).
Whitespace characters (spaces and tabs) are ignored within and around lines, but certain information must be contained on a single line (e.g. BEGIN <Section Name>, each point within the VERTICES section, each line segment within the line segment sections, etc.). Blank lines are ignored. When multiple tokens are given on a single line (e.g. point X and Y coordinates), they may be separated by spaces, tabs, commas or semicolons. Textual tokens are case-insensitive (e.g. “Begin Vertices” would work as well as “BEGIN VERTICES”).

Detailed descriptions of each section are given under respective headings below.

VERTICES Section

The VERTICES section contains a list of two-dimensional points that must be present in the generated mesh. Each point is specified on a single line by its (X, Y) coordinates, and optionally some attached named property values:

```
BEGIN VERTICES
  <X coord. vertex 0> <Y coord. vertex 0> [FREE]
[<Properties>]
  <X coord. vertex 1> <Y coord. vertex 1> [FREE]
[<Properties>]
  # ...
  <X coord. vertex N> <Y coord. vertex N> [FREE]
[<Properties>]
END
```

Note that the first vertex in the section is considered to have index 0, the second has index 1, and so on. This is important if a LINE_SEGMENTS_BY_INDEX section is used (see the description of this section later for details).

If the optional keyword FREE is present after the vertex coordinates, the vertex is considered to be a free vertex instead of a constrained vertex (constrained is the default). A constrained vertex will never be moved, whereas the location of a free vertex may be altered by the mesh generation process when the Lloyd relaxation procedure is used. Note that any vertex used as an endpoint of a line segment (see the LINE_SEGMENTS_BY_INDEX section below) will never be moved, regardless of whether it is specified as FREE or not.

One or more named properties may be given values at each vertex. This is useful for creating an MDD file describes an auxiliary domain from which interpolated values can be retrieved at given points on another mesh. For example, an elevation may be specified at each vertex, ultimately producing an interpolated spatial map of elevations across the whole domain, which can later be exported as layer elevations in MODFLOW-USG file export. Each property value setting comprises a single word property name, followed by a real number value to be assigned at the vertex:

```
  [<Property name> <Property value>]
```

Property names can be anything, as long as they comprise only non-whitespace and non-separator (comma and semicolon) characters. The keyword FREE is reserved and may not be used as a property name. Any number of properties may be set at a single vertex by including them one after another on the same line. An example of property values at vertices is given at the end of this section.

The VERTICES section in the following example would specify corner points of a 1000x1000 square domain:

```
BEGIN VERTICES
  0    0     # Bottom-left corner
  1000 0     # Bottom-right corner
  0    1000  # Top-left corner
  1000 1000  # Top-right corner
END
```

If multiple VERTICES sections are included in one MDD file, vertex index numbering proceeds from where it left off in the previous VERTICES section, as indicated by the comments in the following example:

```
# --- First vertices section ---
BEGIN VERTICES
  0    0     # Vertex 0
 -1.5  0.33  # Vertex 1
END


# --- Second vertices section ---
BEGIN VERTICES
  5.0 -4.1   # Vertex 2 (continued from previous section)
  2.5  2.0   # Vertex 3
  # ...
END
```

Duplicate vertices may be included, but will be removed by AlgoMesh.

The following example applies elevation values (TOP_Z, BOTTOM_Z) and hydraulic conductivity (Kx) values at the corners of the 1000x1000 square. When used to map these property values onto another mesh, a triangulation-linear interpolation takes place (points are Delaunay triangulated, and the property values are assumed to vary linearly across the plane of each triangle, between the values specified at the triangle’s vertices).

```
BEGIN VERTICES
  0    0     TOP_Z 82.5 BOTTOM_Z 70.0 Kx 10.0
  1000 0     TOP_Z 83.0 BOTTOM_Z 69.5 Kx 10.0
  0    1000  TOP_Z 84.0 BOTTOM_Z 71.0 Kx 10.0
  1000 1000  TOP_Z 81.25 BOTTOM_Z 70.0 Kx 10.0
END
```

If duplicate vertices with differing property values are present, it is undefined which of the property values will be taken. As such, it is recommended that you ensure that property values at identical points are equal.

LINE_SEGMENTS_BY_INDEX Section

The LINE_SEGMENTS_BY_INDEX section contains a list of line segments that must be present as edges in the generated triangular mesh. Each line segment is specified on a single line by a starting and ending vertex index and a number of optional line segment properties. Line segment properties differ from vertex properties, in that they are combinations of pre-defined settings, instead of user-defined values.

Polylines and polygons may be formed simply by dividing them into their component line segments (with each pair of subsequent line segments sharing a vertex). If one or more LINE_SEGMENTS_BY_INDEX sections are present in an MDD file, the same file must contain a VERTICES section.

```
BEGIN LINE_SEGMENTS_BY_INDEX
 # First line segment
 <VSTART> <VEND> [<Line segment properties>]

 # Second line segment
 <VSTART> <VEND> [<Line segment properties>]

 # ...

 # Last line segment
 <VSTART> <VEND> [<Line segment properties>]
END
```

VSTART is the 0-based index of the starting vertex of the line segment, and **VEND** is the 0-based index of the ending vertex (see VERTICES section description for details on vertex indexing).

The **Line segment properties** entry comprises one or more of the following settings:

- **EVEN_SPACING <Maximum distance between points>**
  Creates additional points along this line segment, if necessary, such that adjacent points along the segment are no more than the specified maximum distance apart. Additional points are distributed evenly along the line segment. If the line segment’s length does not

divide evenly by the given maximum distance, the next smallest value will be chosen to ensure evenly spaced points. If the line segment’s length is less than the given maximum distance, no points will be added. Note that additional points may be created during mesh generation regardless, but this property may be used to force extra density along line segments.

- **EXACT_SPACING <Maximum distance between points>**
  An alternative to EVEN_SPACING for creating additional points along the line segment. Additional points will be created exactly the given distance apart along the segment, starting from VSTART. If the line segment’s length does not divide evenly by the given maximum distance, there will be a smaller distance between the last added point and VEND.
- **LHS_OUTSIDE**
  Indicates that the region on the left-hand side of the line segment (directed from VSTART to VEND) should be marked as outside the domain. By default, all triangles generated inside the convex hull of the domain are considered to be inside the domain. This flag may be used to change that, as an alternative to fill zones (see the fill zone section later). Generally, only one line segment needs to have an outside marking present per bounded region in the domain; specifying an outside flag for more than one line segment bounding the same region has no additional effect.
- **RHS_OUTSIDE**
  Indicates that the region on the right-hand side of the line segment (directed from VSTART to VEND) should be marked as outside the domain. Its usage is identical to LHS_OUTSIDE, but it applies to the region on the right-hand side of the line segment instead of the left-hand side.
- **LHS_NO_REFINE**
  Indicates that the triangles in the region on the left-hand side of the line segment (directed from VSTART to VEND) should never be considered for refinement. Note that triangles may still be split along an edge on a constrained line segment, unless that line segment has the NO_SPLIT flag specified (see below). This restricts the mesh refinement process from producing a high-quality mesh in this region, but may be useful in particular cases, e.g. to retain manually-constructed geometry within the region.
- **RHS_NO_REFINE**
  Indicates that the triangles in the region on the right-hand side of the line segment (directed from VSTART to VEND) should never be considered for refinement. Its usage is identical to LHS_NO_REFINE, but it applies to the region on the right-hand side of the line segment instead of the left-hand side.
- **NO_SPLIT**
  Indicates that the line segment should never be split during mesh refinement. This may hinder the mesh refinement process such that it produces poor-quality triangles in the area surrounding the line segment, but may be useful in particular cases, e.g. to ensure an exact geometric interface to a manually-meshed region of the domain.
- **NO_VORONOI_DIVIDE**
  Indicates that the line segment should not divide the Voronoi cells generated by its vertices, when a Voronoi mesh is generated. By default, a line segment will cut through Voronoi cells, ensuring that the segment forms an edge in the resulting mesh. With this property specified, the line segment will not be present in the resulting Voronoi mesh (although its vertices will still define centres of Voronoi cells).

The following example describes a 1000x1000 square domain with line segments bounding the outer edges, and with a 100x100 square in the bottom-left excluded from (outside) the domain. One of the edges of the 100x100 square is divided into 5 equally-spaced intervals using the EVEN_SPACING property, to ensure further mesh refinement along that boundary.

```
BEGIN VERTICES
  0    0      # V0: Bottom-left corner
  1000 0      # V1: Bottom-right corner
  0    1000   # V2: Top-left corner
  1000 1000   # V3: Top-right corner
  100  0      # V4: Bottom-right corner of excluded
              #     100x100 square
  0    100    # V5: Top-left corner of excluded
              #     100x100 square
  100  100    # V6: Top-right corner of excluded
              #     100x100 square
END

BEGIN LINE_SEGMENTS_BY_INDEX

# Outer domain boundary: 1000x1000 square
  0 1         # Bottom-left to bottom-right
  1 3         # Bottom-right to top-right
  3 2         # Top-right to top-left
  2 0         # Top-left to bottom left

# Inner 100x100 square excluded from domain
  5 6 RHS_OUTSIDE
  6 4 EVEN_SPACING 20.0

END

# Notes:
#
#   1. Although the outer boundary is broken into two
#      segments along its left and bottom sides (due to the
#      100x100 square), we don’t need to specify this;
#      AlgoMesh will discover the intersection between the
#      segments and include it in the mesh.
#   2. As the default is to include all parts of the mesh
#      inside the domain, we only needed to specify the one
#      RHS_OUTSIDE flag to exclude the 100x100 square (in
#      this case, we used the right-hand side of the
#      100x100 square’s top-left to top-right edge).
```

An illustration of this sample domain is given in Figure 170.

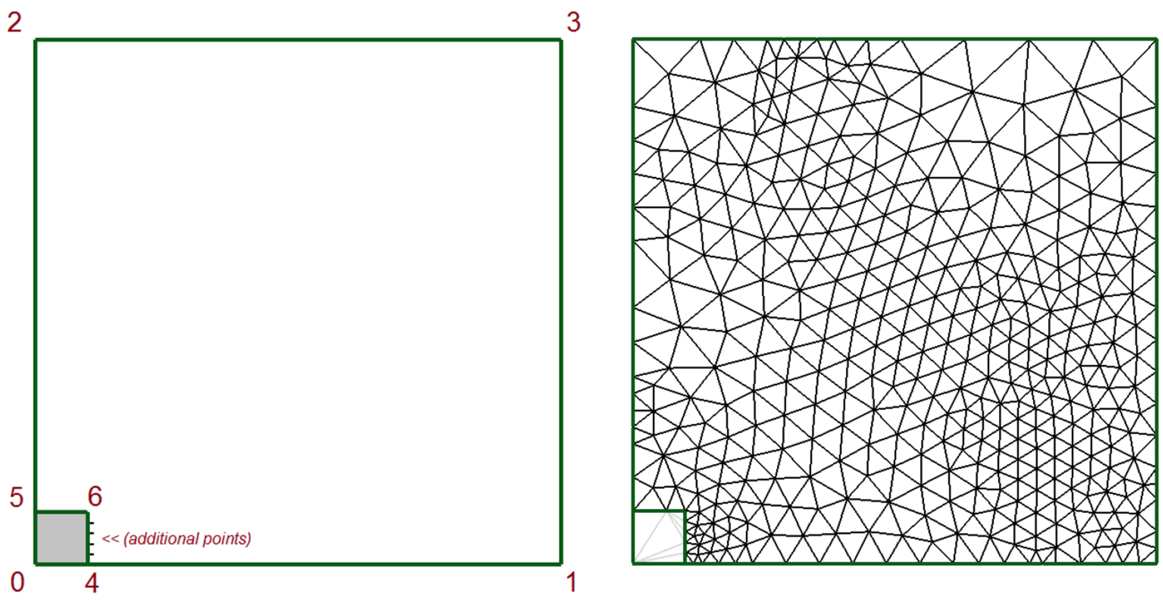


Figure 170: 1000x1000 square domain with an internal excluded 100x100 square, and a mesh generated from it. Vertex indices and additional points from SPACEMODE and MAXSPACE properties are shown.

LINE_SEGMENTS_BY_COORDINATE Section

The LINE_SEGMENTS_BY_COORDINATE section contains a list of line segments that must be present as edges in the generated triangular mesh. It is identical to the LINE_SEGMENTS_BY_INDEX section, except that each vertex is specified by a new pair of (X, Y) coordinates, instead of referring to vertex indices.

```
BEGIN LINE_SEGMENTS_BY_COORDINATE
 # First line segment
 <X1> <Y1> <X2> <Y2> [<Line segment properties>]

 # ...

 # Last line segment
 <X1> <Y1> <X2> <Y2> [<Line segment properties>]
END
```

(X1, Y1) are the coordinates of the starting vertex of the line segment (equivalent to VSTART) and (X2, Y2) are the coordinates of the line segment’s ending vertex (equivalent to VEND). Line segment properties are specified in the same way to those described in the LINE_SEGMENTS_BY_INDEX section.

Specifying line segments in this way adds new points to the mesh domain. A VERTICES section is not required to include a LINE_SEGMENTS_BY_COORDINATE section, but both may be present in the one file if desired. Vertices added by a LINE_SEGMENTS_BY_COORDINATE section are numbered following the end of the last VERTICES section in the entire MDD file (not the previous section), and duplicate vertices will be removed by AlgoMesh. It is not recommended to attempt to refer to these new line segment vertices in a LINE_SEGMENTS_BY_INDEX section, as the numbering is not intuitive (if you need to refer to these points, use a VERTICES section and a LINE_SEGMENTS_BY_INDEX section instead). Furthermore, note that vertices created from this type of section may not have vertex property values specified.

The following example gives an 800x800 square domain, with bounded regions on left and right sides (delineating constant head regions in a groundwater model) and three triangles in the domain interior, representing groundwater wells.

```
BEGIN LINE_SEGMENTS_BY_COORDINATE

  # Outer domain boundary (800x800 square)
     0 800   0   0
     0   0 800   0
   800   0 800 800
   800 800   0 800

  # Constant head boundaries (40x60 divisions at edges)
    40   0  40 800 EVEN_SPACING 60
   760   0 760 800 EVEN_SPACING 60

  # Well 1 (40 base/height triangle around 160,700 well)
   140 680 160 720
   160 720 180 680
   180 680 140 680

  # Well 2 (40 base/height triangle around 400,200 well)
   380 180 400 220
   400 220 420 180
   420 180 380 180

  # Well 3 (40 base/height triangle around 640,500 well)
   620 480 640 520
   640 520 660 480
   660 480 620 480

END
```

An illustration of this sample domain is given in Figure 171.

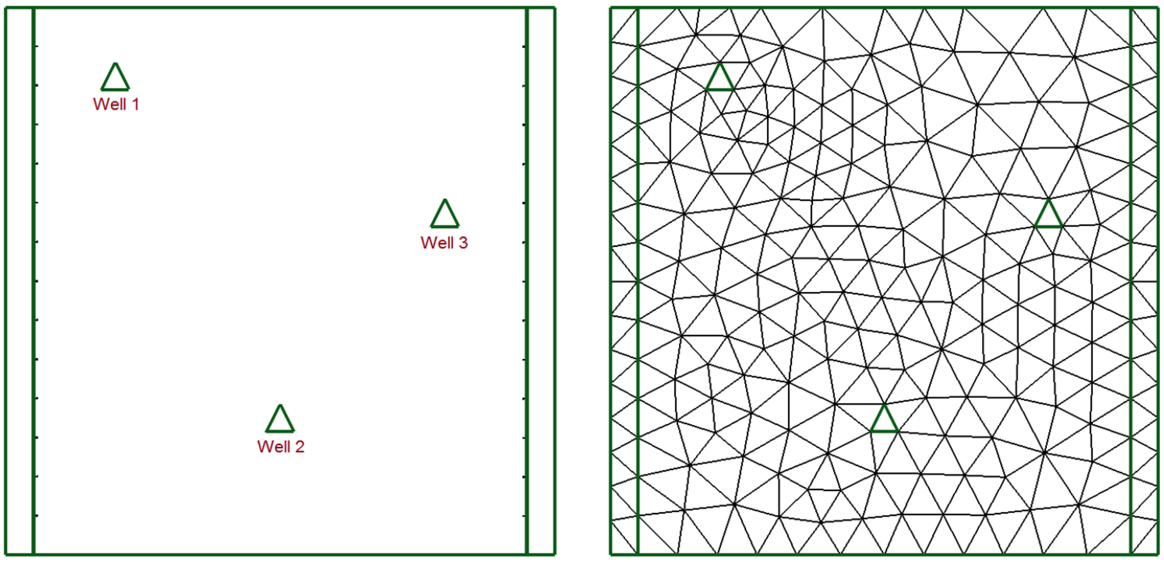


Figure 171: An 800x800 square domain with three triangles (groundwater wells), vertical bounding lines on left and right sides (constant head regions), and a mesh generated from it.

FILL_ZONES Section

The FILL_ZONES section provides a way of marking certain spatial regions as inside or outside the mesh domain. Triangles may initially be generated in regions that are marked as outside the domain – as AlgoMesh triangulates the entire convex hull surrounding all domain elements – but they will not be subject to refinement, and are not included in the final mesh output.

These regions (fill zones) are specified by their type (inside domain markers, outside domain markers, or no refine region markers) and the coordinates of a single point within the region. The specified marker is “flood filled” from the point out to any boundaries it encounters (either line segments specified as part of the domain, or the convex hull of all domain elements).

If any cubic spline curves are present in the mesh domain, fill zones are processed before the splines are sampled.

```
BEGIN FILL_ZONES
 # First fill zone
 <FILLZONETYPE> <X> <Y>

 # Second fill zone
 <FILLZONETYPE> <X> <Y>

 # ...

 # Last fill zone
 <FILLZONETYPE> <X> <Y>
END
```

- **FILLZONETYPE** (required)

One of the following zone types:

- INSIDE
  Marks all triangles inside the zone as inside the domain.
- OUTSIDE
  Marks all triangles inside the zone as outside the domain.
- NO_REFINE
  Indicates that the triangles inside the zone should never be considered for refinement (as per the LHS_NO_REFINE and RHS_NO_REFINE line segment properties). Note that triangles may still be split along an edge on a constrained line segment, unless that line segment has the NO_SPLIT flag specified. This restricts the mesh refinement process from producing a high-quality mesh in this region, but may be useful in particular cases, e.g. to retain manually-constructed geometry within the region.

The following example shows fill zones used to exclude the constant head regions from the LINE_SEGMENTS_BY_COORDINATE sample domain:

```
BEGIN LINE_SEGMENTS_BY_COORDINATE

  # Outer domain boundary (800x800 square)
     0 800   0   0
     0   0 800   0
   800   0 800 800
   800 800   0 800

  # Constant head boundaries (40x60 divisions at edges)
    40   0  40 800 EVEN_SPACING 60
   760   0 760 800 EVEN_SPACING 60

  # Well 1 (40 base/height triangle around 160,700 well)
   140 680 160 720
   160 720 180 680
   180 680 140 680

  # Well 2 (40 base/height triangle around 400,200 well)
   380 180 400 220
   400 220 420 180
   420 180 380 180

  # Well 3 (40 base/height triangle around 640,500 well)
   620 480 640 520
   640 520 660 480
   660 480 620 480

END

BEGIN FILL_ZONES

  # Exclude constant head boundary (left)
  OUTSIDE 20 20
  # Exclude constant head boundary (right)
  OUTSIDE 780 20

END
```

An illustration of this sample domain is given in Figure 172.

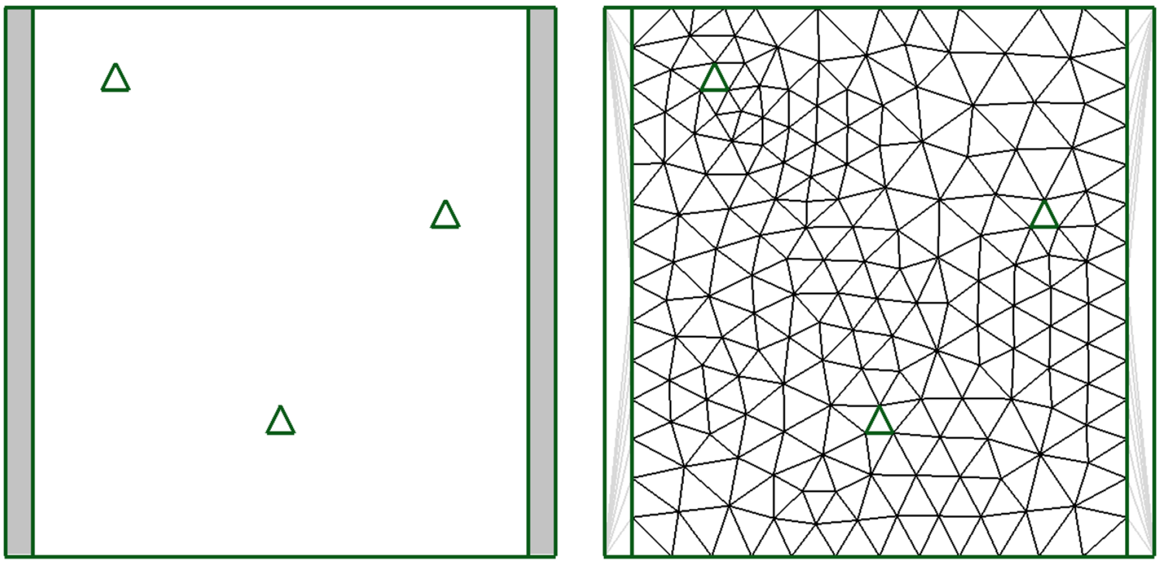


Figure 172: The sample domain with left and right constant head regions marked as outside the domain, and a mesh generated from it.

VERTEX_PROPERTY_DEFAULT_VALUES Section

Each vertex may have one or more named property values associated with it, as described for the VERTICES section. Specifying a named property for at least one vertex in the MDD file makes that property available for interpolation across the domain. For any vertex that does not specify a value for a given named property, a default value is applied. These default values may be specified in the VERTEX_PROPERTY_DEFAULT_VALUES section.

```
BEGIN VERTEX_PROPERTY_DEFAULT_VALUES
  <Property name> <Default property value>
  <Property name> <Default property value>
  # ...
  <Property name> <Default property value>
END
```

Any property that does not have a default value specified in this section is assumed to have a default value of 0.0.

The following example applies a value of 100 for the property INITIAL_HEAD, except at the left-hand side, where a value of 90 is specified to override the default. Default values of 100 and 0 are given for two other properties, TOP_Z and BOTTOM_Z, to apply constant elevations across the domain.

```
BEGIN VERTEX_PROPERTY_DEFAULT_VALUES
  INITIAL_HEAD 100
  TOP_Z        100
  BOTTOM_Z     0
END

BEGIN VERTICES
  0    0     INITIAL_HEAD 90 # Also TOP_Z=100, BOTTOM_Z=0
  1000 0     # Also INITIAL_HEAD=100, TOP_Z=100, BOTTOM_Z=0
  0    1000  INITIAL_HEAD 90 # Also TOP_Z=100, BOTOTM_Z=0
  1000 1000  # Also INITIAL_HEAD=100, TOP_Z=100, BOTTOM_Z=0
END
```

SPLINE_SET Section

Cubic spline curves may be used in combination with vertices and line segments for specifying input geometry for the mesh domain. A SPLINE_SET section defines a collection of these (called a spline set in the AlgoMesh user interface). The splines specified do not go directly as input to the mesh generation process, but are sampled as sequences of constrained vertices and linear edges. Multiple spline sets may be included in one mesh domain description file to assist in organising the splines into groups. A spline set is defined as follows:

```
BEGIN SPLINE_SET
  [NAME <Name>]
  [DISABLED]

  BEGIN SPLINE
    [SPLINE_TYPE        <Spline type>]
    [SAMPLING_TYPE      <Sampling type>]
    [TENSION            <Tension>]
    [START_LENGTH       <Start length>]
    [END_LENGTH         <End length>]
    [CHORDS_PER_SEGMENT <Chords per segment>]
    [NO_VORONOI_DIVIDE  <NO_VORONOI_DIVIDE flag>]
    [NO_SPLIT           <NO_SPLIT flag>]

    BEGIN POINTS
      <X> <Y> [LENGTH_AFTER <LA>]
      <X> <Y> [LENGTH_BEFORE <LB>] [LENGTH_AFTER <LA>]
      # ...
      <X> <Y> [LENGTH_BEFORE <LB>] [LENGTH_AFTER <LA>]
      <X> <Y> [LENGTH_BEFORE <LB>]
    END
  END
  BEGIN SPLINE
    # ...
  END

  # ...
END
```

Each set may have a name, one or more splines (specified by BEGIN SPLINE … END inner sections), and may be active or inactive (active by default, inactive if the keyword DISABLED is present in the section). Inactive spline sets are ignored by the mesh generation process. Spline set names are only used by the user interface, not by the command-line tool.

SPLINE Sub-section

Each spline within the set has a number of parameters controlling the shape and sampling of the curve, and a set of generating points through which the spline must run. All parameters are optional, and if left unspecified will stay at their default values. The following parameters may be specified:

- **SPLINE_TYPE**
  May be CARDINAL (default), LINEAR or NATURAL.
  Defines the type of curve to fit through the generating points. The default is a cardinal spline (CARDINAL), with the "tightness" of the curve controllable by a tension parameter (see below). A linear polyline may be used instead (LINEAR), allowing convenient resampling of points along the polyline, without deviating along a curve. Specifying NATURAL results in a natural cubic spline, which is continuous in first and second derivatives (whereas cardinal splines are continuous only in the first derivative). In general, cardinal splines are recommended over natural splines, as they give much better control over shape and curvature.
- **SAMPLING_TYPE**
  May be PER_SEGMENT (default) or ENTIRE_SPLINE.
  Determines the range over which sampling intervals are calculated. The default of PER_SEGMENT results in every generating point being included in the mesh domain, as well as additional points along each spline segment (section of the curve between two generating points) at intervals determined by parameters local to each spline segment; see the discussion on the POINTS inner section below for information on how these are defined. Specifying ENTIRE_SPLINE indicates that local spline segment parameters should be ignored, and the spline should be sampled between the first and last generating points using the spline's START_LENGTH and END_LENGTH parameters (see below). In the ENTIRE_SPLINE case, only the first and last generating points are guaranteed to be present in the mesh.
- **TENSION**
  Real value between -1.0 and 1.0, inclusive (default 0.5).
  A parameter controlling the "tightness" of a cardinal spline curve. A higher (more positive) value results in a tighter curve (with 1.0 giving a linear polyline), and a lower value results in a looser curve. This value is ignored if SPLINE_TYPE is LINEAR or NATURAL.
- **START_LENGTH**
  Non-zero real value (default -1.0).
  If a positive value is specified, START_LENGTH is the desired distance between sampled points immediately after the first generating point of the spline. If a negative value *v* is specified, the magnitude of the value is instead taken as the desired average number of sampling intervals along each segment of the spline i.e. (total spline length / (number of

generating points * -*v*)). This value is ignored if SAMPLING_TYPE is PER_SEGMENT.

- **END_LENGTH**
  Non-zero real value (default -1.0).
  As per START_LENGTH, this parameter gives either the desired sampling interval immediately before the last generating point of the spline (if positive) or the desired average number of sampling intervals (if negative). If START_LENGTH and END_LENGTH are equal, a constant sampling interval is used to generate points along the spline. If they differ, the sampling interval is gradually increased or decreased as appropriate along the length of the spline in a geometric progression between the two given interval lengths. Note that, in all cases, the interval lengths may be adjusted slightly from what is given such that an integer number of intervals along the spline is obtained.
- **CHORDS_PER_SEGMENT**
  Positive integer value (default 50).
  AlgoMesh approximates all cubic splines using a sequence of linear chords. The CHORDS_PER_SEGMENT parameter determines the number of chords to be used for this approximation between each subsequent pair of generating points. Using a value of 1 would give a linear polyline through the generating points (suitable for use when SPLINE_TYPE is LINEAR), and increasing the value from there results in a closer and closer approximation of the true cubic spline curve.
- **NO_VORONOI_DIVIDE**
  May be 1 (flag set – default) or 0 (flag not set).
  Specifies whether the NO_VORONOI_DIVIDE flag should be set (1) or not (0) on line segment constraints created when the spline is sampled. This flag indicates whether Voronoi cells should be clipped by the line segment or not. See the documentation of this flag in the LINE_SEGMENTS_BY_INDEX section for more information.
- **NO_SPLIT**
  May be 1 (flag set) or 0 (flag not set – default).
  Specifies whether the NO_SPLIT flag should be set (1) or not (0) on line segment constraints created when the spline is sampled. This flag indicates whether the line segments may be split during the mesh generation process or not. See the documentation of this flag in the LINE_SEGMENTS_BY_INDEX section for more information.

A BEGIN POINTS … END inner section defines the set of generating points for each spline. Each line in this section consists of an X coordinate, a Y coordinate, and optionally the following parameters determining the desired sampling intervals local to each spline segment:

- **LENGTH_AFTER <LA>**
  <LA> is a non-zero real number (default -1.0).
  Specifies the desired distance between sampled points immediately following the given point (if positive), or the desired number of sampling intervals between the given point and the following point (if negative). This follows the same rules as START_LENGTH and END_LENGTH (see above), but for an individual spline segment when SAMPLING_TYPE is PER_SEGMENT instead of the entire spline. This value is ignored if SAMPLING_TYPE is ENTIRE_SPLINE, or if specified for the last generating point of the spline (after which there is no segment).
- **LENGTH_BEFORE <LB>**
  <LB> is a non-zero real number (default -1.0).
  As per LENGTH_AFTER. Specifies the desired distance between sampled points immediately prior to the given point (if positive), or the desired number of sampling intervals between the given point and the previous point (if negative). This value is ignored if SAMPLING_TYPE is ENTIRE_SPLINE, or if specified for the first generating point of the spline (before which there is no segment).

When SAMPLING_TYPE is PER_SEGMENT, the sampling intervals along each spline segment are determined by the LENGTH_AFTER value of the first generating point of the segment and the LENGTH_BEFORE value of the next generating point, e.g. the segment between points 1 and 2 uses the LENGTH_AFTER value of point 1 and the LENGTH_BEFORE value of point 2, the segment between points 2 and 3 uses the LENGTH_AFTER value of point 2 and the LENGTH_BEFORE value of point 3, and so on.

6.2 Property TIN Mapping (.ptm)

The PTM file specifies a number of property TINs to be triangulated, and maps property values in those TINs to variables controlling mesh generation or MODFLOW-USG model file output.

The file is divided into sections with BEGIN and END tags, exactly like the MDD file, and follows the same whitespace and commenting rules. It has two sections:

- **PROPERTY_TINS**
  A list of MDD files to be loaded and triangulated as property TINs.
- **PROPERTY_MAPPING**
  A list of mappings from mesh variables used by AlgoMesh to property names defined in the property TINs.

These sections are described under their respective headings below.

PROPERTY_TINS Section

The PROPERTY_TINS section specifies a collection of property TINs to be loaded, each comprising a unique name and a path to an MDD file containing the property domain to be triangulated. Each of these is entered on a single line:

```
BEGIN PROPERTY_TINS
  <Property TIN name> <MDD file path>
  <Property TIN name> <MDD file path>
  # ...
  <Property TIN name> <MDD file path>
END
```

The property TIN names are arbitrary identifiers, and are used only to refer back to the correct property TIN in the PROPERTY_MAPPINGS section. They may not contain any whitespace (space or tab) or separator (comma or semicolon) characters.

The MDD file paths may be specified as absolute paths (e.g. c:\tins\propertytin.mdd) or paths relative to the folder containing the PTM file (e.g. specifying simply propertytin.mdd would cause AlgoMesh to look for a file called propertytin.mdd in the same folder as the PTM file).

Each property TIN may contain any number of properties. The next section, PROPERTY_MAPPING, defines which of these properties map to which AlgoMesh mesh variables.

PROPERTY_MAPPING Section

The PROPERTY_MAPPING section contains a line for each mapping from a property contained in a property TIN to a mesh variable that AlgoMesh recognises, as well as optional stress period and layer specifications:

```
BEGIN PROPERTY_MAPPING
  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Property TIN name> <Property name>

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Property TIN name> <Property name>

  # ...

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Property TIN name> <Property name>
END
```

Each property TIN name must be one of the names specified in the PROPERTY_TINS section. The property name should be one of the named vertex properties for which values have been specified within that property TIN (see the MDD file format description for details on how to specify vertex property values). Note that two or more different property TINs may contain properties of the same name; this is allowed, as the property TIN name is always used to differentiate them.

The mesh variable name determines how AlgoMesh should interpret the specified property values, and must be one of the following:

- **ACTIVE_MODEL_AREA**
  Mesh cell centres with an Active Model Area value of 0 are removed from the MODFLOW-USG model.
- **BOUNDARY_HEAD**
  Mesh cell centres inside the property TIN domain are output as general head boundary cells in the MODFLOW-USG GHB file with the interpolated property values giving the boundary head value. BOUNDARY_HEAD and either BOUNDARY_CONDUCTANCE or BOUNDARY_CONDUCTANCE_PER_UNIT_AREA must be mapped for the GHB file to be written out.
- **BOUNDARY_CONDUCTANCE**
  Mesh cell centres inside the property TIN domain are output as general head boundary cells in the MODFLOW-USG GHB file with the interpolated property values giving the boundary conductance value. BOUNDARY_HEAD and either BOUNDARY_CONDUCTANCE or BOUNDARY_CONDUCTANCE_PER_UNIT_AREA must be mapped for the GHB file to be written out.
- **BOUNDARY_CONDUCTANCE_PER_UNIT_AREA**
  Mesh cell centres inside the property TIN domain are output as general head boundary cells in the MODFLOW-USG GHB file with the interpolated property values multiplied by cell area

to give the boundary conductance value. BOUNDARY_HEAD and either BOUNDARY_CONDUCTANCE or BOUNDARY_CONDUCTANCE_PER_UNIT_AREA must be mapped for the GHB file to be written out.

- **CONSTANT_HEAD**
  Mesh cell centres inside the property TIN domain are output as constant head cells in the MODFLOW-USG CHD file with the interpolated property values giving the constant head value.
- **DRAIN_ELEVATION**
  Mesh cell centres inside the property TIN domain are output as drain cells in the MODFLOW-USG DRN file with the interpolated property values giving the drain elevation. DRAIN_ELEVATION and either DRAIN_CONDUCTANCE or DRAIN_CONDUCTANCE_PER_UNIT_AREA must be mapped for the DRN file to be written out.
- **DRAIN_CONDUCTANCE**
  Mesh cell centres inside the property TIN domain are output as drain cells in the MODFLOW-USG DRN file with the interpolated property values giving the drain conductance. DRAIN_ELEVATION and either DRAIN_CONDUCTANCE or DRAIN_CONDUCTANCE_PER_UNIT_AREA must be mapped for the DRN file to be written out.
- **DRAIN_CONDUCTANCE_PER_UNIT_AREA**
  Mesh cell centres inside the property TIN domain are output as drain cells in the MODFLOW-USG DRN file with the interpolated property values multiplied by cell area to give the drain conductance. DRAIN_ELEVATION and either DRAIN_CONDUCTANCE or DRAIN_CONDUCTANCE_PER_UNIT_AREA must be mapped for the DRN file to be written out.
- **ELEVATION_TOP**
  Interpolated property values are output as layer top elevations in the MODFLOW-USG DISU file.
- **ELEVATION_BOTTOM**
  Interpolated property values are output as layer bottom elevations in the MODFLOW-USG DISU file.
- **ET_SURFACE_ELEVATION**
  Interpolated property values are output as evapotranspiration surface elevations in the MODFLOW-USG EVT file. ET_SURFACE_ELEVATION, ET_MAXIMUM_FLUX and ET_EXTINCTION_DEPTH must all be mapped for the EVT file to be written out.
- **ET_MAXIMUM_FLUX**
  Interpolated property values are output as evapotranspiration maximum flux values in the MODFLOW-USG EVT file. No modification for cell area is made. ET_SURFACE_ELEVATION, ET_MAXIMUM_FLUX and ET_EXTINCTION_DEPTH must all be mapped for the EVT file to be written out.
- **ET_EXTINCTION_DEPTH**
  Interpolated property values are output as evapotranspiration extinction depth values in the MODFLOW-USG EVT file. ET_SURFACE_ELEVATION, ET_MAXIMUM_FLUX and ET_EXTINCTION_DEPTH must all be mapped for the EVT file to be written out.
- **INITIAL_HEAD**
  Interpolated property values are output as initial heads in the MODFLOW-USG BAS file.
- **KX**
  Interpolated property values are output as Kx values in the MODFLOW-USG LPF file. Anisotropy is not supported currently (i.e. Ky = Kx).
- **KZ**
  Interpolated property values are output as Kz values in the MODFLOW-USG LPF file.
- **LATERAL_CONNECTION_GROUP**
  Mesh cell centres are assigned an LCG from this variable for advanced control over the

lateral connections in the MODFLOW-USG DISU file created by AlgoMesh. See the Lateral Connection Groups advanced topic for more information.

- **MESH_CELL_EDGE_LENGTH**
  Interpolated property values are taken as a maximum desired triangular edge length local to a given location during edge length field generation and mesh generation. Locations outside the property TIN domain are given no local maximum edge length.
- **MESH_CELL_EDGE_LENGTH_FIELD_GENERATION_ONLY**
  Interpolated property values are taken as a maximum desired triangular edge length local to a given location during edge length field generation (but not mesh generation). Locations outside the property TIN domain are given no local maximum edge length.
- **NO_REFINE**
  AlgoMesh will not split triangles inside areas of a mesh domain with a No Refinement Area value of 1 during mesh generation. See the Edge Constraint Flags and No Refinement Areas advanced topic for more information.
- **RECHARGE**
  Interpolated property values are output as recharge rates in the MODFLOW-USG RCH file. No modification for cell area is made.
- **RIVER_STAGE**
  Mesh cell centres inside the property TIN domain are output as river cells in the MODFLOW-USG RIV file with the interpolated property values giving the river stage. RIVER_STAGE, RIVER_BOTTOM_ELEVATION and RIVER_CONDUCTANCE must all be mapped for the RIV file to be written out.
- **RIVER_BOTTOM_ELEVATION**
  Mesh cell centres inside the property TIN domain are output as river cells in the MODFLOW-USG RIV file with the interpolated property values giving the bottom elevation of the riverbed. RIVER_STAGE, RIVER_BOTTOM_ELEVATION and either RIVER_CONDUCTANCE or RIVER_CONDUCTANCE_PER_UNIT_AREA must be mapped for the RIV file to be written out.
- **RIVER_CONDUCTANCE**
  Mesh cell centres inside the property TIN domain are output as river cells in the MODFLOW-USG RIV file with the interpolated property values giving the river cell conductance. RIVER_STAGE, RIVER_BOTTOM_ELEVATION and either RIVER_CONDUCTANCE or RIVER_CONDUCTANCE_PER_UNIT_AREA must be mapped for the RIV file to be written out.
- **RIVER_CONDUCTANCE_PER_UNIT_AREA**
  Mesh cell centres inside the property TIN domain are output as river cells in the MODFLOW-USG RIV file with the interpolated property values multiplied by cell area to give the river cell conductance. RIVER_STAGE, RIVER_BOTTOM_ELEVATION and either RIVER_CONDUCTANCE or RIVER_CONDUCTANCE_PER_UNIT_AREA must be mapped for the RIV file to be written out.
- **SPECIFIC_STORAGE**
  Interpolated property values are output as specific storage in the MODFLOW-USG LPF file. AlgoMesh outputs specific storage when there is at least one transient stress period. If no mapping is specified, a zero value will be written for every cell.
- **SPECIFIC_YIELD**
  Interpolated property values are output as specific yield in the MODFLOW-USG LPF file. AlgoMesh outputs specific yield when there is at least one transient stress period. If no mapping is specified, a zero value will be written for every cell.
- **TVM_KX**
  Mesh cell centres inside the property TIN domain are output as cells with Kx property changes in the MODFLOW-USG Time-Variant Materials (TVM) file, with the changed property value obtained from the interpolated TVM_KX value and applied at the end of the mapped stress period.

- **TVM_KZ**
  Mesh cell centres inside the property TIN domain are output as cells with Kz property changes in the MODFLOW-USG Time-Variant Materials (TVM) file, with the changed property value obtained from the interpolated TVM_KZ value and applied at the end of the mapped stress period.
- **TVM_SPECIFIC_STORAGE**
  Mesh cell centres inside the property TIN domain are output as cells with Ss property changes in the MODFLOW-USG Time-Variant Materials (TVM) file, with the changed property value obtained from the interpolated TVM_SPECIFIC_STORAGE value and applied at the end of the mapped stress period.
- **TVM_SPECIFIC_YIELD**
  Mesh cell centres inside the property TIN domain are output as cells with Sy property changes in the MODFLOW-USG Time-Variant Materials (TVM) file, with the changed property value obtained from the interpolated TVM_SPECIFIC_YIELD value and applied at the end of the mapped stress period.
- **WELL_PUMPING_RATE**
  Mesh cell centres inside the property TIN domain are output as well cells in the MODFLOW-USG WEL file with the interpolated property values giving the pumping rate.

Lines enclosed in square brackets above (SP and LAYER specifications) are optional, and if used, should not include the square brackets, parentheses or angle brackets. Examples are given at the end of this section.

The SP specification, if used, specifies that the next mapping, and all mappings below until the next SP specification, apply to the given stress period number (e.g. SP 1 applies the subsequent mappings to the first stress period). If no SP specification is provided, mappings apply to the default stress period 1. If the word ALL appears instead of a stress period number, subsequent mappings are applied at all stress periods.

The LAYER specification operates in the same way as the SP specification, but for model layers. If no layer specifications are given, the default layer 1 is used. The word ALL may also be used here to specify that subsequent mappings should be applied to all model layers.

Each mesh variable may only have one mapping for each combination of layer and stress period, but a single property may be mapped to multiple mesh variables (or the same mesh variable in different layers and stress periods) by including a line for each (e.g. a surface elevation property could map to both ELEVATION_TOP and INITIAL_HEAD).

The following listing gives an example of a single-layer, steady-state PTM file. Note that no SP or LAYER specifications are necessary, as the model only has one layer and stress period.

```
BEGIN PROPERTY_TINS
 TIN_EdgeLength river-properties-edge-length.mdd
 TIN_RegionWideProperties river-properties-region-wide.mdd
 TIN_ConstantHeadBoundaries river-properties-constant-
head.mdd
 TIN_Wells river-properties-wells.mdd
 TIN_River river-properties-river.mdd
END

BEGIN PROPERTY_MAPPING
```

```
 MESH_CELL_EDGE_LENGTH TIN_EdgeLength MeshCellEdgeLength
 ELEVATION_BOTTOM TIN_RegionWideProperties Bottom
 ELEVATION_TOP TIN_RegionWideProperties Top
 INITIAL_HEAD TIN_RegionWideProperties InitialHead
 KX TIN_RegionWideProperties Kx
 KZ TIN_RegionWideProperties Kz
 RECHARGE TIN_RegionWideProperties Recharge
 CONSTANT_HEAD TIN_ConstantHeadBoundaries ConstantHead
 WELL_PUMPING_RATE TIN_Wells PumpingRate
 RIVER_BOTTOM_ELEVATION TIN_River RiverBedBottomElevation
 RIVER_STAGE TIN_River RiverStage
 RIVER_CONDUCTANCE TIN_River RiverConductance
END
```

A multi-layer, transient PTM example is given below.

```
BEGIN PROPERTY_TINS
 TIN1 river-properties-constant-head.mdd
 TIN2 river-properties-region-wide.mdd
 TIN3 river-properties-river.mdd
 TIN4 river-properties-wells-transient.mdd
END

BEGIN PROPERTY_MAPPING
 SP 1
 LAYER ALL
  KX TIN2 Kx
  KZ TIN2 Kz
  INITIAL_HEAD TIN2 InitialHead
 LAYER 1
  CONSTANT_HEAD TIN1 ConstantHead
  ELEVATION_TOP TIN2 TopL1
  ELEVATION_BOTTOM TIN2 BotL1
  RECHARGE TIN2 Recharge
  RIVER_STAGE TIN3 RiverStage
  RIVER_BOTTOM_ELEVATION TIN3 RiverBedBottomElevation
  RIVER_CONDUCTANCE TIN3 RiverConductance
  WELL_PUMPING_RATE TIN4 PumpingRate2
 LAYER 2
  ELEVATION_BOTTOM TIN2 BotL2
 LAYER 3
  ELEVATION_BOTTOM TIN2 BotL3
 SP 3
 LAYER 1
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 5
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 7
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 9
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 11
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 13
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 15
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 17
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 19
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 21
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 23
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 25
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 27
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 29
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 31
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 33
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 35
  WELL_PUMPING_RATE TIN4 PumpingRate
 SP 37
  WELL_PUMPING_RATE TIN4 PumpingRate2
 SP 39
  WELL_PUMPING_RATE TIN4 PumpingRate
END
```

6.3 Property Polygon Mapping (.ppm)

The PPM file specifies a collection of property polygon sets, including the polygons and properties they contain, and mappings between the polygon properties and AlgoMesh variables controlling mesh generation or MODFLOW-USG model file output.

The file is divided into sections with BEGIN and END tags, exactly like the MDD file, and follows the same whitespace and commenting rules. It has two section types:

- **PROPERTY_POLYGON_SET**
  A collection of polygons, with geometry and properties specified for each.
- **PROPERTY_POLYGON_MAPPING**
  A list of mappings from mesh variables used by AlgoMesh to polygon properties.

Each of these sections may appear more than once in a PPM file. The order of PROPERTY_POLYGON_SET sections in the file determines the order of evaluation for duplicate mappings, as per the ordering of polygon sets in the user interface. Each section is described under its respective heading below.

PROPERTY_POLYGON_SET Section

The PROPERTY_POLYGON_SET section specifies a collection of polygons with property values associated to each. Multiple property polygon sets may be included in a single PPM file by specifying multiple PROPERTY_POLYGON_SET sections.

```
BEGIN PROPERTY_POLYGON_SET
  [ID   <ID>]
  [NAME <Name>]
  [HIDDEN]

  BEGIN PROPERTY_DEFAULT_VALUES
    <Property name> <Default value>
    <Property name> <Default value>
    # ...
    <Property name> <Default value>
  END

  BEGIN PROPERTY_POLYGON
    BEGIN SPLINE
      [SPLINE_TYPE        <Spline type>]
      [SAMPLING_TYPE      <Sampling type>]
      [TENSION            <Tension>]
      [START_LENGTH       <Start length>]
      [END_LENGTH         <End length>]
      [CHORDS_PER_SEGMENT <Chords per segment>]
      [NO_VORONOI_DIVIDE  <NO_VORONOI_DIVIDE flag>]
      [NO_SPLIT           <NO_SPLIT flag>]

      BEGIN POINTS
        <X> <Y> [LENGTH_AFTER <LA>]
        <X> <Y> [LENGTH_BEFORE <LB>] [LENGTH_AFTER <LA>]
        # ...
        <X> <Y> [LENGTH_BEFORE <LB>] [LENGTH_AFTER <LA>]
        <X> <Y> [LENGTH_BEFORE <LB>]
      END
    END

    BEGIN PROPERTIES
      <Property name> <Value>
      <Property name> <Value>
      # ...
      <Property name> <Value>
    END
  END

  BEGIN PROPERTY_POLYGON
    # ...
  END

  # ...
END
```

The ID of the polygon set is used to define mappings to mesh variables (see the PROPERTY_POLYGON_MAPPING section below). It may not contain any whitespace (space or tab) or separator (comma or semicolon) characters. Any polygon sets for which the ID is omitted will be given IDs PROPERTY_POLYGON_SET_1, PROPERTY_POLYGON_SET_2 and so on, in the order they appear in the PPM file.

The NAME of the polygon set is used only in the user interface, and does not affect any property mappings. The optional HIDDEN keyword, if specified, causes the polygon set to be invisible in the user interface - but any mappings to its properties still apply.

Each set specifies a number of properties in the PROPERTY_DEFAULT_VALUES sub-section. Each property must be specified on a separate line within this sub-section, comprising its name followed by its default value (a real number). Property names can be anything, as long as they comprise only non-whitespace and non-separator (comma and semicolon) characters.

Any number of PROPERTY_POLYGON sub-sections may be included in a PROPERTY_POLYGON_SET section. Each PROPERTY_POLYGON sub-section itself must contain a SPLINE sub-section and a PROPERTIES sub-section.

The SPLINE sub-section is identical to the SPLINE sub-section used in the Mesh Domain Description (.mdd) file format; refer to that section for details. It is used to define the geometry of the polygon's boundary. Note that the START_LENGTH, END_LENGTH, NO_VORONOI_DIVIDE, NO_SPLIT, LENGTH_AFTER and LENGTH_BEFORE options may be specified but will be ignored for polygons, as polygons are only used for property mappings, and are not used to add geometry to the mesh.

The PROPERTIES sub-section contains one or more property values explicitly assigned to the polygon. Each property assignment must be written on a single line and comprises the property name followed by its value for that polygon. Any property that appears in the PROPERTY_DEFAULT_VALUES section but does not appear in a polygon's PROPERTIES sub-section is given the property's default value for that polygon.

PROPERTY_POLYGON_MAPPING Section

The PROPERTY_POLYGON_MAPPING section contains a line for each mapping from a property contained in a property polygon set to a mesh variable that AlgoMesh recognises, as well as optional stress period and layer specifications:

```
BEGIN PROPERTY_POLYGON_MAPPING
```

```
  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Polygon set ID> <Property name>

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Polygon set ID> <Property name>

  # ...

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Polygon set ID> <Property name>
END
```

Each polygon set ID must be an ID specified in one of the PROPERTY_POLYGON_SET sections in the same PPM file. The property name should be one of the named properties for which values have been specified within that property polygon set. Note that two or more different property polygon sets may contain properties of the same name; this is allowed, as the property polygon set ID is always used to differentiate them.

The stress period and layer specifications, as well as the mesh variable name used to select the AlgoMesh variable for each mapping, are identical to those used in the PROPERTY_MAPPING section of the Property TIN Mapping (.ptm) file format; refer to that section for details. Instead of values being interpolated inside a property TIN domain, property values are constant within each polygon contained in the property polygon set specifying the property. A mapping provides no value in areas of the mesh domain that are not covered by a polygon in the mapped polygon set.

6.4 Property Table Mapping (.ptab)

The PTAB file specifies a collection of property tables to be read from external text files, including information on their format and index fields, and mappings between the table fields and AlgoMesh variables controlling mesh generation or MODFLOW-USG model file output.

The file is divided into sections with BEGIN and END tags, exactly like the MDD file, and follows the same whitespace and commenting rules. It has two section types:

- **PROPERTY_TABLE_CSV**
  A specification of an external, delimited text file to be read as a property table, and details of its format.
- **PROPERTY_TABLE_MAPPING**
  A list of mappings from mesh variables used by AlgoMesh to table fields.

Each of these sections may appear more than once in a PTAB file. They are described under their respective headings below.

PROPERTY_TABLE_CSV Section

The PROPERTY_TABLE_CSV section specifies the location of an external, delimited text file containing property data to be mapped, and details on how to read the file. Multiple property tables may be referenced in a single PTAB file by specifying multiple PROPERTY_TABLE_CSV sections.

```
BEGIN PROPERTY_TABLE_CSV
  [ID              <ID>]
  [NAME            <Name>]
  [FILENAME        <File path>]
  [DELIMITERS      <Character> <Character> ... <Character>]
  [TEXT_QUALIFIERS <Character> <Character> ... <Character>]
  [TREAT_CONSECUTIVE_DELIMITERS_AS_ONE]
  [FIRST_LINE_HAS_HEADERS]
  [IGNORE_HEADER_FIELD_NAMES]
  [FIELD_NAME_GLOBAL_CELL_INDEX <Field name>]
  [FIELD_NAME_LAYER_CELL_INDEX  <Field name>]
  [FIELD_NAME_LAYER             <Field name>]
  [FIELD_NAME_STRESS_PERIOD     <Field name>]
END
```

The ID of the property table is used to define mappings to mesh variables (see the PROPERTY_TABLE_MAPPING section below). It may not contain any whitespace (space or tab) or separator (comma or semicolon) characters. Any polygon sets for which the ID is omitted will be given IDs PROPERTY_TABLE_CSV_1, PROPERTY_TABLE_CSV_2 and so on, in the order they appear in the PTAB file.

The NAME of the property table is used only in the user interface, and does not affect any property mappings.

The FILENAME specifies the path to the delimited text file containing the property table data. It may be specified as an absolute path (e.g. c:\data\propertytable.csv) or as a path relative to the folder containing the PTAB file (e.g. specifying simply propertytable.csv would cause AlgoMesh to look for a file called propertytable.csv in the same folder as the PTAB file).

The remaining lines in the PROPERTY_TABLE_CSV section determine how the text file is interpreted (its format) and the fields in the table that are used as cell, layer and stress period indices. These are described under their respective headings below. Note that either FIELD_NAME_GLOBAL_CELL_INDEX or FIELD_NAME_LAYER_CELL_INDEX (but not both), should be specified; otherwise, property mappings to this table will have no effect, as AlgoMesh will be unable to determine the cell to which each row pertains.

DELIMITERS

Specifies one or more ASCII characters that are used to delimit fields in the table file. The word SPACE may be used to indicate the space character, TAB to indicate the tab character, COMMA to indicate a comma, and SEMICOLON to indicate a semicolon. Any other characters should be inserted explicitly. For example, the following line would specify that tabs, colons and forward slash characters may all be used as delimiters in the file:

```
DELIMITERS      TAB : /
```

Default: COMMA

TEXT_QUALIFIERS

Specifies one or more ASCII characters that are used to denote strings of text in the table file. Delimiter characters after a text qualifier are ignored until a matching text qualifier is encountered to close the text string. The word SPACE may be used to indicate the space character, TAB to indicate the tab character, COMMA to indicate a comma, and SEMICOLON to indicate a semicolon. Any other characters should be inserted explicitly. For example, the following line would specify that double quotation marks, single quotation marks and semicolon characters may all be used as text qualifiers in the file:

```
TEXT_QUALIFIERS " ' SEMICOLON
```

Default: " (double quotation mark)

TREAT_CONSECUTIVE_DELIMITERS_AS_ONE

If this option is specified, multiple consecutive instances of delimiter characters in the file are considered a single break between fields. If not specified, each consecutive delimiter character encountered is assumed to indicate an additional (empty) field.

Default: not specified (turned off)

FIRST_LINE_HAS_HEADERS

If this option is specified, AlgoMesh does not attempt to read values from the first line of the file; instead, it will use this line to determine the name of each field. If not specified, the first line is assumed to contain data values and fields are named FIELD1, FIELD2, etc.

Default: not specified (turned off)

IGNORE_HEADER_FIELD_NAMES

If this option is specified, AlgoMesh will ignore any field names it reads from the first line of the file and instead use generic field names FIELD1, FIELD2, etc. If not specified, field names will be determined from the first line of the file if FIRST_LINE_HAS_HEADERS is specified. This option has no effect if FIRST_LINE_HAS_HEADERS is not also specified.

Default: not specified (turned off)

FIELD_NAME_GLOBAL_CELL_INDEX

Specifies the name of a table field to be used to index cell-by-cell mappings as a global cell index over the entire 3D model. Note that a global cell index may not be used to map Top Elevation or Bottom Elevation if Clean Layer Elevations is turned on in Model Setup, and may not be used to map

Active Model Area at all, as these properties affect the way cells are index globally. To map to these properties, use FIELD_NAME_LAYER_CELL_INDEX instead.

FIELD_NAME_LAYER_CELL_INDEX

Specifies the name of a table field to be used to index cell-by-cell mappings as the 2D mesh cell index within a layer of the 3D model. The 2D mesh cell indexing scheme is identical for each layer and does not change when cells are removed from the 3D model by means of pinch-outs or Active Model Area mappings.

FIELD_NAME_LAYER

Specifies the name of a table field defining the layer in which cell-by-cell mappings are applied. This is optional; the layer may alternatively be specified with each mapping in the PROPERTY_TABLE_MAPPING section. This has no effect if FIELD_NAME_GLOBAL_CELL_INDEX is used, as each global cell index incorporates the layer of a cell implicitly.

FIELD_NAME_STRESS_PERIOD

Specifies the name of a table field defining the stress period in which transient cell-by-cell mappings are applied. This is optional; the stress period may alternatively be specified with each mapping in the PROPERTY_TABLE_MAPPING section.

PROPERTY_TABLE_MAPPING Section

The PROPERTY_TABLE_MAPPING section contains a line for each mapping from a field contained in a property table to a mesh variable that AlgoMesh recognises, as well as optional stress period and layer specifications:

```
BEGIN PROPERTY_TABLE_MAPPING
  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Property table ID> <Field name>

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Property table ID> <Field name>

  # ...

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Property table ID> <Field name>
END
```

Each property table ID must be an ID specified in one of the PROPERTY_TABLE_CSV sections in the same PTAB file. The field name should be the name of one of the columns in the property table. Note that two or more different property tables may contain fields of the same name; this is allowed, as the property table ID is always used to differentiate them.

The stress period and layer specifications, as well as the mesh variable name used to select the AlgoMesh variable for each mapping, are identical to those used in the PROPERTY_MAPPING section of the Property TIN Mapping (.ptm) file format; refer to that section for details. Instead of values being interpolated inside a property TIN domain, property values are mapped explicitly on a cell-by-cell basis from rows in the property table. A mapping provides no value for any cell without a corresponding row in the mapped property table.

6.5 Python Script Mapping (.pym)

The PYM file specifies the code for a collection of Python scripts, and mappings between the scripts and AlgoMesh variables controlling mesh generation or MODFLOW-USG model file output. See the Python Scripts advanced topic for more information on using Python scripts with AlgoMesh.

The file is divided into sections with BEGIN and END tags, exactly like the MDD file, and follows the same whitespace and commenting rules. It has two section types:

- **PYTHON_SCRIPT**
  A specification of an embedded Python script.
- **PYTHON_SCRIPT_MAPPING**
  A list of mappings from mesh variables used by AlgoMesh to Python scripts.

Each of these sections may appear more than once in a PYM file. They are described under their respective headings below.

PYTHON_SCRIPT Section

The PYTHON_SCRIPT section specifies the code for an embedded Python script. Multiple Python scripts may be included in a single PYM file by specifying multiple PYTHON_SCRIPT sections.

```
BEGIN PYTHON_SCRIPT
  [ID                <ID>]
  [NAME              <Name>]
  [SCRIPT_END_TOKEN <Script end token>]
  SCRIPT
  <Script code>
  ...
  <Script end token>
END
```

The ID of the Python script is used to define mappings to mesh variables (see the PYTHON_SCRIPT_MAPPING section below). It may not contain any whitespace (space or tab) or separator (comma or semicolon) characters. Any polygon sets for which the ID is omitted will be given IDs PYTHON_SCRIPT_1, PYTHON_SCRIPT_2 and so on, in the order they appear in the PTAB file.

The NAME of the property table is used only in the user interface, and does not affect any property mappings.

The code for the Python script itself is included from the line after the SCRIPT token is read. As Python script code may contain arbitrary keywords and variable names, including the typical BEGIN and END tokens used in the rest of the PYM file, a unique script end token is used instead to indicate the end of the script code. A custom script end token may be specified prior to the start of the script code using the SCRIPT_END_TOKEN setting. The script end token should be something that does not appear in the actual Python script code. If a custom script end token is not specified, the default script end token is used: ~~~END_SCRIPT~~~

All lines appearing after the line containing the SCRIPT token are included in the Python script code precisely as-is (i.e. they are not trimmed of whitespace or comments), until a line containing the script end token is read. This end line should contain only the script end token (it may at most contain whitespace around the script end token and a comment after it). The end line itself is not included in the Python script code.

PYTHON_SCRIPT_MAPPING Section

The PYTHON_SCRIPT_MAPPING section contains a line for each mapping from a Python script to a mesh variable that AlgoMesh recognises, as well as optional stress period and layer specifications:

```
BEGIN PYTHON_SCRIPT_MAPPING
  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Python script ID>

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Python script ID>

  # ...

  [SP (<Stress Period> | ALL)]
  [LAYER (<Layer> | ALL)]

  <Mesh variable name> <Python script ID>
END
```

Each Python script ID must be an ID specified in one of the PYTHON_SCRIPT sections in the same PTAB file. Note that, unlike property TIN, property polygon and property table mappings, Python scripts have no associated property names, so these do not appear in the PYTHON_SCRIPT_MAPPING section.

The stress period and layer specifications, as well as the mesh variable name used to select the AlgoMesh variable for each mapping, are identical to those used in the PROPERTY_MAPPING section of the Property TIN Mapping (.ptm) file format; refer to that section for details. Instead of values being interpolated inside a property TIN domain, property values obtained by executing the Python script and inspecting the **value** variable that the script sets, as described in the Python Scripts advanced topic. If **value** is set to **None** when the script finishes executing, no value is provided for the property mapping.

PYM Example

The following listing gives an example of a PYM file used to implement a custom Mesh Cell Edge Length (Field Generation Only) mapping.

```
BEGIN PYTHON_SCRIPT
     ID PYTHON_SCRIPT_1
     NAME Script 1
     SCRIPT_END_TOKEN ~~~END_SCRIPT~~~
     SCRIPT
if(x >= 3000.0 and x <= 8000.0 and y >= 3000.0 and y <=
8000.0):
     value = 200.0
else:
     value = 500.0
     ~~~END_SCRIPT~~~
END

BEGIN PYTHON_SCRIPT_MAPPING
     SP 1
          LAYER 1
               MESH_CELL_EDGE_LENGTH_FIELD_GENERATION_ONLY
PYTHON_SCRIPT_1
END
```

6.6 Model Setup File (.msf)

The MSF specifies model setup information for a MODFLOW-USG model. This information has no effect on the two-dimensional mesh generation itself, but controls what AlgoMesh exports to the MODFLOW-USG model files.

Model setup information is specified by setting parameters from the available set below:

```
LAYERS                           1
LAYER_TYPE                       4
CLEAN_LAYER_ELEVATIONS           1
FORCED_MINIMUM_LAYER_THICKNESS   0.0
MINIMUM_ACTIVE_LAYER_THICKNESS   0.0
STRESS_PERIODS                   1
STRESS_PERIODS_TRANSIENT         0
```

```
STRESS_PERIOD_LENGTH            1.0
TIME_STEPS_PER_STRESS_PERIOD    1
TIME_STEP_MULTIPLIER            1.2
HDRY                            1.0e+30
HNOFLO                          9999.0
OUTPUT_CBB                      1
CONSTANTCV                      1
NOVFC                           1
NOCVCORRECTION                  0
# (Optional) Repeat the following any number of times
LCG_CONNECTION_BY_APPORTIONMENT 0 1
# (Optional) Repeat the following any number of times
LCG_CONNECTION_BY_ELEVATION     2 3
# (Optional) Repeat the following any number of times
LCG_CONNECTION_BY_LAYER         4 5
```

All parameters are optional, and take their default values if not specified. Each parameter is described under its respective heading below.

LAYERS

```
LAYERS                          Integer > 0
```

Determines the number of layers in the MODFLOW-USG model.

MODFLOW-USG DISU variable: NLAY

Default: 1

LAYER_TYPE

```
LAYER_TYPE                      3 or 4
```

Determines the layer type in the MODFLOW-USG model. Currently, AlgoMesh accepts the values 3 (convertible) or 4 (upstream weighting).

MODFLOW-USG LPF variable: LAYTYP (all layers)

Default: 4

CLEAN_LAYER_ELEVATIONS

```
CLEAN_LAYER_ELEVATIONS          0 or 1
```

Determines whether AlgoMesh sets the top elevation of every cell to the bottom elevation of the next active cell above it (1) or leaves elevations untouched (0). Must be 1 for FORCED_MINIMUM_LAYER_THICKNESS or MINIMUM_ACTIVE_LAYER_THICKNESS to be used.

Default: 1

FORCED_MINIMUM_LAYER_THICKNESS

```
FORCED_MINIMUM_LAYER_THICKNESS  Real number >= 0.0
```

If CLEAN_LAYER_ELEVATIONS is 1, AlgoMesh will adjust cell bottom elevations to ensure that each cell has a thickness of at least the value specified for FORCED_MINIMUM_LAYER_THICKNESS.

Default: 0.0

MINIMUM_ACTIVE_LAYER_THICKNESS

```
MINIMUM_ACTIVE_LAYER_THICKNESS  Real number >= 0.0
```

If CLEAN_LAYER_ELEVATIONS is 1, AlgoMesh will remove any cell from the model whose thickness is less than the value specified for MINIMUM_ACTIVE_LAYER_THICKNESS.

Default: 0.0

STRESS_PERIODS

```
STRESS_PERIODS                  Integer > 0
```

Determines the number of stress periods in the MODFLOW-USG model. All stress periods are exported using the same properties (transient/steady-state, length, number of time steps and time step multiplier), specified by the other parameters in the MSF.

MODFLOW-USG DISU variable: NPER

Default: 1

STRESS_PERIODS_TRANSIENT

```
STRESS_PERIODS_TRANSIENT        0 or 1
```

Specifies whether stress periods should be transient (1) or steady-state (0). If transient, AlgoMesh will include Specific Storage and Specific Yield in its export of the LPF package file.

MODFLOW-USG DISU variable: Ss/Tr (all stress periods)

Default: 0 (Steady-state)

STRESS_PERIOD_LENGTH

```
STRESS_PERIOD_LENGTH            Real number > 0.0
```

Specifies the length of each stress period. Units are not specified, but should be consistent with units used for other values in the model.

MODFLOW-USG DISU variable: PERLEN (all stress periods)

Default: 1.0

TIME_STEPS_PER_STRESS_PERIOD

```
TIME_STEPS_PER_STRESS_PERIOD    Integer > 0
```

Determines the number of time steps in each stress period.

MODFLOW-USG DISU variable: NSTP (all stress periods)

Default: 1

TIME_STEP_MULTIPLIER

```
TIME_STEP_MULTIPLIER            Real number > 0.0
```

Specifies the multiplier for the length of successive time steps.

MODFLOW-USG DISU variable: TSMULT (all stress periods)

Default: 1.2

HDRY

```
HDRY                            Real number
```

Specifies the head to be assigned to cells that are converted to dry during a MODFLOW-USG simulation.

MODFLOW-USG LPF variable: HDRY

Default: 1.0e+30

HNOFLO

```
HNOFLO                                Real number
```

Specifies the head to be assigned to no-flow cells.

MODFLOW-USG BAS variable: HNOFLO

Default: 9999.0

OUTPUT_CBB

```
OUTPUT_CBB                            0 or 1
```

If set to 1, AlgoMesh will specify a single .CBB binary output file in the MODFLOW-USG .NAM file, and direct the cell-by-cell flow outputs for all packages to this file. If set to 0, cell-by-cell flow outputs will be omitted.

Default: 1

CONSTANTCV

```
CONSTANTCV                            0 or 1
```

If turned on, the keyword CONSTANTCV will be added to the LPF package header.

Default: 1

NOVFC

```
NOVFC                                 0 or 1
```

If turned on, the keyword NOVFC will be added to the LPF package header.

Default: 1

NOCVCORRECTION

```
NOCVCORRECTION                        0 or 1
```

If turned on, the keyword NOCVCORRECTION will be added to the LPF package header.

Default: 0

LCG_CONNECTION_BY_APPORTIONMENT

```
LCG_CONNECTION_BY_APPORTIONMENT Integer A  Integer B
```

Specifies that AlgoMesh should use the *connection by apportionment* method to laterally connect cells of Lateral Connection Group A with cells of Lateral Connection Group B. See the advanced topic Lateral Connection Groups for more information.

This line may be repeated any number of times, with a different (A, B) pair each time.

Default: no LCG pairs are connected by apportionment

LCG_CONNECTION_BY_ELEVATION

```
LCG_CONNECTION_BY_ELEVATION     Integer A  Integer B
```

Specifies that AlgoMesh should use the *connection by elevation* method to laterally connect cells of Lateral Connection Group A with cells of Lateral Connection Group B. See the advanced topic Lateral Connection Groups for more information.

This line may be repeated any number of times, with a different (A, B) pair each time.

Default: no LCG pairs are connected by elevation

LCG_CONNECTION_BY_LAYER

```
LCG_CONNECTION_BY_LAYER         Integer A  Integer B
```

Specifies that AlgoMesh should use the standard *connection by layer* method to laterally connect cells of Lateral Connection Group A with cells of Lateral Connection Group B. See the advanced topic Lateral Connection Groups for more information.

This line may be repeated any number of times, with a different (A, B) pair each time.

Default: only cells with the same LCG value are connected by layer

6.7 Meshing Control File (.mcf)

The MCF provides parameters that affect the mesh generation process, including a choice of mesh generation method, global control over element sizing, and quality constraints. It may be used as an input to the AlgoMeshCmd.exe command line tool installed with AlgoMesh. Note that not all options are available when using the mesh generator via the command-line tool; for the best experience, the use of the AlgoMesh user interface for mesh generation is recommended.

All available parameters and their default values are given in the sample MCF below:

```
# General parameters
MESH_GENERATION_METHOD                     REFINEMENT
CELL_CENTRE_TYPE                   CIRCUMCENTRE_IF_POSSIBLE
GNC_CALCULATION_METHOD                     NONE
IDW_POWER                                  1.0

# Delaunay refinement parameters
MIN_ANGLE_USED                             1
MIN_ANGLE                                  26.0
MAX_AREA_USED                              0
MAX_AREA                                   0.0
MAX_TRIANGLES_USED                         1
MAX_TRIANGLES                              2000000
AVOID_SPLITTING_SMALL_CONSTRAINED_ANGLES   1
SEDITIOUS_EDGE_EPSILON                     1.0
BAD_TRIANGLE_QUEUES                        2100
LLOYD_ITERATIONS_PER_REFINEMENT            0
TOTAL_REFINEMENT_ITERATIONS                1
FINISH_MODE                                REFINE_LAST

# Edge length field generation parameters
ELFG_GENERATE_PROPERTY_TIN                 0
ELFG_DISTANCE_FACTOR                       0.4
ELFG_BOUNDARY_FEATURE_SIZE_FACTOR          2.0
ELFG_GRID_SAMPLES                          25000
ELFG_BOUNDARY_RESAMPLING_FACTOR            0.5
ELFG_MIN_RESAMPLING_INTERVAL_USED          0
ELFG_MIN_RESAMPLING_INTERVAL               0.1
ELFG_MAX_EDGE_LENGTH_AT_POINTS_USED        0
ELFG_MAX_EDGE_LENGTH_AT_POINTS             1.0
ELFG_MAX_EDGE_LENGTH_AT_EDGES_USED         0
ELFG_MAX_EDGE_LENGTH_AT_EDGES              1.0
ELFG_SAVE_PROPERTY_TIN_MDD_USED            0
ELFG_SAVE_PROPERTY_TIN_MDD_PATH            edgelength.mdd

# Optimisation parameters
OPT_GLOBAL_MAX_EDGE_LENGTH_USED            0
OPT_GLOBAL_MAX_EDGE_LENGTH                 1.0
OPT_QUALITY_PRESET                         3
```

```
OPT_SIZING_RATIO_REDUCTION_FACTOR          0.5774
OPT_TARGET_MOVE_RATIO_TYPE_REFINE          MAX
OPT_TARGET_MOVE_RATIO_REFINE               0.03
OPT_TARGET_MOVE_RATIO_TYPE_FINAL_RELAX     MAX
OPT_TARGET_MOVE_RATIO_FINAL_RELAX          0.01
OPT_QUADRATURE_COUNT_REFINE                24
OPT_QUADRATURE_COUNT_FINAL_RELAX           12
OPT_MAX_REFINE_ITERATIONS_USED             0
OPT_MAX_REFINE_ITERATIONS                  25
OPT_MAX_LLOYD_ITERATIONS_USED_REFINE       0
OPT_MAX_LLOYD_ITERATIONS_REFINE            10
OPT_MAX_LLOYD_ITERATIONS_USED_FINAL_RELAX  0
OPT_MAX_LLOYD_ITERATIONS_FINAL_RELAX       20
```

Each parameter is specified together with its value on a single line. The order of parameters may be changed arbitrarily, and all parameters are optional (the values above are used as defaults for omitted parameters). Use of whitespace and separators is the same as for MDD files.

The parameters above are grouped according to the process they affect: general (all processes), Delaunay refinement mesh generation, edge length field generation and optimisation-based mesh generation. Details on each parameter are given in the respective sections below.

General Parameters

General parameters control which mesh generation method is used and certain output options that are independent of the mesh generation method.

MESH_GENERATION_METHOD

```
MESH_GENERATION_METHOD          REFINEMENT or OPTIMISATION
```

Determines whether Delaunay refinement (REFINEMENT) or the multi-level optimisation process (OPTIMISATION) is used for mesh generation. If REFINEMENT is used, the Delaunay refinement set of MCF parameters are taken and the optimisation parameters are ignored. Conversely, if OPTIMISATION is used, the optimisation set of parameters are taken and the Delaunay refinement parameters are ignored.

Default: REFINEMENT

CELL_CENTRE_TYPE

```
CELL_CENTRE_TYPE                CENTROID_ONLY
                                or CIRCUMCENTRE_IF_POSSIBLE
```

Specifies the type of cell centres to use in writing out MODFLOW-USG files for triangular meshes (and in calculating ghost-node correction factors). Voronoi mesh cell centres are not affected (these are always the dual Delaunay vertex where possible, and the centroid otherwise).

- **CENTROID_ONLY**
  Uses the centroid of each triangle as its cell centre:

  $$\left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3}\right)$$

  where (x_1, y_1), (x_2, y_2) and (x_3, y_3) are the coordinates of the triangle's three vertices.
- **CIRCUMCENTRE_IF_POSSIBLE**
  (recommended)
  Uses the circumcentre of a triangle (centre of the circle passing through its three vertices) as the cell centre if it is contained within the triangle, otherwise uses the centroid. The circumcentre of a triangle is located at the intersection of the perpendicular bisectors of its three edges, making it ideal for MODFLOW-USG's Control Volume Finite Difference (CVFD) calculations and removing the need for ghost nodes. However, the circumcentre is not always located within the triangle, and therefore cannot always be used; for this reason, the centroid is used as a fall-back for those triangles that do not contain their circumcentre.

Default: CIRCUMCENTRE_IF_POSSIBLE

GNC_CALCULATION_METHOD

```
GNC_CALCULATION_METHOD          PROJECTED_FRACTIONAL_DISTANCE
                                or IDW_VERTEX_NEIGHBOURS
                                or IDW_NEIGHBOURS
                                or NONE
```

Specifies the method used to calculate ghost node correction factors when writing the GNC package for MODFLOW-USG.

- **PROJECTED_FRACTIONAL_DISTANCE**
  (Triangular and Voronoi mesh output)
  Calculates head contributions for a link from a given cell T to a neighbouring cell A (link TA) from the other cell edge neighbours (those sharing an edge with the cell) based on the projected distance of link TA's ghost node along the links from T to the respective edge neighbours.

  This method of calculation is relatively fast, but only takes into account cell edge neighbours; cells that are nearby but which only share a vertex with the given cell are not taken into account.
- **IDW_VERTEX_NEIGHBOURS**
  (Triangular mesh output only)
  Calculates head contributions for a link from a given triangle T to a neighbouring triangle A from all triangles sharing at least one vertex with T. Each contributing triangle's factor is calculated using inverse distance weighting, based on the distance of that triangle's cell centre to the ghost node location for link TA:

$$w_i = \frac{dist(g, c_i)^{-IDW\_POWER}}{\sum_i dist(g, c_i)^{-IDW\_POWER}}$$

where w_i is the computed weight for contributing triangle i, $dist(g, c_i)$ is the Euclidean distance from the ghost node location to the cell centre of contributing triangle i, and IDW_POWER is a configurable exponent (see parameter IDW_POWER below).

Note that as this calculation method considers all vertex neighbours of each triangle, it can take some time to compute ghost-node correction factors. It may also add a significant number of new connections to the MODFLOW-USG matrix (as connections between vertex neighbours are introduced, in addition to the usual edge neighbour connections). This may make it harder for the simulation to converge, as well as slowing it down.

- **IDW_NEIGHBOURS**
  (Triangular and Voronoi mesh output)
  As for the IDW_VERTEX_NEIGHBOURS method, but only cells sharing an edge with the given cell are considered, instead of all cells sharing a vertex.
- **NONE**
  (recommended)
  No ghost node correction factors will be calculated (and hence no GNC package will be written).

Default: NONE

IDW_POWER

```
IDW_POWER                                    Real number
```

Sets the exponent in the inverse-distance weighting formula when either the IDW_VERTEX_NEIGHBOURS or IDW_NEIGHBOURS ghost-node correction factor calculation method is used. Ignored for other ghost-node correction factor calculation methods.

Default: 1.0

Delaunay Refinement Parameters

Delaunay refinement is a greedy process wherein points (and hence triangles) are added into a triangular mesh one by one such that certain “bad” triangles or edges are eliminated and replaced with higher-quality triangles or edges. An edge is considered bad if the angle opposite the edge in an adjacent triangle is greater than 90 degrees. A triangle may be considered bad if it contains a small angle or if it is too large. The process continues until the mesh contains no bad triangles or edges, or some alternative stopping criterion is met (see for example MAX_TRIANGLES below).

The parameters in this section control the Delaunay refinement process.

MIN_ANGLE_USED

```
MIN_ANGLE_USED                               0 or 1
```

```
```

Enables (1) or disables (0) the minimum angle criterion. When this is enabled, AlgoMesh will attempt to refine the mesh until all triangles have angles of at least MIN_ANGLE.

Default: Enabled (1)

MIN_ANGLE

```
MIN_ANGLE                                    0.0 to 60.0
```

Sets the desired minimum angle, in degrees, for triangles in the generated mesh. Ignored if MIN_ANGLE_USED is disabled (0).

Mesh refinement is fairly sensitive to changes in MIN_ANGLE, particularly above ~30 degrees. It is recommended that this parameter is used in conjunction with MAX_TRIANGLES (see below), to ensure that the mesh refinement process does not continue indefinitely.

Default: 26.0

MAX_AREA_USED

```
MAX_AREA_USED                                0 or 1
```

Enables (1) or disabled (0) the maximum area criterion. When this is enabled, AlgoMesh will refine the mesh until all triangles have an area of at most MAX_AREA.

Default: Disabled (0)

MAX_AREA

```
MAX_AREA                                     > 0.0
```

Sets the desired maximum triangular cell area. Units (L2) are not specified and depend directly on the coordinates specified in the mesh domain. The final generated mesh is guaranteed to have triangles with area less than or equal to MAX_AREA (unless the MAX_TRIANGLES criterion stops mesh refinement early; see below). Ignored if MAX_AREA_USED is disabled (0).

Default: Unused (0.0)

MAX_EDGE_LENGTH_USED

```
MAX_EDGE_LENGTH_USED                         0 or 1
```

Enables (1) or disabled (0) the maximum edge length criterion. When this is enabled, AlgoMesh will refine the mesh until all triangles have a longest edge length of at most MAX_EDGE_LENGTH.

Default: Disabled (0)

MAX_EDGE_LENGTH

```
MAX_EDGE_LENGTH                                   > 0.0
```

Sets the desired maximum triangular edge length. Units (L) are not specified and depend directly on the coordinates specified in the mesh domain. The final generated mesh is guaranteed to have triangles with longest edge length less than or equal to MAX_EDGE_LENGTH (unless the MAX_TRIANGLES criterion stops mesh refinement early; see below). Ignored if MAX_EDGE_LENGTH_USED is disabled (0).

Default: Unused (0.0)

MIN_AREA_USED

```
MIN_AREA_USED                                     0 or 1
```

Enables (1) or disabled (0) the minimum area criterion. When this is enabled, AlgoMesh will not choose triangles with area less than or equal to MIN_AREA as candidates for refinement.

Default: Disabled (0)

MIN_AREA

```
MIN_AREA                                          > 0.0
```

Sets the desired minimum triangular cell area. Units (L2) are not specified and depend directly on the coordinates specified in the mesh domain. This is a guide to the mesh refinement algorithm only; the final generated mesh is NOT guaranteed to have only triangles with area greater than or equal to MIN_AREA. Triangles with area less than or equal to MIN_AREA are not chosen as candidates for refinement (but smaller triangles may still be generated by refining other, larger, triangles nearby). Ignored if MIN_AREA_USED is disabled (0).

Default: Unused (0.0)

MIN_EDGE_LENGTH_USED

```
MIN_EDGE_LENGTH_USED                              0 or 1
```

Enables (1) or disabled (0) the minimum area criterion. When this is enabled, AlgoMesh will not choose triangles with longest edge length less than or equal to MIN_EDGE_LENGTH as candidates for refinement.

Default: Disabled (0)

MIN_EDGE_LENGTH

```
MIN_EDGE_LENGTH                       > 0.0
```

Sets the desired minimum triangular edge length. Units (L) are not specified and depend directly on the coordinates specified in the mesh domain. This is a guide to the mesh refinement algorithm only; the final generated mesh is NOT guaranteed to have only triangles with longest edge length greater than or equal to MIN_EDGE_LENGTH. Triangles with longest edge length less than or equal to MIN_EDGE_LENGTH are not chosen as candidates for refinement (but smaller triangles may still be generated by refining other, larger, triangles nearby). Ignored if MIN_EDGE_LENGTH_USED is disabled (0).

Default: Unused (0.0)

MAX_TRIANGLES_USED

```
MAX_TRIANGLES_USED                    1 or 0
```

Enables (1) or disables (0) the maximum number of triangles criterion. When this is enabled, AlgoMesh will terminate Delaunay refinement of the mesh early if the number of triangles reaches MAX_TRIANGLES. It is recommended that this option is turned on with a reasonable upper limit set for MAX_TRIANGLES, to ensure that mesh generation does not run indefinitely (or until memory limits are reached).

Default: Enabled (1)

MAX_TRIANGLES

```
MAX_TRIANGLES                         > 0
```

Sets an approximate upper limit on the number of triangles for the generated mesh. With MAX_TRIANGLES_USED enabled (1), AlgoMesh will terminate Delaunay refinement of the mesh early if the number of triangles reaches MAX_TRIANGLES. If the other criteria (MIN_ANGLE and MAX_AREA) are satisfied before MAX_TRIANGLES is reached, the mesh will contain fewer than MAX_TRIANGLES cells.

The limit is approximate for three reasons:

1. In each iteration of the Delaunay mesh refinement process, AlgoMesh inserts a new point into the triangulation, which in general produces two additional triangles, rather than one (the

exception is at the edges of the mesh, but see number 3 below).

2. AlgoMesh maintains a Constrained Delaunay Triangulation (CDT) at all times. This means that the entire interior of the convex hull of the mesh domain is covered with triangles – even those parts that have been marked outside the domain. Although these triangles are excluded from refinement, they are still counted towards the total number of triangles in the mesh when testing against the MAX_TRIANGLES criterion.
3. AlgoMesh maintains additional, invisible triangles (called “ghost triangles”) around the outside of the convex hull of the mesh domain at all times, as part of its internal mesh representation. When the mesh is tested against the MAX_TRIANGLES criterion, it uses the total number of triangles, including ghost triangles. This means that, even if the entire convex hull is marked as inside the mesh domain, the number of real cells in the final mesh may be less than MAX_TRIANGLES by the number of edges present in the convex hull of the mesh.

Default: 2000000

AVOID_SPLITTING_SMALL_CONSTRAINED_ANGLES

```
AVOID_SPLITTING_SMALL_CONSTRAINED_ANGLES  0 or 1
```

When this flag is enabled and MIN_ANGLE_USED is also enabled, edges adjacent to a constrained input angle that is smaller than MIN_ANGLE will not be split during refinement, unless there is also an unconstrained small angle in a triangle containing the edge. When this flag is disabled (or MIN_ANGLE_USED is disabled), any edges with an opposite angle larger than 90 degrees may be split to guarantee a fully Delaunay triangulation (unless they are seditious; see SEDITIOUS_EDGE_EPSILON description below).

Default: Enabled (1)

SEDITIOUS_EDGE_EPSILON

```
SEDITIOUS_EDGE_EPSILON                > 0.0
```

When an edge is considered for splitting during refinement, it is first checked to see if it is likely to cause an infinite refinement loop due to a small constrained angle in the input domain. This check involves comparing the lengths of pairs of split edges to determine whether they are approximately equal. The value of SEDITIOUS_EDGE_EPSILON is used as a tolerance for these comparisons.

A larger value will lead to less refinement within and around small constrained domain angles. A smaller value will refine further, possibly over-refining. If you find large, poor-quality triangles in the vicinity of acute constrained angles in your mesh, try decreasing this value by a factor of 10 at a time. If you instead find too many very small triangles around such angles, try increasing the value by a factor of 10 at a time.

Default: 1.0

BAD_TRIANGLE_QUEUES

```
BAD_TRIANGLE_QUEUES                          > 0
```

During the refinement process, bad triangles (too large or with a small angle) are considered one by one, approximately in order of their shortest edge length. The ordering is approximated by placing triangles into a finite number of queues according to the log of their shortest edge length; this allows for faster processing than a strict ordering, without a major compromise on mesh quality.

The default value is generally reasonable, and the parameter tends to be fairly insensitive to changes.

Default: 2100

LLOYD_ITERATIONS_PER_REFINEMENT

```
LLOYD_ITERATIONS_PER_REFINEMENT          0
```

Specifies the number of Lloyd relaxation iterations that should be applied to the vertices of the mesh after Delaunay refinement is completed.

Each Lloyd iteration moves every unconstrained vertex in the mesh to the centroid of its dual bounded Voronoi cell. This tends to create triangles that are closer to equilateral (better quality) in unconstrained areas of the mesh. If an edge length property TIN mapping is used, the Lloyd iteration takes a weighted centroid based on the given edge length field; otherwise, it assumes uniform sizing is desired and tends to equalise element size throughout the mesh.

If Lloyd iterations are used, it is recommended that they be interleaved between Delaunay refinement steps. AlgoMesh does this by default; see parameters TOTAL_REFINEMENT_ITERATIONS and FINISH_MODE below for more control over the process.

As an alternative to refinement followed by Lloyd relaxation, consider using the multi-level optimisation process (set MESH_GENERATION_METHOD to OPTIMISATION), which incorporates similar techniques in a structured way to produce high-quality meshes. This feature is available from version 1.0.3 of AlgoMesh onwards.

Overall, mesh quality is often improved by using Lloyd iterations, but at the cost of more cells (due to additional Delaunay refinement steps) and slower mesh generation (Lloyd iterations are usually more computationally expensive than Delaunay refinement, as the entire mesh is re-triangulated after each iteration).

Values of 1-5 are generally feasible for large meshes (millions of cells), while larger values (20 and upwards) are reasonable for much smaller meshes (tens of thousands of cells or less).

A value of 0 indicates that no Lloyd relaxation will take place.

Default: 0 (no Lloyd relaxation)

TOTAL_REFINEMENT_ITERATIONS

```
TOTAL_REFINEMENT_ITERATIONS              >= 0
```

Sets the number of times Delaunay refinement combined with Lloyd relaxation will be executed. Mesh generation proceeds according to the following process:

```
1. Do
2.     Perform Delaunay mesh refinement
3.     Do
4.         Move vertices according to Lloyd relaxation procedure
5.         Re-triangulate mesh
6.     Repeat LLOYD_ITERATIONS_PER_REFINEMENT times
7. Repeat TOTAL_REFINEMENT_ITERATIONS times
8. Perform final Delaunay mesh refinement if necessary
   (if FINISH_MODE = REFINE_LAST and TOTAL_REFINEMENT_ITERATIONS > 0)
```

Note that setting TOTAL_REFINEMENT_ITERATIONS to a value greater than 1 has no effect if Lloyd iterations are not used, as Delaunay mesh refinement completes only when its criteria are met (meaning that if a second refinement step is immediately executed after the first, it will have nothing to do).

Setting TOTAL_REFINEMENT_ITERATIONS to 0 causes AlgoMesh to construct the Constrained Delaunay Triangulation (CDT) of the mesh domain, but to perform no refinement on it (not recommended in general).

Default: 1

FINISH_MODE

```
FINISH_MODE                         REFINE_LAST or LLOYD_LAST
```

Controls the final step taken by AlgoMesh in generating the mesh.

- **REFINE_LAST**
  (Recommended)
  Indicates that the final step in mesh generation should be Delaunay mesh refinement. If at least one refinement iteration and at least one Lloyd iteration are performed, AlgoMesh will perform one additional Delaunay mesh refinement step after all other refinement steps are complete.
- **LLOYD_LAST**
  Indicates that the final step in mesh generation should be Lloyd relaxation (if enabled). The additional Delaunay mesh refinement after all other refinement steps are complete is omitted.

Default: REFINE_LAST

Edge Length Field Generation Parameters

AlgoMesh is able to automatically generate an element sizing field (desired edge length property TIN) according to a parameterised sizing function. This is not a mesh generation method in itself, but rather a pre-processing step that may be used to specify how element sizes should vary over a mesh domain. It may be used with either the Delaunay refinement (REFINEMENT) or the multi-level optimisation (OPTIMISATION) approach. **It is recommended that edge length field generation is enabled when the multi-level optimisation approach is used**, unless a precise, uniform element size is desired over the entire mesh.

The edge length values are generated as a function of the so-called *local feature size* at boundaries (constrained points and line segments) and the distance from these boundaries. The boundary local feature size is a measure of the proximity of one part of the mesh domain’s boundary to others (it is smaller in areas where many boundaries are close to one another, or are sampled by many points, and larger in areas with few boundaries that are sparsely sampled).

The parameters in this section control the edge length field generation process. Usually, the first three parameters (ELFG_GENERATE_PROPERTY_TIN, ELFG_DISTANCE_FACTOR, ELFG_BOUNDARY_FEATURE_SIZE_FACTOR) have the most impact on the resulting mesh. ELFG_MAX_EDGE_LENGTH_AT_POINTS and ELFG_MAX_EDGE_LENGTH_EDGES (and their respective _USED flags) are also useful for controlling element sizes around features in the mesh domain. The remainder can be left at their defaults unless fine-tuning is required.

ELFG_GENERATE_PROPERTY_TIN

```
ELFG_GENERATE_PROPERTY_TIN            1 or 0
```

Enables (1) or disables (0) the edge length field generation process. When this is enabled, AlgoMesh will automatically generate a property TIN containing a sizing field and map it to the desired mesh cell edge length property.

Default: Disabled (0)

ELFG_DISTANCE_FACTOR

```
ELFG_DISTANCE_FACTOR                  >= 0.0
```

The multiplier of distance from a boundary on the desired edge length calculation. A zero value here results in a uniform element size of the minimum boundary local feature size throughout the whole mesh. A small positive value results in a gradual change in element size moving from small elements at the boundaries to larger elements in empty areas, while larger values make this change in size occur much more suddenly.

Using too small a value may result in a mesh with a large number of elements. Using too large a value may reduce element quality close to the boundaries, as they are stretched to accommodate the rapid increase in element size away from the boundaries.

Values of 0.1 to 1.0 are recommended.

Default: 0.4

ELFG_BOUNDARY_FEATURE_SIZE_FACTOR

```
ELFG_BOUNDARY_FEATURE_SIZE_FACTOR     > 0.0
```

The multiplier of boundary local feature size on the desired edge length calculation. A smaller value results in a larger number of smaller elements being generated along the boundaries, while a larger value results in fewer, larger elements at the boundaries.

Using too small a value may result in a mesh with a large number of elements. Using too large a value may reduce element quality close to the boundaries, as elements may not be created small enough to obtain a good shape between adjacent boundaries.

Values of 0.5 to 3.0 are recommended.

Default: 2.0

ELFG_GRID_SAMPLES

```
ELFG_GRID_SAMPLES                     >= 0
```

The property TIN that is generated for the edge length field contains three types of points: boundary points (present in the original mesh domain), poles (approximations of the so-called *medial axis* of the domain) and grid points spaced regularly across the domain. These grid points are used to obtain a better approximation of the local feature size across the domain. The value specified for ELFG_GRID_SAMPLES determines approximately how many grid points will be included in the final property TIN. The actual number used may vary based on how much of the mesh domain bounding box is flagged as outside the mesh domain, and based on the square root calculation used to obtain regular spacing between grid points.

It is normally fine to leave this value at its default. You may consider increasing it if the edge length field generated appears to give bad results with some complex boundary configurations, or decreasing it if the resulting property TIN is too large relative to the size of the mesh and you want to speed up the mesh generation process.

Default: 25000

ELFG_BOUNDARY_RESAMPLING_FACTOR

```
ELFG_BOUNDARY_RESAMPLING_FACTOR       > 0.0
```

Prior to calculating approximate local feature size values, the boundaries of the original mesh domain are resampled using a larger number of points where boundaries are close to one another. The value of the ELFG_BOUNDARY_RESAMPLING_FACTOR parameter is a multiplier on the spacing between resampled points. The smaller this value, the more points inserted into the property TIN (which may slow down mesh generation somewhat), and the smaller the local feature size values will be, particularly near corners and points that are very close together (leading to more elements in the mesh).

Values of 0.1 to 2.0 are recommended.

Default: 0.5

ELFG_MIN_RESAMPLING_INTERVAL_USED

```
ELFG_MIN_RESAMPLING_INTERVAL_USED     0 or 1
```

Determines whether the ELFG_MIN_RESAMPLING_INTERVAL value is used (1) or not (0) during boundary resampling.

Default: Disabled (0)

ELFG_MIN_RESAMPLING_INTERVAL

```
ELFG_MIN_RESAMPLING_INTERVAL          > 0.0
```

The minimum interval between any two resampled points produce during the boundary resampling phase of edge length field generation (see the description of ELFG_BOUNDARY_RESAMPLING_FACTOR). If this value is specified and ELFG_MIN_RESAMPLING_INTERVAL_USED is 1, additional points will not be added such that two adjacent points are less than this distance apart.

Default: 0.1

ELFG_MAX_EDGE_LENGTH_AT_POINTS_USED

```
ELFG_MAX_EDGE_LENGTH_AT_POINTS_USED   0 or 1
```

Determines whether desired edge length values at lone constrained points are limited by the ELFG_MAX_EDGE_LENGTH_AT_POINTS value (1) or not (0).

Default: Disabled (0)

ELFG_MAX_EDGE_LENGTH_AT_POINTS

```
ELFG_MAX_EDGE_LENGTH_AT_POINTS        > 0.0
```

The maximum desired edge length value to set at any constrained input point in the mesh domain that does not belong to a constrained edge. If the calculation of desired edge length at a lone constrained point based on local feature size and distance from boundaries is larger than this value, this value is used instead. This is useful for specifying element sizing in the vicinity of groundwater wells, for example. The value used here is an absolute length value, and depends on the scale of the mesh domain and the desired resulting element sizes.

Default: 1.0

ELFG_MAX_EDGE_LENGTH_AT_EDGES_USED

```
ELFG_MAX_EDGE_LENGTH_AT_EDGES_USED    0 or 1
```

Determines whether desired edge length values along constrained edges are limited by the ELFG_MAX_EDGE_LENGTH_AT_EDGES value (1) or not (0).

Default: Disabled (0)

ELFG_MAX_EDGE_LENGTH_AT_EDGES

```
ELFG_MAX_EDGE_LENGTH_AT_EDGES         > 0.0
```

The maximum desired edge length value to set at any constrained edge in the mesh domain. If the calculation of desired edge length along an edge based on local feature size and distance from boundaries is larger than this value, this value is used instead. This is useful for ensuring a maximum element size in the vicinity of all boundary features in the mesh domain (regardless of point spacing along the boundaries or proximity to other boundaries). The value used here is an absolute length value, and depends on the scale of the mesh domain and the desired resulting element sizes.

Default: 1.0

ELFG_SAVE_PROPERTY_TIN_MDD_USED

```
ELFG_SAVE_PROPERTY_TIN_MDD_USED       0 or 1
```

Determines whether the desired edge length property TIN generated is saved to an .MDD file for later re-use (1) or not (0).

Default: Disabled (0)

ELFG_SAVE_PROPERTY_TIN_MDD_PATH

```
ELFG_SAVE_PROPERTY_TIN_MDD_PATH       0 or 1
```

A path and filename to save the property TIN created by the edge length field generation process. This value is ignored if ELFG_SAVE_PROPERTY_TIN_MDD_USED is 0. The property TIN is saved in .MDD format so that it may be later re-used as a desired cell edge length mapping without repeating the generation process, which can be time-consuming for complex mesh domains.

Default: edgelength.mdd

Optimisation Parameters

AlgoMesh provides a multi-level optimisation process as an alternative to the default greedy Delaunay refinement approach. This process can be much slower than Delaunay refinement, but can also produce higher-quality meshes with an appropriate set of parameters specified. It is activated by setting MESH_GENERATION_METHOD to OPTIMISATION.

The optimisation-based mesh generation method operates initially by inserting a set of new points that eliminate triangles and edges that are too large according to a given desired edge length field and a target sizing ratio (actual size / desired size). The placement of these points is performed in a similar way to regular Delaunay refinement. The target sizing ratio starts out at a relatively large value, such that only the largest elements are split, and is iteratively reduced by a given factor, effectively limiting each iteration to adding a small subset of possible points. After each set of points is added to the mesh, a weighted Lloyd relaxation process is applied to the mesh to bring the elements closer to the distribution of edge lengths in the given edge length field. Once an iteration is executed in which no more points are added, a final Lloyd relaxation process is performed, with tighter stopping criteria than those performed during refinement.

Contrasted with adding all points at once to satisfy the desired edge length field and following it with Lloyd relaxation, this controlled, multi-level approach tends to produce meshes that contain fewer elements of higher-quality. The quality of elements is heavily dependent on the given edge length field, however, so **it is highly recommended that the automated edge length field generation process is enabled when using this method** (see the Edge Length Field Generation Parameters section for details).

The parameters in this section may be used to control the multi-level optimisation process. In general, it is enough to set OPT_QUALITY_PRESET and leave the remainder of the parameters at their defaults, unless fine-tuning is required. The OPT_GLOBAL_MAX_EDGE_LENGTH and OPT_GLOBAL_MAX_EDGE_LENGTH_USED parameters may be used to set a global maximum element size limit, or to set a uniform element size if no edge length property TIN mapping is provided.

OPT_GLOBAL_MAX_EDGE_LENGTH_USED

```
OPT_GLOBAL_MAX_EDGE_LENGTH_USED       0 or 1
```

Enables (1) or disables (0) use of a maximum edge length value for all elements in the mesh.

Default: Disabled (0)

OPT_GLOBAL_MAX_EDGE_LENGTH

```
OPT_GLOBAL_MAX_EDGE_LENGTH            > 0.0
```

Sets the maximum edge length for all elements in the mesh. This is taken as a single uniform desired edge length value if no edge length property TIN mapping is given (and edge length field generation is disabled). This value is ignored if the flag OPT_GLOBAL_MAX_EDGE_LENGTH_USED is set to 0. The value used here is an absolute length value, and depends on the scale of the mesh domain and the desired resulting element sizes.

Default: 1.0

OPT_QUALITY_PRESET

```
OPT_QUALITY_PRESET                1 to 6
```

An integer between 1 and 6 (inclusive), which sets other optimisation parameters to reasonable defaults for a general desired level of quality. Level 1 provides the fastest mesh generation but may produce many more elements than necessary, and potentially of a lower quality. Level 6 provides a higher-quality mesh containing fewer elements, at the cost of taking a very long time to generate a mesh. Levels 2 to 5 provide a sliding scale in between these two extremes to trade off meshing quality and speed.

Note that the term “quality” is used loosely here; the interaction between the optimisation parameter values and mesh element quality is indirect and complex. The actual quality of the resulting mesh is heavily dependent on the given desired edge length field. Higher quality preset levels are more likely to produce meshes that closely match the input edge length field.

The quality preset levels set parameter defaults as in the following table:

	OPT_QUALITY_PRESET					
	1	**2**	**3**	**4**	**5**	**6**
OPT_SIZING_RATIO _REDUCTION_FACTOR	0.5774 (1/√3)	0.5774 (1/√3)	0.5774 (1/√3)	0.75	0.9	0.95
OPT_TARGET_MOVE_RATIO_TYPE _REFINE	RMS	RMS	MAX	MAX	MAX	MAX
OPT_TARGET_MOVE_RATIO _ REFINE	0.03	0.005	0.03	0.03	0.02	0.01
OPT_TARGET_MOVE_RATIO_TYPE _ FINAL_RELAX	RMS	RMS	MAX	MAX	MAX	MAX
OPT_TARGET_MOVE_RATIO _FINAL_RELAX	0.01	0.001	0.01	0.01	0.01	0.005
OPT_QUADRATURE_COUNT _REFINE	6	24	24	24	64	128
OPT_QUADRATURE_COUNT _FINAL_RELAX	6	12	12	12	64	128

Default: 3

OPT_SIZING_RATIO_REDUCTION_FACTOR

```
OPT_SIZING_RATIO_REDUCTION_FACTOR     > 0.0 and < 1.0
```

Sets the factor by which target sizing ratio is multiplied for each new iteration of the optimisation process. The closer this value is to 1.0, the fewer points are added at each iteration prior to Lloyd relaxation.

A lower value results in faster mesh generation but a sub-optimal mesh (more elements than necessary and a poorer fit to the desired edge length field). A higher value results in a better mesh (fewer elements and closer fit to the desired edge length field) but slower mesh generation.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_TARGET_MOVE_RATIO_TYPE_REFINE

```
OPT_TARGET_MOVE_RATIO_TYPE_REFINE     MAX or RMS
```

Each optimisation iteration, Lloyd relaxation is repeatedly executed until vertices of the triangular mesh are moved less than a certain threshold. For each vertex, this threshold is defined as a fraction of the width of its dual Voronoi cell. This movement threshold may be enforced either such that the maximum movement of any vertex must be lower than the given fraction of its corresponding Voronoi cell width (MAX), or such that the root-mean-square aggregation of all vertex movement, each as a fraction of its corresponding Voronoi cell, must be lower than the given fraction (RMS). MAX is a stricter condition than RMS. The OPT_TARGET_MOVE_RATIO_TYPE_REFINE parameter specifies the type of threshold (MAX or RMS) for the Lloyd relaxation that is performed after each refinement step (addition of points), and is used in conjunction with OPT_TARGET_MOVE_RATIO_REFINE, which specifies the corresponding threshold value.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_TARGET_MOVE_RATIO_REFINE

```
OPT_TARGET_MOVE_RATIO_REFINE          > 0.0 and < 1.0
```

OPT_TARGET_MOVE_RATIO_REFINE specifies the vertex movement threshold value during refinement (point addition) stages of the optimisation process. See OPT_TARGET_MOVE_RATIO_TYPE_REFINE for details.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_TARGET_MOVE_RATIO_TYPE_FINAL_RELAX

```
OPT_TARGET_MOVE_RATIO_TYPE_FINAL_RELAX MAX or RMS
```

As per OPT_TARGET_MOVE_RATIO_TYPE_REFINE, but for the final relaxation process that occurs after all refinement is complete.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_TARGET_MOVE_RATIO_FINAL_RELAX

```
OPT_TARGET_MOVE_RATIO_FINAL_RELAX     > 0.0 and < 1.0
```

As per OPT_TARGET_MOVE_RATIO_ REFINE, but for the final relaxation process that occurs after all refinement is complete.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_QUADRATURE_COUNT_REFINE

```
OPT_QUADRATURE_COUNT_REFINE           > 0
```

AlgoMesh divides each triangular mesh element into smaller quadrature triangles (and similarly, each edge into smaller quadrature line segments) and interpolates desired edge length values at each quadrature element, to approximate the weighted centroid positions that are used during Lloyd relaxation. The OPT_QUADRATURE_COUNT_REFINE value determines the number of quadrature elements that will be generated for each mesh element.

A lower value will result in faster mesh generation, but a mesh that is a poorer fit to the desired edge length field. A higher value will slow down the mesh generation process, but produce a mesh that fits the edge length field more closely. More complex domains with rapidly varying edge length fields may demand higher numbers of quadrature elements to accurately represent their edge length fields.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_QUADRATURE_COUNT_FINAL_RELAX

```
OPT_QUADRATURE_COUNT_FINAL_RELAX      > 0
```

As per OPT_QUADRATURE_COUNT_REFINE, but for the final relaxation process that occurs after all refinement is complete.

Default: (dependent on OPT_QUALITY_PRESET)

OPT_MAX_REFINE_ITERATIONS_USED

```
OPT_MAX_REFINE_ITERATIONS_USED        0 or 1
```

```

```

Specifies whether a limit on the number of refinement iterations should be applied (1) or not (0). Note that stopping any part of the optimisation process early may result in an incomplete, poor-quality mesh.

Default: Disabled (0)

OPT_MAX_REFINE_ITERATIONS

```
OPT_MAX_REFINE_ITERATIONS            > 0
```

Specifies the maximum number of refinement iterations (point addition + Lloyd relaxation until threshold) that are to be performed by the optimisation process. This value is ignored if OPT_MAX_REFINE_ITERATIONS_USED is 0.

Default: 25

OPT_MAX_LLOYD_ITERATIONS_USED_REFINE

```
OPT_MAX_LLOYD_ITERATIONS_USED_REFINE  0 or 1
```

Specifies whether a limit on the number of Lloyd iterations after each refinement should be applied (1) or not (0). Note that stopping any part of the optimisation process early may result in an incomplete, poor-quality mesh.

Default: Disabled (0)

OPT_MAX_LLOYD_ITERATIONS_REFINE

```
OPT_MAX_LLOYD_ITERATIONS_REFINE       > 0
```

Specifies the maximum number of Lloyd iterations that are to be performed after each refinement (addition of points). If the target movement ratio threshold is not met after this many Lloyd iterations are performed, then the Lloyd relaxation process is stopped and the next refinement iteration begins. This value is ignored if OPT_MAX_LLOYD_ITERATIONS_USED_REFINE is 0.

Default: 25

OPT_MAX_LLOYD_ITERATIONS_USED_FINAL_RELAX

```
OPT_MAX_LLOYD_ITERATIONS_USED_FINAL_RELAX 0 or 1
```

Specifies whether a limit on the number of Lloyd iterations during the final relaxation process should be applied (1) or not (0). Note that stopping any part of the optimisation process early may result in an incomplete, poor-quality mesh.

Default: Disabled (0)

OPT_MAX_LLOYD_ITERATIONS_FINAL_RELAX

```
OPT_MAX_LLOYD_ITERATIONS_FINAL_RELAX  > 0
```

Specifies the maximum number of Lloyd iterations that are to be performed during the final relaxation process (after all refinement has finished). If the target movement ratio threshold is not met after this many Lloyd iterations are performed, then the optimisation process is stopped early. This value is ignored if OPT_MAX_LLOYD_ITERATIONS_USED_FINAL_RELAX is 0.

Default: 25

6.8 AlgoMesh-HydroGeoSphere 2D Mesh Interchange (.ah2)

A triangular mesh may be exported from the AlgoMesh user interface in AH2 format and read in by HydroGeoSphere (HGS) utilities to build a HGS model.

The AH2 format specifies the nodes (triangular vertices) and elements (triangles) in a triangular mesh.

AlgoMesh always uses the Reverse Cuthill-McKee (RCM) algorithm to order nodes in the mesh immediately prior to exporting them in AH2 format.

The format of the AH2 file is as follows:

```
<Number of nodes N>
<Node 1's X Coordinate> <Node 1's Y Coordinate>
<Node 2's X Coordinate> <Node 2's Y Coordinate>
...
<Node N's X Coordinate> <Node N's Y Coordinate>
<Number of elements M>
<Elem. 1 Vertex 1> <Elem. 1 Vertex 2> <Elem. 1 Vertex 3>
<Elem. 2 Vertex 1> <Elem. 2 Vertex 2> <Elem. 2 Vertex 3>
...
<Elem. M Vertex 1> <Elem. M Vertex 2> <Elem. M Vertex 3>
```

The X and Y coordinates of each node are specified as real numbers. A single space character is used to delimit the X and Y values.

Each element is triangular, and is specified by the integer vertex indices of its three nodes. A single space character is used as a delimiter between each vertex index. Vertex indices start from 1 and go up to N, the number of nodes in the mesh.

7 References

Amenta, N., and Bern, M., 1998, Surface reconstruction by Voronoi filtering, in Proc. 14th Symp. Computational Geometry (SCG'98), p. 39-48.

Cheng, S-W., Dey, T.K., and Shewchuk, J.R., 2012, Delaunay Mesh Generation, CRC Press, xii +375 p.

Panday, S., Langevin, C.D., Niswonger, R.G., Ibaraki, M., and Hughes, J.D., 2013, MODFLOW–USG version 1: An unstructured grid version of MODFLOW for simulating groundwater flow and tightly coupled processes using a control volume finite-difference formulation: USGS Techniques and Methods 6-A45.

Shewchuk, J.R., 1996, Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator, in LNCS 1148, Applied computational geometry: towards geometric engineering, Springer-Verlag.

Tournois, J., Alliez, P., and Devillers, O., 2008, Interleaving Delaunay refinement and optimization for 2D triangle mesh generation, in Proc. 16th Int. Meshing Roundtable, Springer Berlin Heidelberg.

8 Trademarks and Third-Party Licences

Parts of the AlgoMesh software utilise third-party libraries, and as such these libraries are included with the AlgoMesh distribution. Some of these libraries have additional licence restrictions, and differing copyright from the rest of AlgoMesh. Please refer to the file THIRD-PARTY-LICENCES.TXT installed with AlgoMesh for details of these licences.