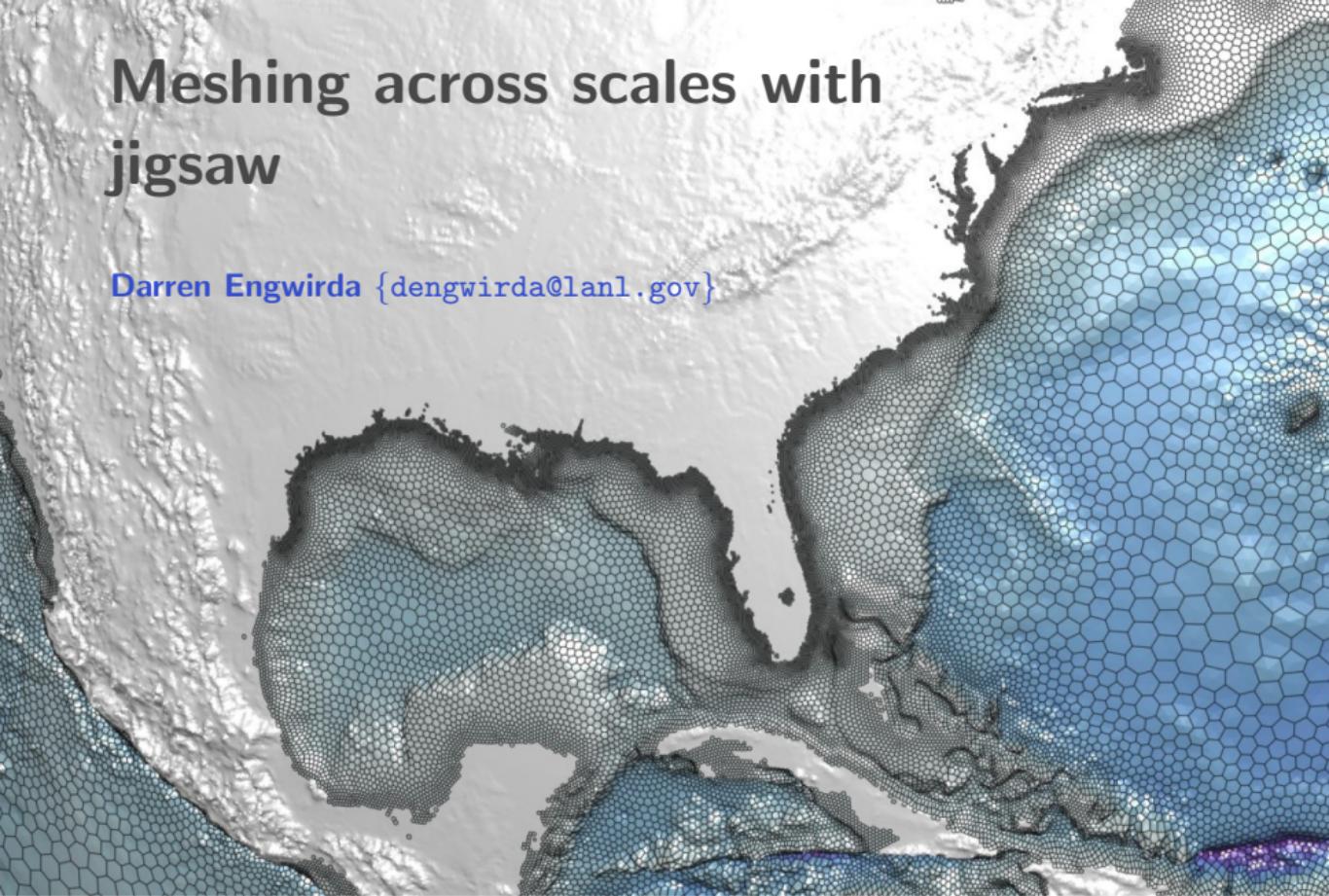


Meshering across scales with jigsaw

Darren Engwirda {dengwirda@lanl.gov}



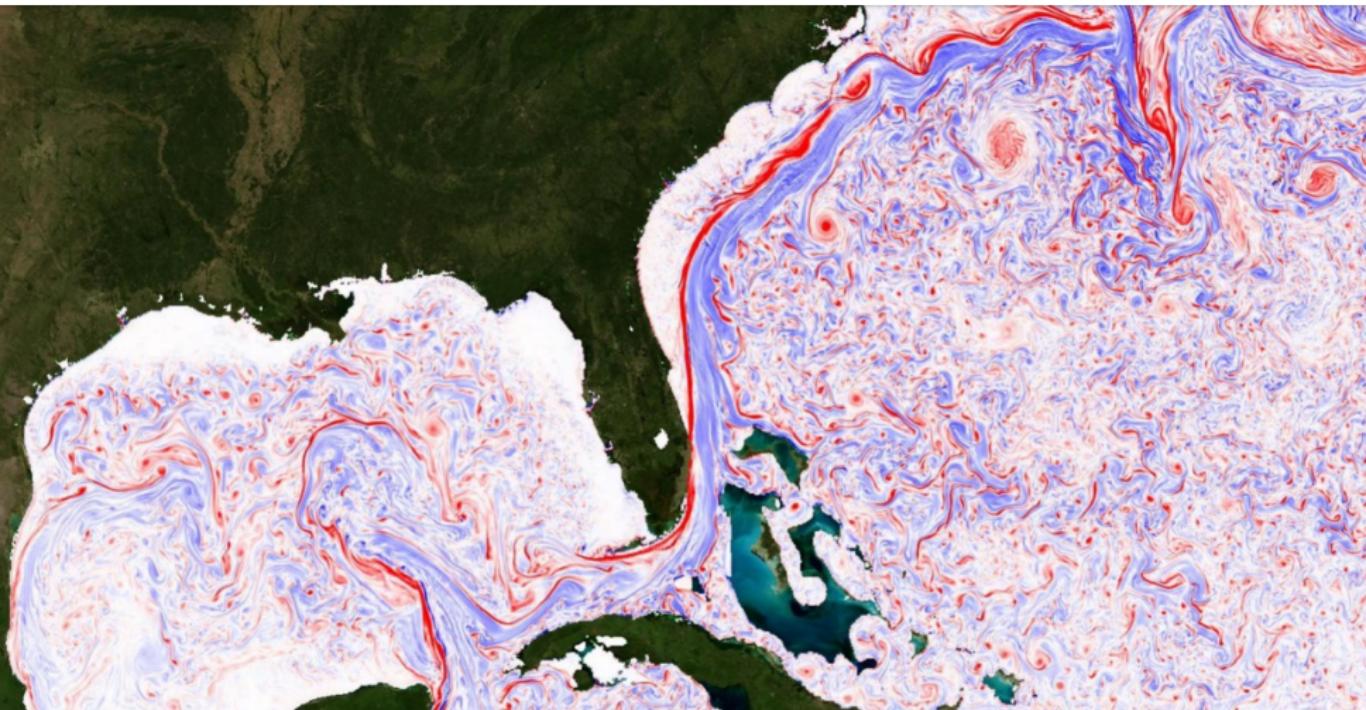
What is 'ocean modelling' target?

3d circulation? 2d storm-surge + tides?

Short-term forecast? Long-term climate sensitivity?

Which model numerical formulation?

All answers strongly effect 'optimal' approach to mesh generation...



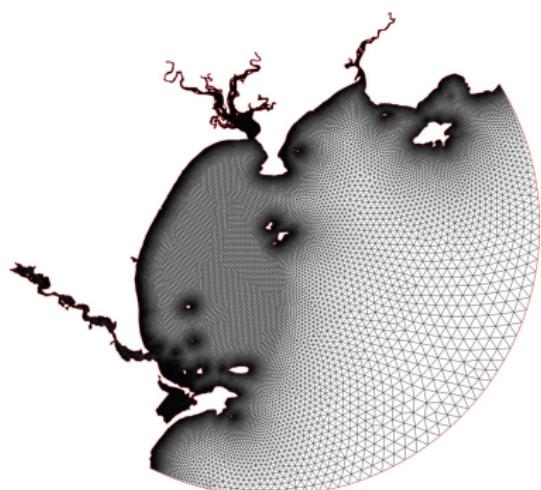
'Unified' meshes for Earth System Models

Support a wide range of Earth System Models through generation of multi-scale regional and global unstructured meshes.

Global meshes



Regional meshes [2d proj.]

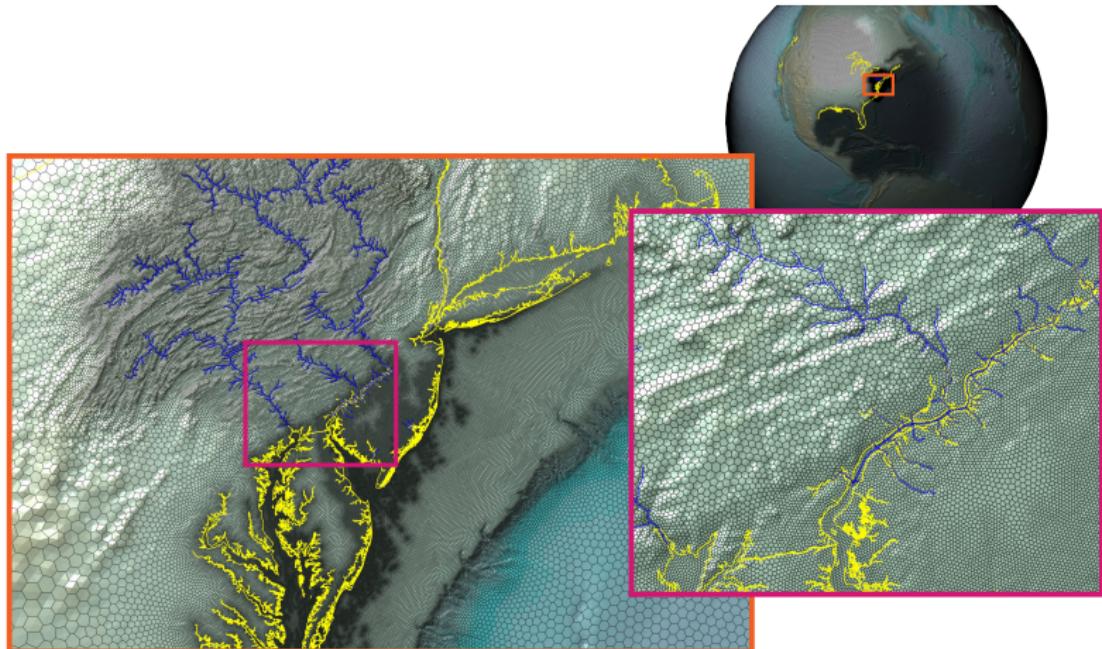


Target variable resolution meshes consisting of **polygonal cells** (Voronoi tessellations) and **triangle cells** (Delaunay meshes).

Used by: E3SM (DOE), COMPAS (CSIRO), FESOM (AWI), and others...

'Unified' meshes for Earth System Models

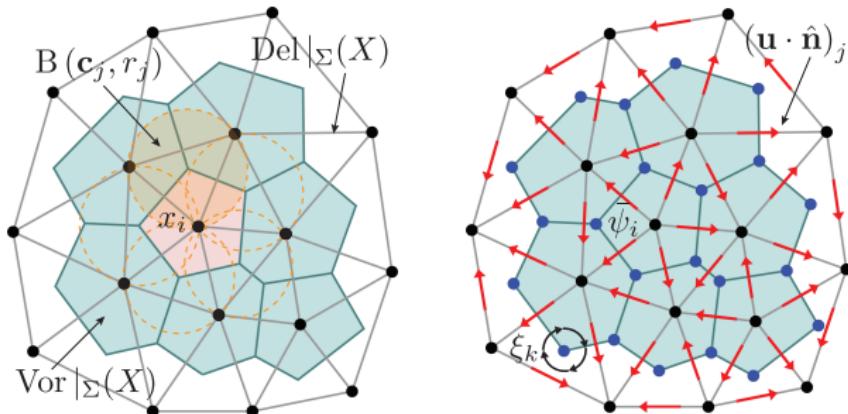
A new 'fully-unified' modelling approach for DOE's Energy Exascale Earth System Model (E3SM) based on common polygonal grid + discretisation strategy:



**Embedded mid-Atlantic coastal-zone: ICoM project, Engwirda et al, 2020.

Common physics + coast + stream-aligned meshes for **ocean**, sea/land-ice, **land-surface**, **hydrology**, and **sub-surface** flow models in E3SM.

E3SM model components are based on *conservative, structure-preserving* numerical methods — an generalisation of GFDL's MOM6 discretisation to unstructured meshes:



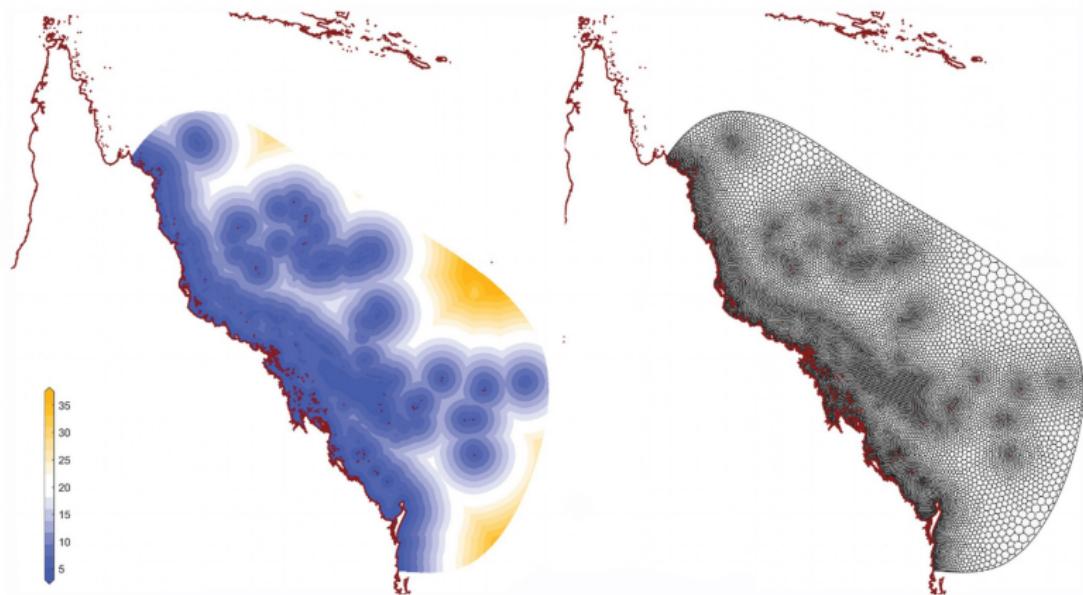
Meshes consist of dual parts: a (Delaunay) triangulation and its orthogonal (Voronoi) **polygonal** decomposition.

Numerics consist of dual parts: cell-centred tracers, staggered edge-aligned momentum, and vertex-centred vorticity.

jigsaw builds dual meshes with tight theoretical guarantees on mesh quality/shape.

jigsaw: meshing across scales

It is not uncommon (ie. occurs every time!) for the geophysical 'geometry' (coastline, river network, watershed) to be **defined at the 'wrong' resolution**.

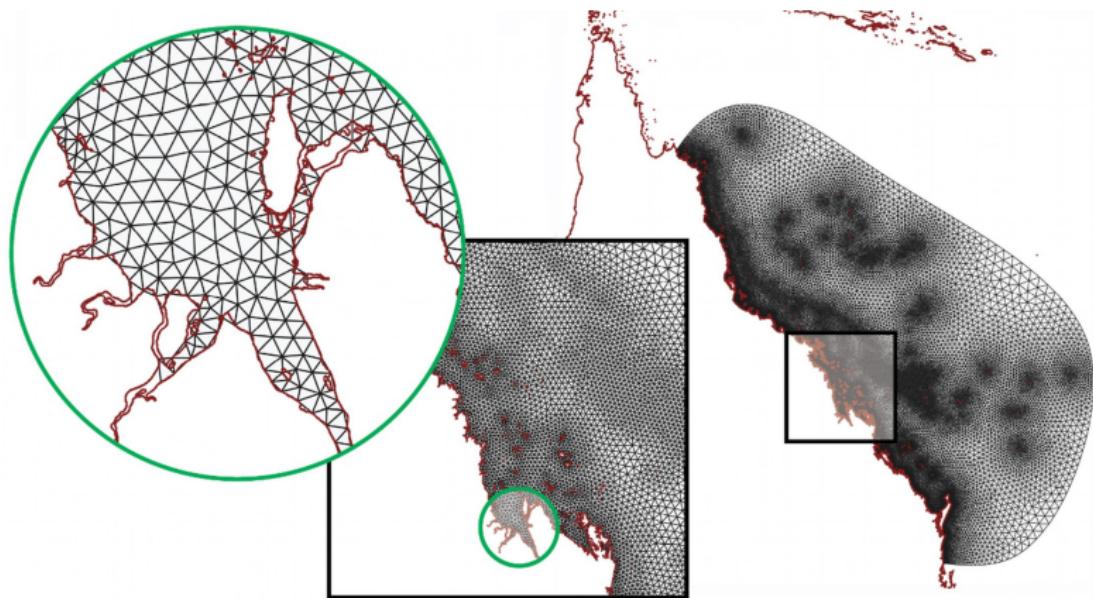


**A coastal unstructured model using Voronoi meshes and C-grid staggering, Engwirda et al, 2020.

Mesh spacing function $\bar{h}(x)$ composed to capture '**scientist-scale**' physics: **wave-lengths, eddy radii, watershed drainage patterns, etc.**

jigsaw: meshing across scales

Build **scale-selective** triangulation as a ‘restricted filtration’ of the input geometry, sampled at **scientifically relevant length-scales**.



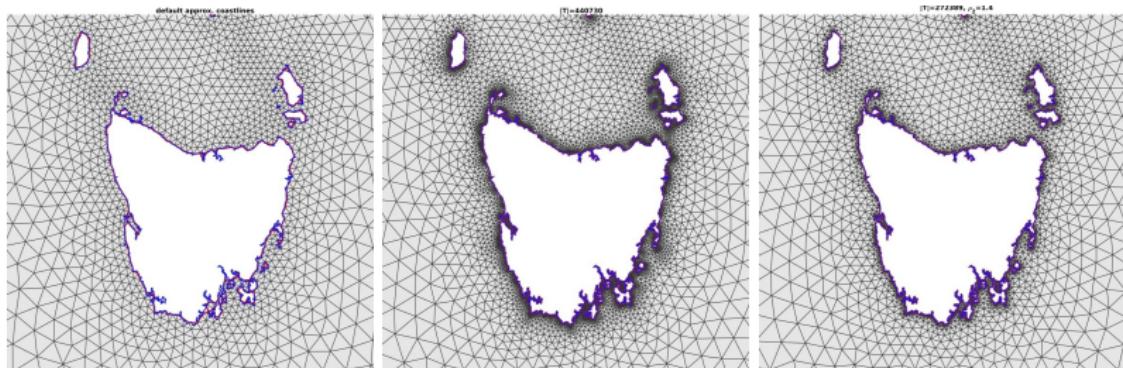
**A coastal unstructured model using Voronoi meshes and C-grid staggering, Engwirda et al, 2020.

Restricted filtration is a (re)meshing of the domain + boundary, effectively ‘**filtering-out**’ features in the input geometry **smaller than the scales of interest**.

jigsaw: local-refinement, non-default options, etc...

jigsaw tries to do a good default job out-of-the-box, but there are many different user-config. options (see eg. <https://github.com/dengwirda/jigsaw/wiki>).

Can generate, for example, **sparse locally-refined meshes at coastlines**:



Various workflows (<https://github.com/dengwirda/jigsaw-geo-tutorial>):

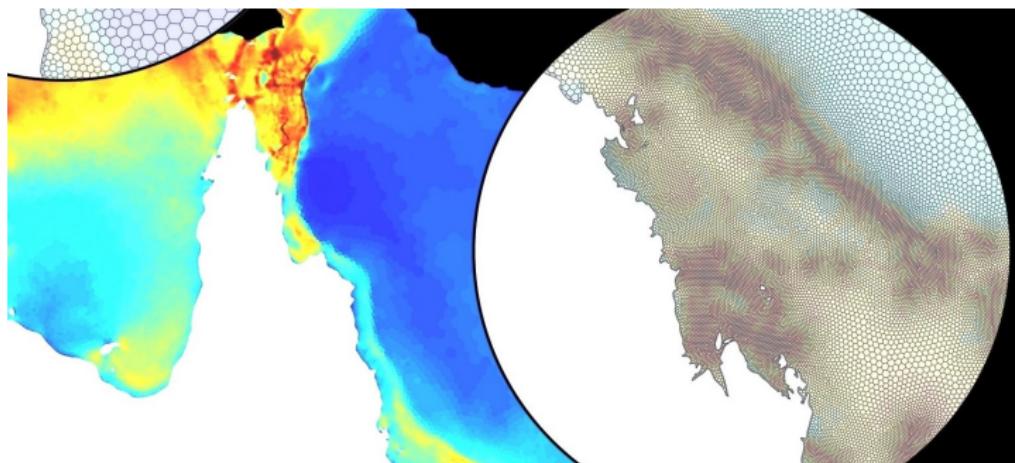
- ‘Initial condition’ points + meshes for local refinement.
- ‘Internal’ boundaries + alignment.
- ‘Multi-part’ geometries.
- Arbitrary mesh resolution patterns + preparation algorithms.

jigsaw: new feature 'to-do' list...

There are a number of new features under consideration for jigsaw:

- Improved mesh optimisation efficiency, through multi-core optimisation kernels.
- Full support for mixed triangle + quadrilateral meshes.
- Generation + optimisation of anisotropic meshes.

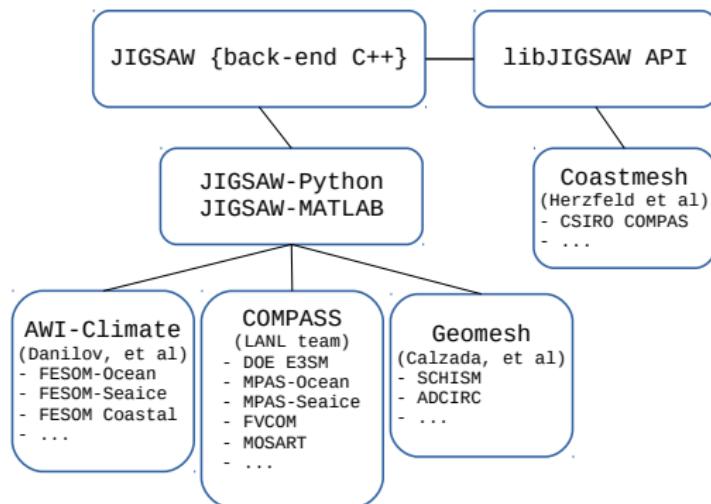
jigsaw is an open-source community meshing engine, with opportunities for collaborative development and enhancement.



jigsaw is not mesh2d!

Back in my undergraduate degree, I wrote a simple, heuristics-based meshing scheme: mesh2d.

This MATLAB library is still (surprisingly) popular, though I do not recommend it for real work. **Use jigsaw instead!**



Python and/or MATLAB scripting interface to C++ back-end meshing engine.

jigsaw: simple script-based workflows

At DOE we use Python scripting environments to enable collaborative mesh design → GitHub-based workflow to ‘spin-up’ E3SM model configurations:

The screenshot shows a GitHub pull request interface. On the left, there's a comment from user `mark-peerson` 4 days ago, which includes a plot of grid cell size versus latitude and several maps of different regions (Gulf of Mexico, Bering Sea) with their respective cell width distributions. On the right, another comment from user `zylar` 6 days ago discusses higher resolution in the North Atlantic, accompanied by a map of the North Atlantic region with a color scale for cell width. Below the comments, a code snippet in Jupyter Notebook syntax defines a mesh geometry using a square grid and lists of vertices and edges.

`grid cell size (km) versus latitude`

`narrow EC60to30`

`cellWidth`

`region Gulf of Mexico mask`

`cellWidth`

`region Bering Sea mask`

`cellWidth`

`Higher resolution in the North Atlantic`

`As requested at the last E3SM Cryosphere meeting, @maltrud came up with the following request for enhanced resolution:`

`define JIGSAW geometry`

```
38 R=-----+-----+
39 geom.meshID = "euclidean-mesh"
40 geom.ndims = +2
41 geom.vert2 = np.array([
42     # list of xy "node" coordinate
43     ((0, 0), 0),                                # outer square
44     ((1, 0), 0),
45     ((2, 0), 0),
46     ((3, 0), 0),
47     ((0, 1), 0),                                # inner square
48     ((1, 1), 0),
49     ((2, 1), 0),
50     ((3, 1), 0),
51     ((0, 2), 0),
52     ((1, 2), 0),
53     ((2, 2), 0),
54     ((3, 2), 0),
55     ((0, 3), 0),
56     ((1, 3), 0),
57     ((2, 3), 0),
58     ((3, 3), 0)),
59 dtype=jigساپy.jigsaw_msh_t.VERTEX_t)
60
61 geom.edge2 = np.array([
62     # list of "edges" between vert
63     ((0, 1), 0),                                # outer square
64     ((1, 2), 0),
65     ((2, 3), 0),
66     ((3, 0), 0),
67     ((4, 1), 0),                                # inner square
68     ((5, 2), 0),
69     ((6, 3), 0),
70     ((7, 0), 0),
71     ((8, 1), 0),
72     ((9, 2), 0),
73     ((10, 3), 0)),
74 dtype=jigساپy.jigsaw_msh_t.EDGE_t)
```