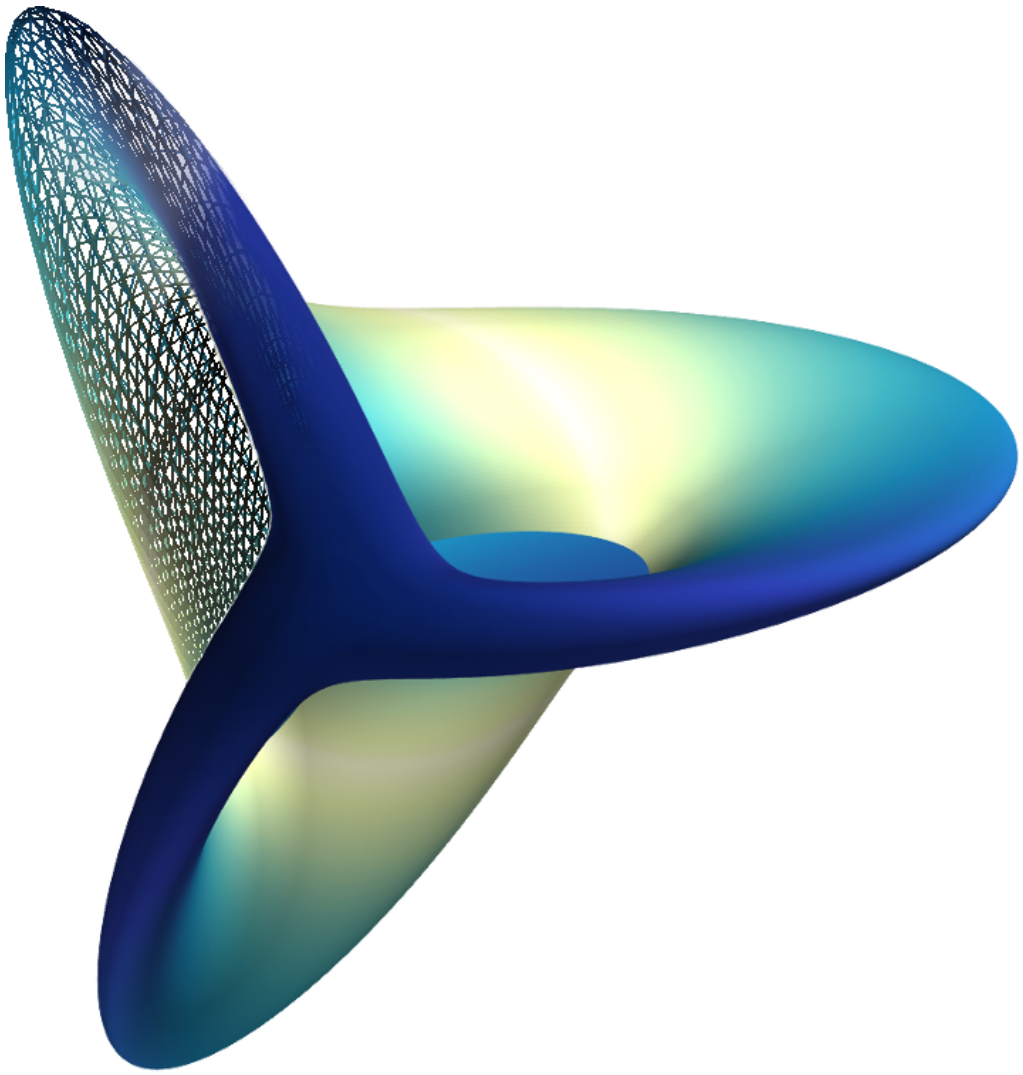
Mayavi2 tutorial

Prabhu Ramachandran and Gaël Varoquaux



Introduction to Mlab

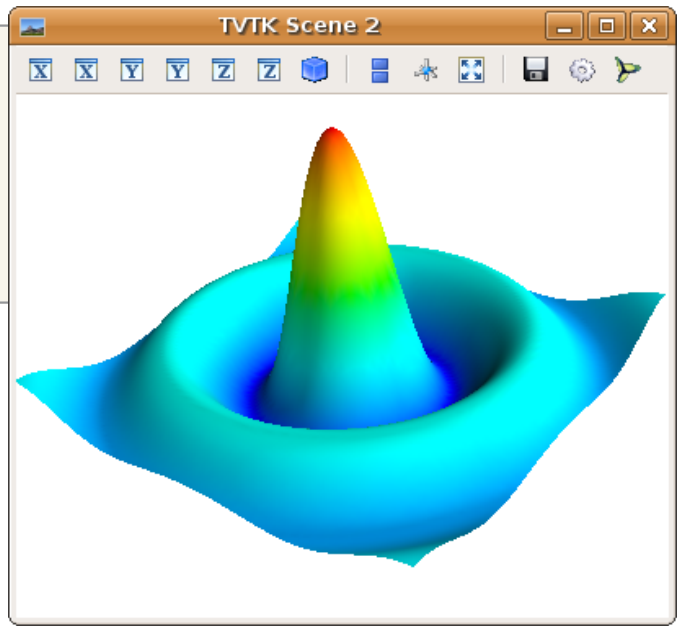
Introduction to Mlab

1 Mlab: simple scripting for Mayavi2

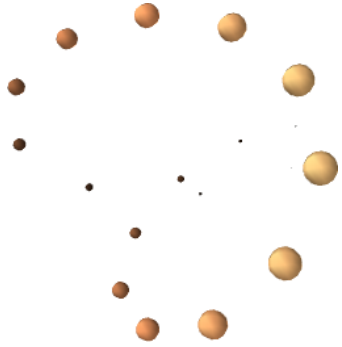
from enthought.mayavi **import** mlab

- **Simple problems should have simple solutions.**
- Work interactively in IPython (`-wthread`).
- Work with `numpy` arrays.
- People know the pylab/matlab plotting API.

```
x, y = ogrid[-10:10:100j, -10:10:100j]
r = sqrt(x**2 + y**2)
from enthought.mayavi import mlab
mlab.surf(sin(r)/r)
```

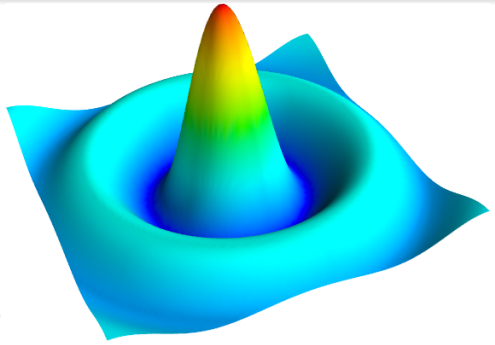


1 Mlab: plotting functions



0D data

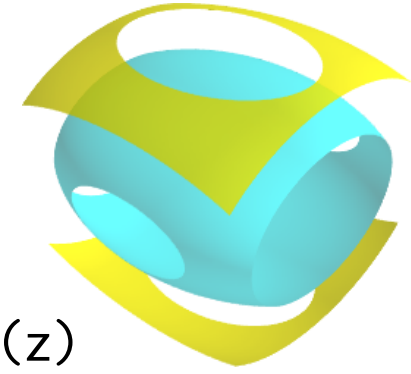
mlab.points3d(x, y, z)



2D data

mlab.surf(z)

mlab.mesh(x, y, z)

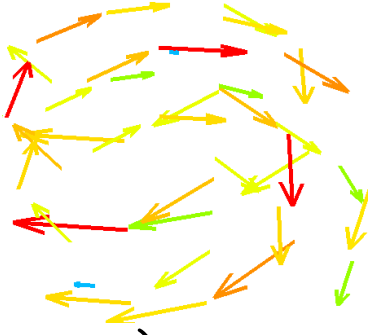


3D data

mlab.contour3d(z)

1D data

mlab.plot3d(x, y, z)

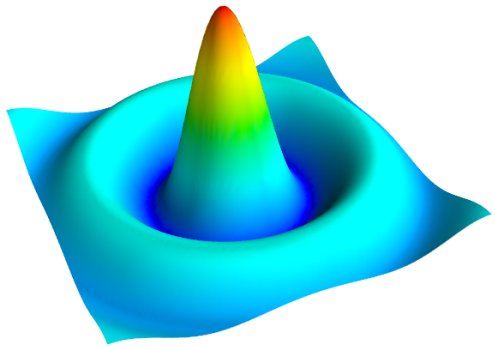


Vector fields

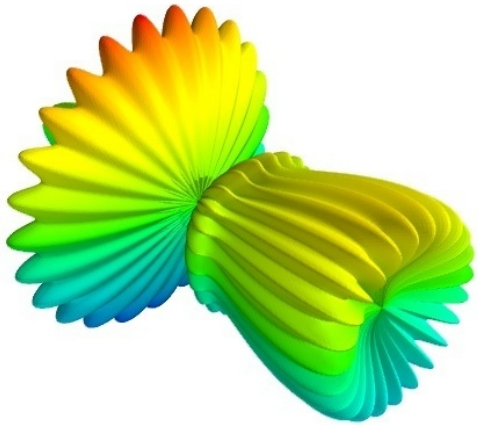
mlab.quiver3d(x, y, z)

1 Mlab: plotting functions (remarks)

- List of plotting functions in user guide.
- Every plotting function has a test/demo function.
- `mlab.surf` vs. `mlab.mesh`:
  - `mlab.surf` = carpet plot:
    image-like data + elevation



  - `mlab.mesh` = orthogonal-grid,
    but general shape

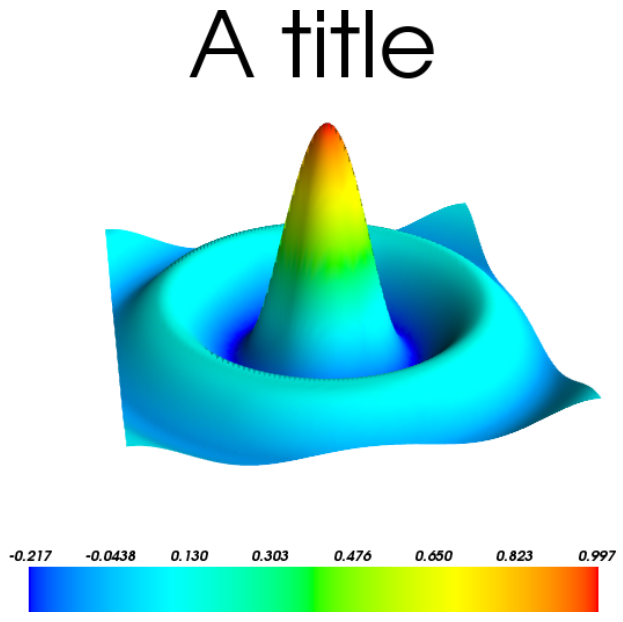


1 Mlab: sprucing up the plot

```
mlab.title('A title')
mlab.colorbar()
```

+ axes, outlines, text, ...

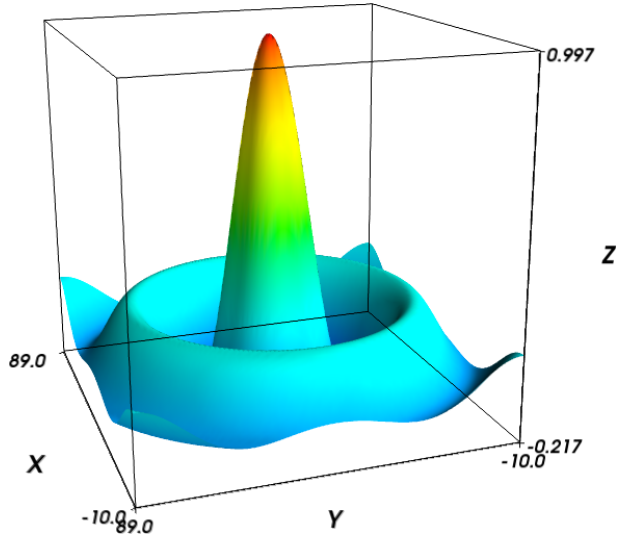
- Many keyword arguments for plotting functions.



Gotcha: extents

In VTK the extents are given by the data extents.

```
x, y = mgrid[-10:10:100j, -10:10:100j]
r = sqrt(x**2 + y**2)
s = mlab.mesh(x, y, sin(r)/r, extent
    =(0,1, 0,1, 0,1))
mlab.outline(s, extent=(0,1, 0,1, 0,1))
mlab.axes(s, extent=(0,1, 0,1, 0,1))
```



1 Mlab: managing figures

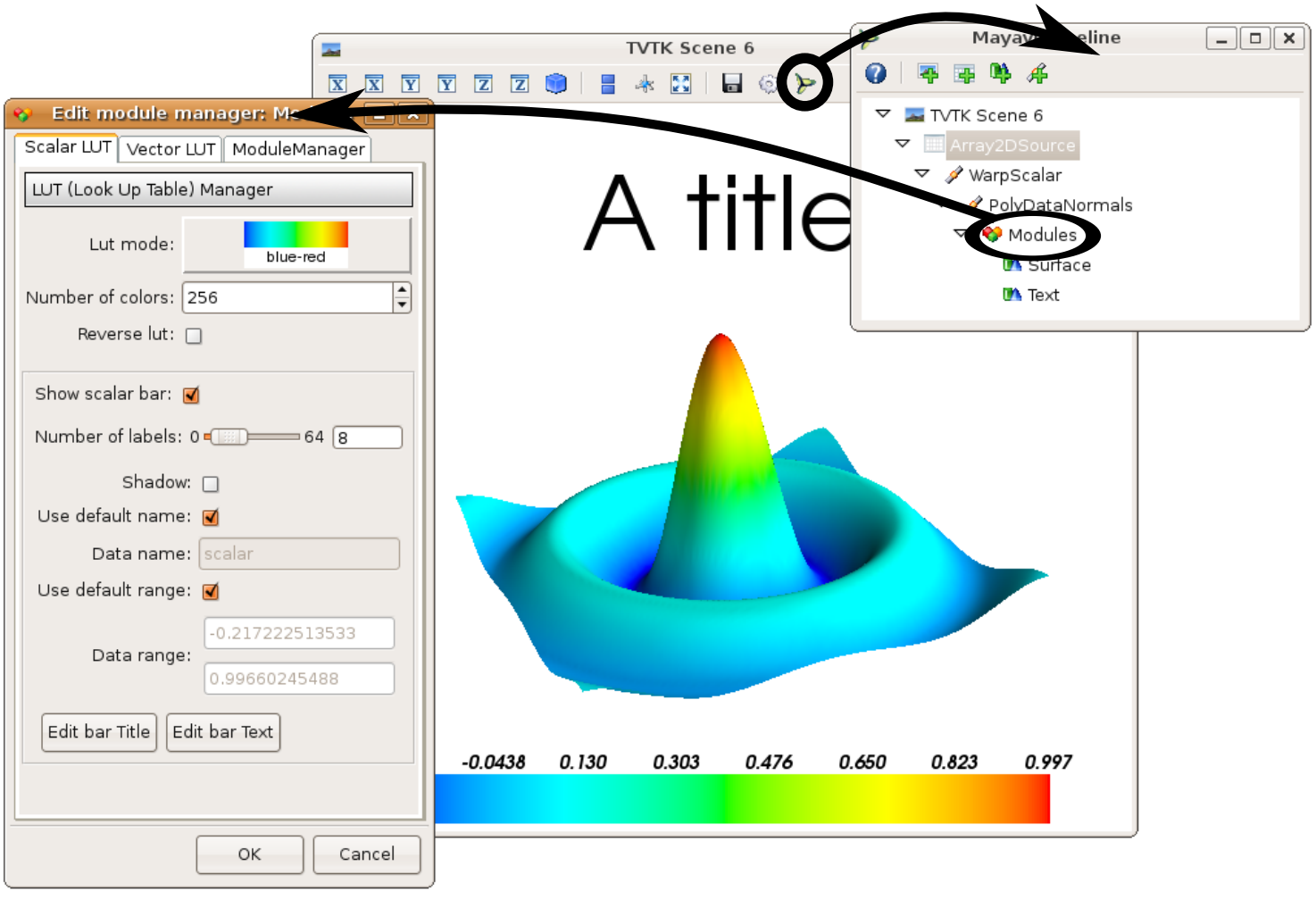
Figures

- `mlab.figure()`: create a new figure or retrieve an existing figure.
- `mlab.clf()`: clear the current figure.
- `mlab.gcf()`: return the current figure.

show and the event loop

- GUI event loop needs to be running:
  `mlab.show()` to display the visualization (after creating it).
- `@mlab.show`: decorator to make sure a function runs in the event loop.

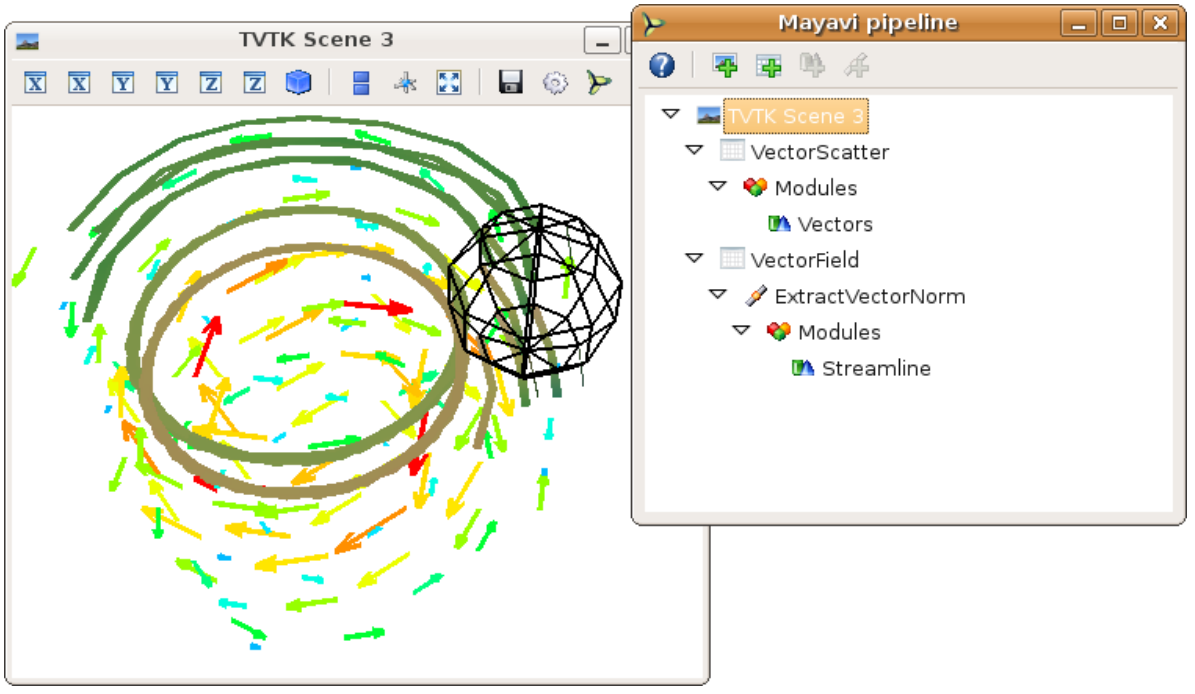
1 Mlab: Interacting graphically




```
mlab.show_pipeline()
```

1 Mlab: the pipeline

```
x, y, z = mgrid[-2:3, -2:3, -2:3]
r = sqrt(x**2 + y**2 + z**4)
mlab.quiver3d(y*sin(r)/r, -x*sin(r)/r, zeros_like(z))
mlab.flow(y*sin(r)/r, -x*sin(r)/r, zeros_like(z),
          colormap='gist_earth', linetype='ribbon')
```

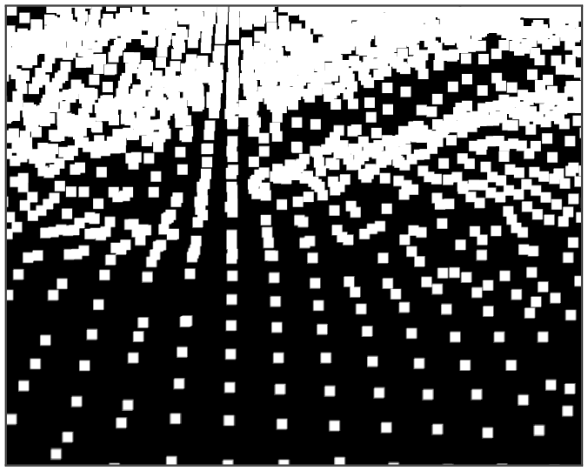


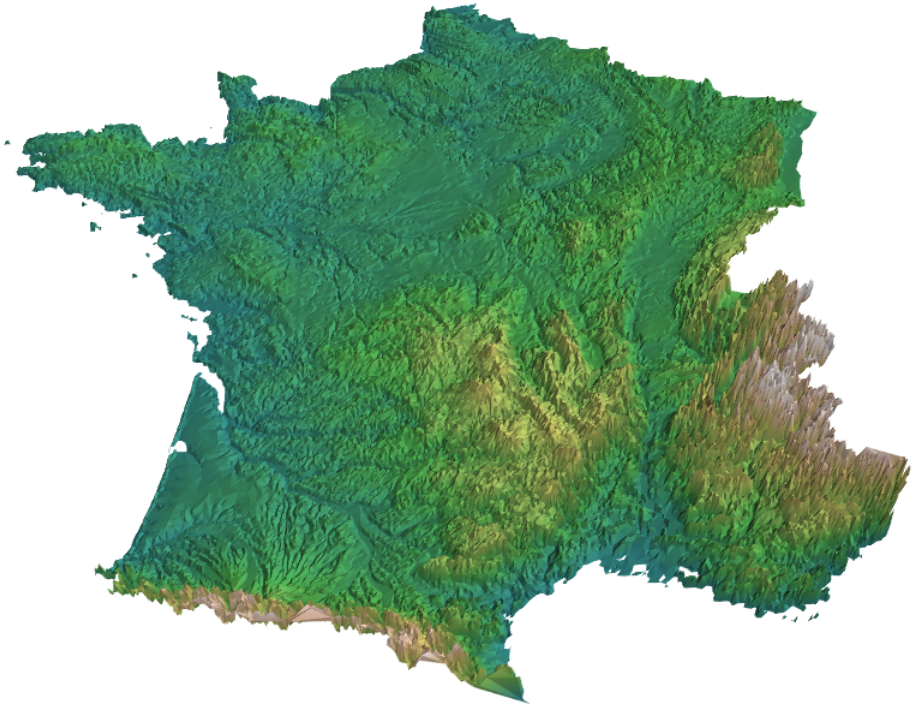
We visualize the same data with two different methods:

Visualizations = data sources + visualization modules

1 Mlab: a more complex pipeline

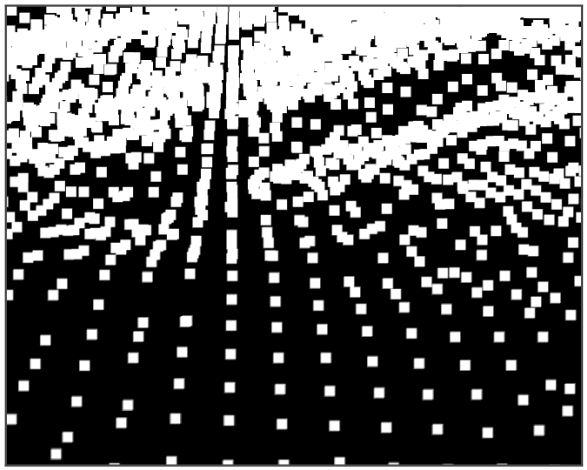
30 000 points

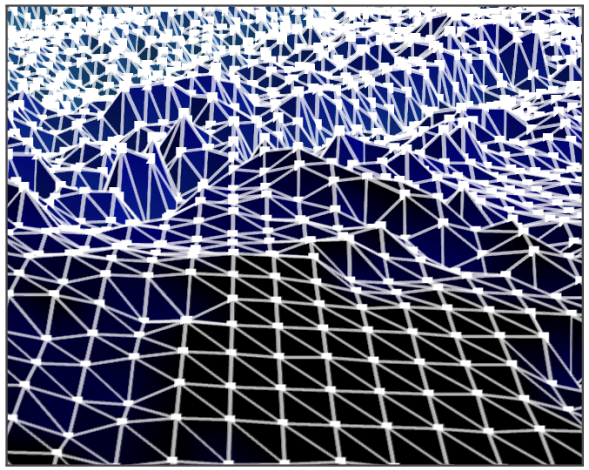


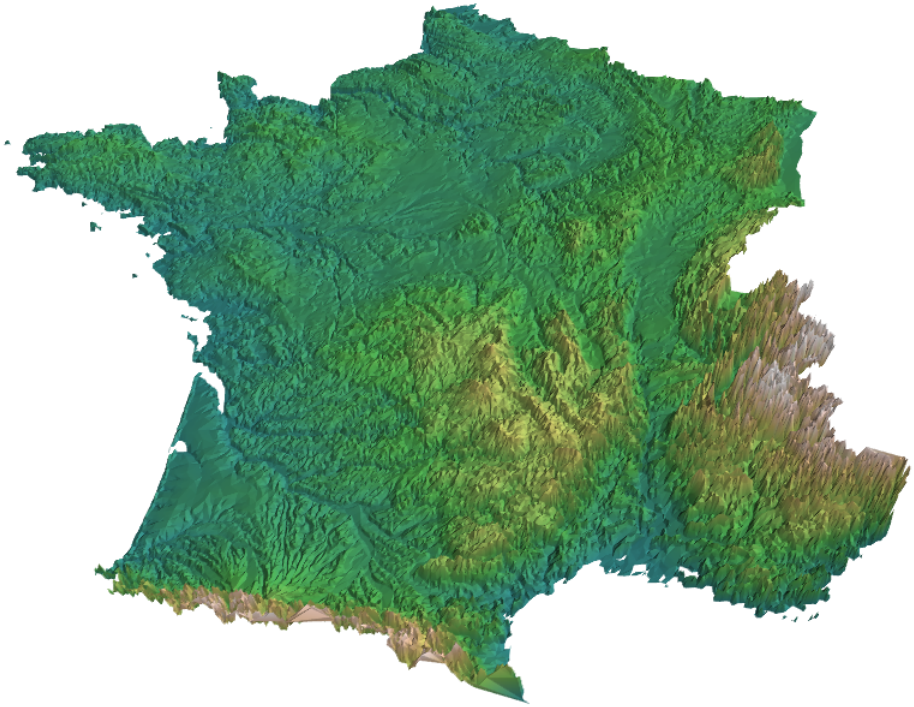


1 Mlab: a more complex pipeline

30 000 points $\xrightarrow{\text{Delaunay2D}}$ Mesh

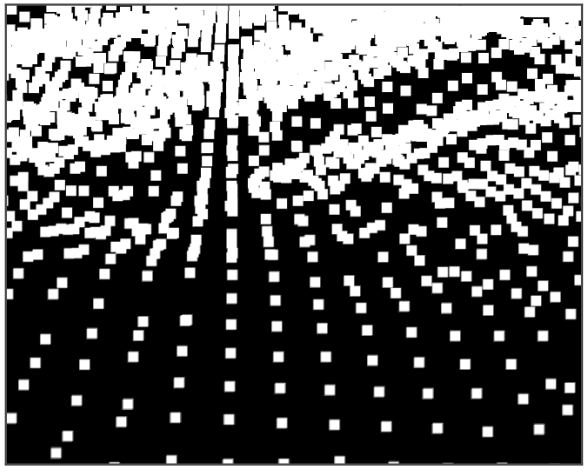


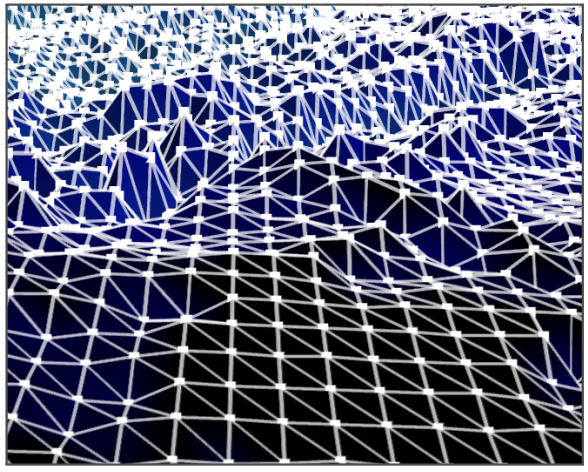


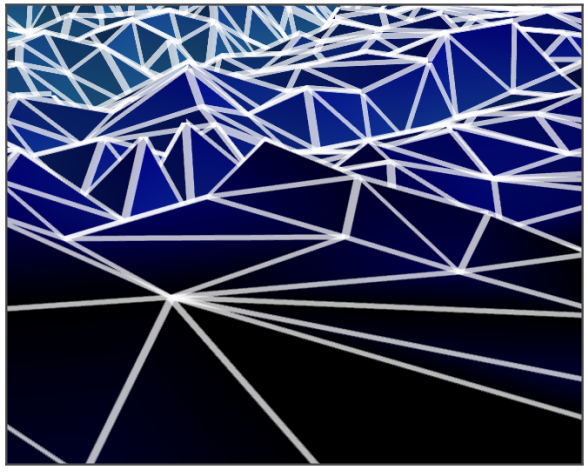


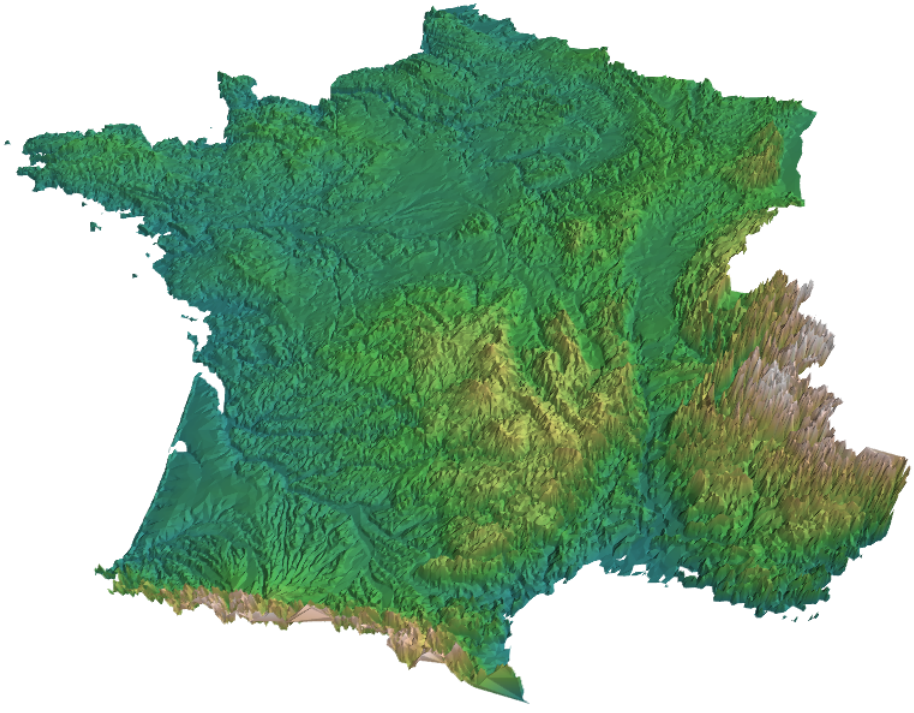
1 Mlab: a more complex pipeline

30 000 points $\xrightarrow{\text{Delaunay2D}}$ Mesh $\xrightarrow{\text{Decimation}}$ Irregular mesh



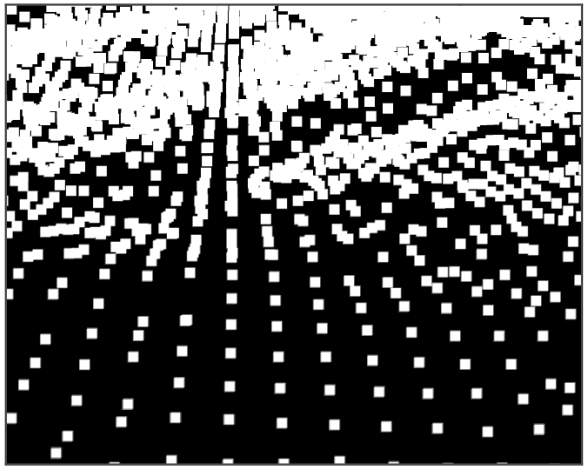
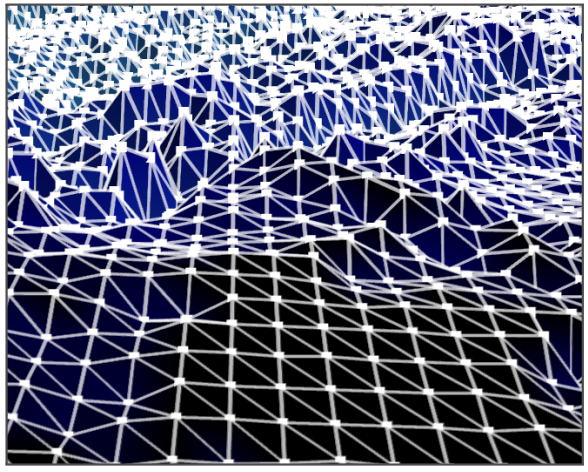
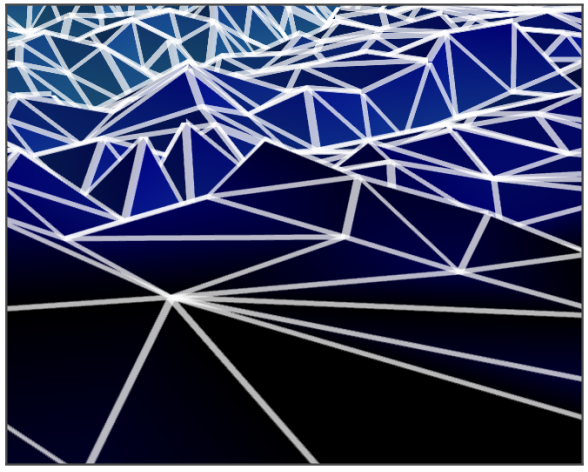




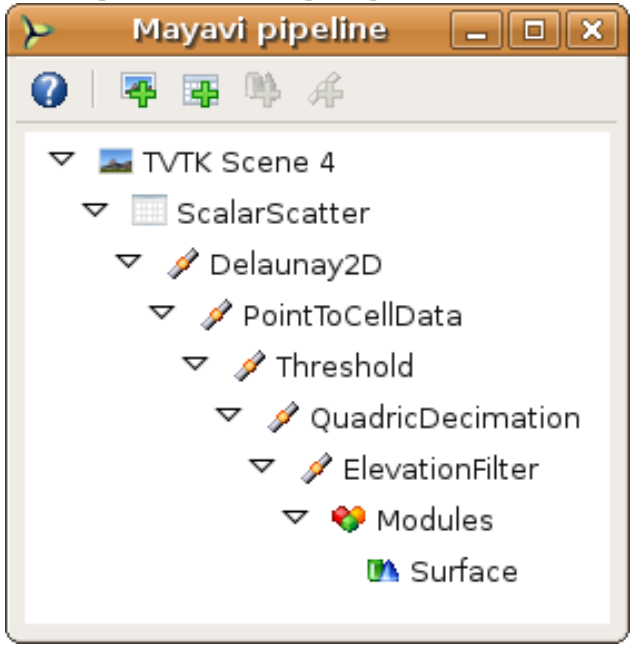


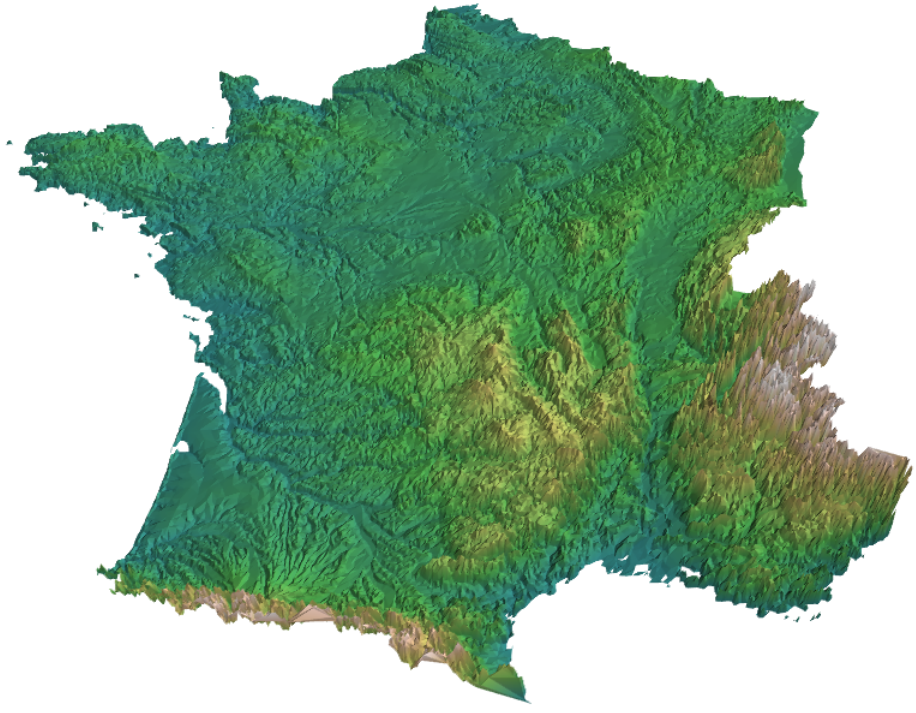
1 Mlab: a more complex pipeline

30 000 points —Delaunay2D→ Mesh —Decimation→ Irregular mesh

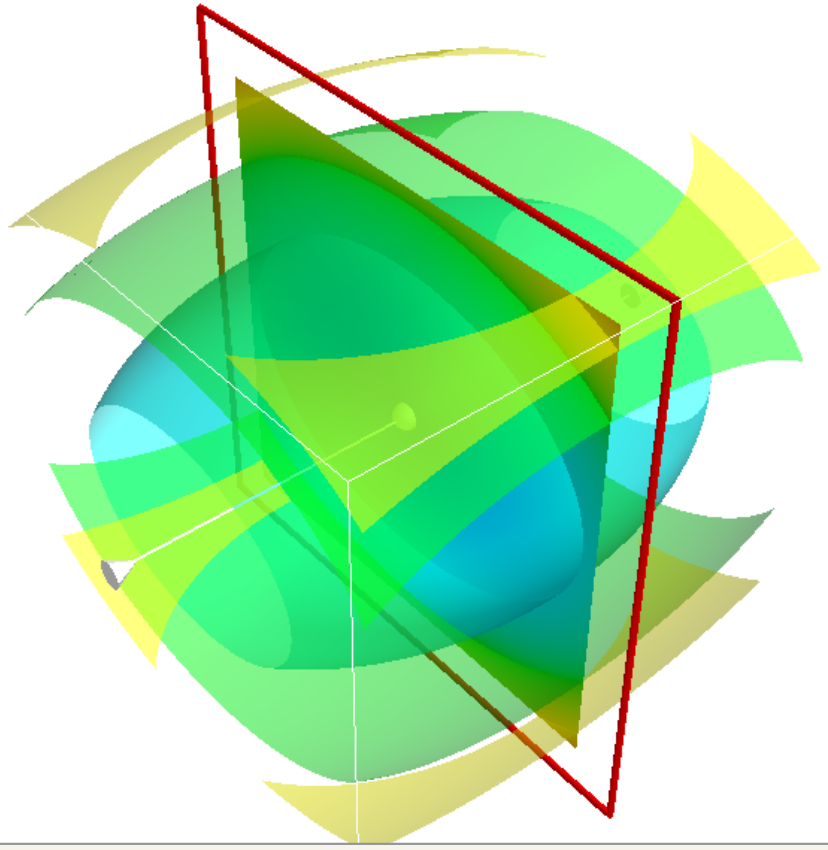




Complete pipeline

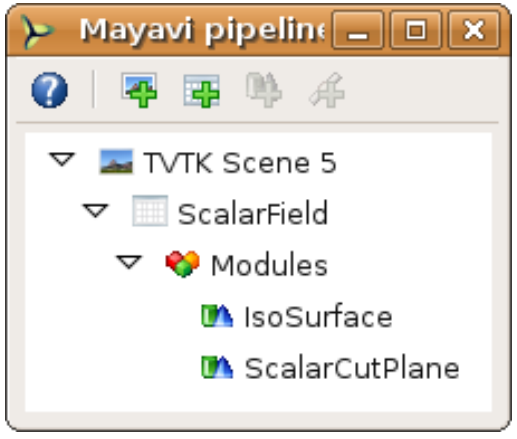




1 Mlab: creating the pipeline



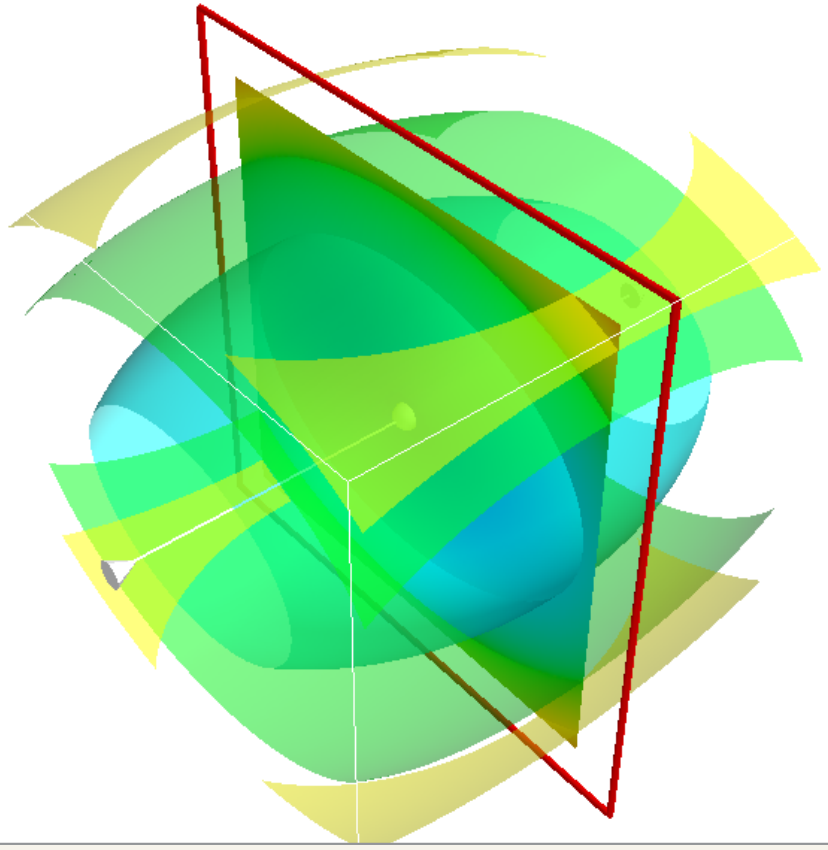
Adding a cut plane



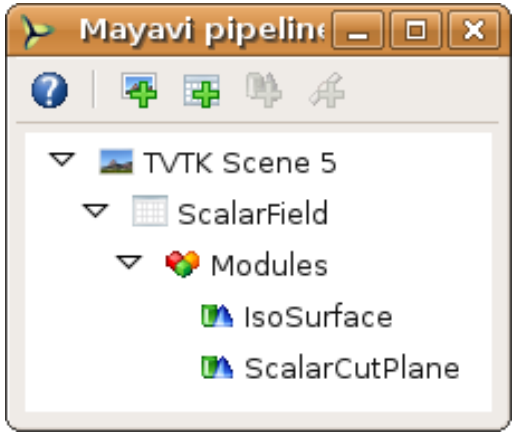

```
x, y, z = ogrid[-5:5:100j, -5:5:100j, -5:5:100j]

scalars = x*x*0.5 + y*y + z*z*2.0
obj = mlab.contour3d(scalars)
```

1 Mlab: creating the pipeline



Adding a cut plane



mlab.pipeline object

```
x, y, z = ogrid[-5:5:100j, -5:5:100j, -5:5:100j]

scalars = x*x*0.5 + y*y + z*z*2.0
obj = mlab.contour3d(scalars, opacity=0.5)
mlab.pipeline.scalar_cut_plane(obj)
```

1 Mlab and Mayavi2

- `mlab.options.backend = 'envisage'`
  ⇒ mlab commands open up a full blown mayavi application.
- By default, mlab uses a full blown application if open.
  ⇒ Use in interactive shell and `mayavi2 -x foo.py`