TRƯỜNG ĐẠI HỌC MỎ-ĐỊA CHẤT

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO HỌC THUẬT

XỬ LÝ DỮ LIỆU KHÔNG GIAN VỚI THƯ VIỆN NGUỒN MỞ GEMGIS

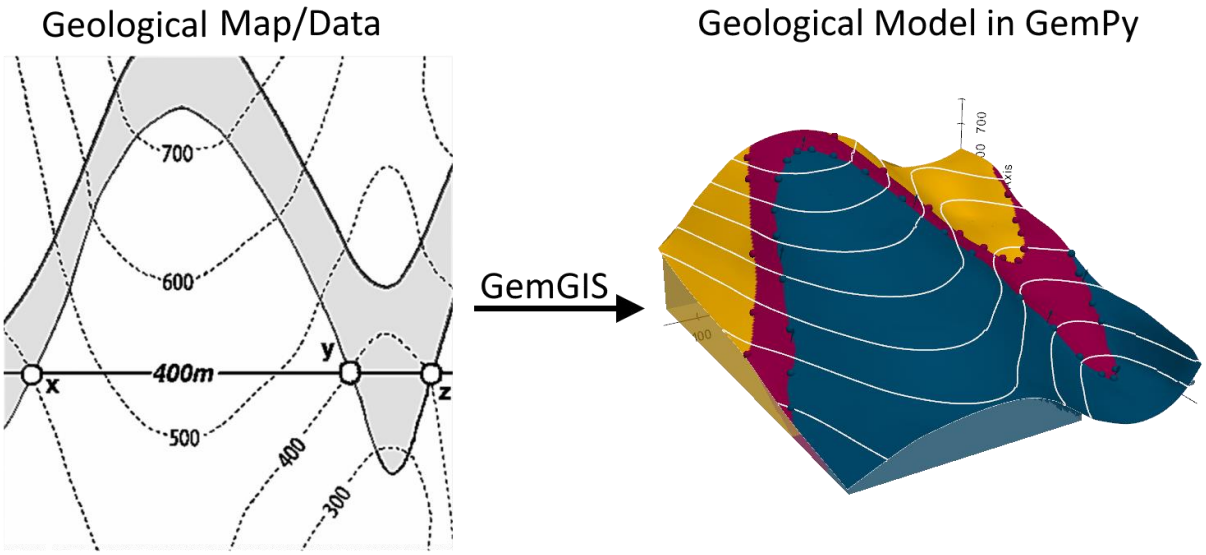
PHẠM AN CƯƠNG

HÀ NỘI, 12/2023

MỤC LỤC

MỞ ĐẦU

GemGIS là một thư viện xử lý thông tin địa lý nguồn mở, dựa trên Python. Nó có khả năng xử lý trước dữ liệu không gian như dữ liệu vectơ (tệp hình dạng, tệp Geojson, gói địa lý,…), dữ liệu raster (tif, png,…), dữ liệu thu được từ các dịch vụ trực tuyến (WCS, WMS, WFS) hoặc tệp XML/KML (sớm). Dữ liệu được xử lý trước có thể được lưu trữ trong Lớp dữ liệu chuyên dụng để chuyển đến gói tạo mô hình địa lý GemPy nhằm đẩy nhanh quá trình xây dựng mô hình. Quá trình xử lý hậu kỳ các kết quả mô hình sẽ cho phép xuất từ GemPy sang các hệ thống thông tin địa lý như QGIS và ArcGIS hoặc sang Google Earth để sử dụng tiếp.
GemGIS sử dụng và kết hợp đầy đủ chức năng của GeoPandas, rasterio, OWSLib, Pandas, Shapely, PyVista và NumPy để đơn giản hóa, tăng tốc và tự động hóa các quy trình công việc được sử dụng để xử lý trước dữ liệu không gian cho mô hình hóa địa lý.



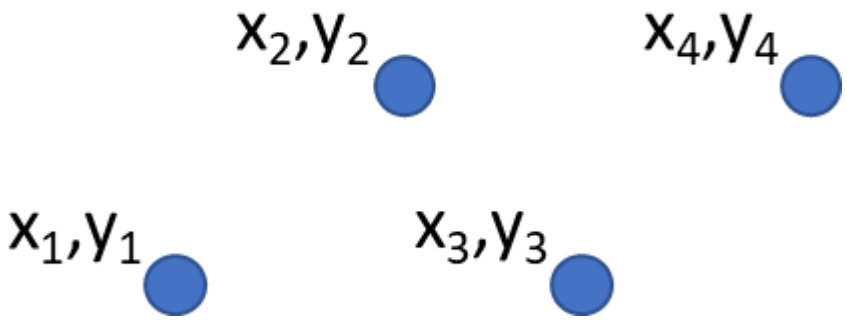
Phần Tham khảo API cung cấp thông tin về các chức năng khác nhau được triển khai trong GemGIS. Điều này bao gồm Đối tượng dữ liệu GemGIS, Làm việc với dữ liệu vectơ, Làm việc với dữ liệu raster, Làm việc với các dịch vụ trực tuyến, các Công cụ tiện ích bổ sung khác, Trực quan hóa và Vẽ đồ thị, các phương pháp khác không được sử dụng thường xuyên hoặc cho các trường hợp cụ thể trong phần Linh tinh và cuối cùng nhưng không kém phần quan trọng là Xử lý hậu kỳ của các mô hình GemPy.
Mỗi bộ chức năng được tập hợp trong một mô-đun khác nhau. Các chức năng của từng mô-đun có thể được truy cập như sau:

```
import gemgis as gg

data = gg.vector.function_name(...)

data = gg.raster.function_name(...)

data = gg.visualization.function_name(...)

data = gg.web.function_name(...)

data = gg.utils.function_name(...)

data = gg.misc.functions_name(...)
```

1. TỔNG QUAN VỀ CÁC DỮ LIỆU TRONG GEMGIS

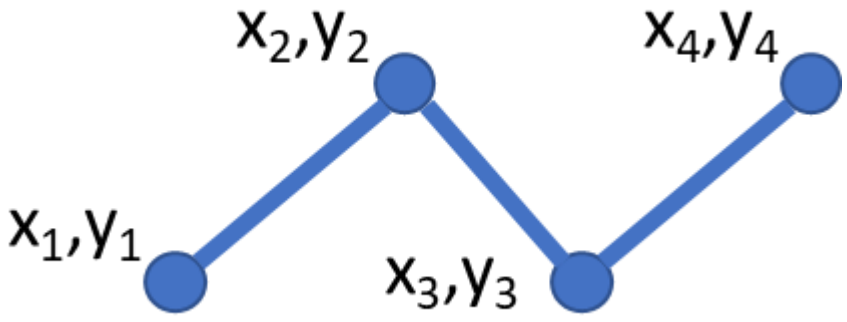
1.1 Dữ liệu điểm

Một đối tượng điểm có giá trị X, Y và tùy chọn Z. Giá trị X và Y sẽ phụ thuộc vào Hệ thống Tham chiếu Tọa độ (CRS) đang được sử dụng.CRS là một cách để mô tả chính xác vị trí của một địa điểm cụ thể trên bề mặt trái đất. Một trong những hệ quy chiếu phổ biến nhất là Kinh độ và Vĩ độ. Các đường kinh độ chạy từ Cực Bắc đến Cực Nam. Các đường vĩ độ chạy từ Đông sang Tây. Bạn có thể mô tả chính xác vị trí của mình ở bất kỳ nơi nào trên trái đất bằng cách cung cấp cho ai đó Kinh độ (X) và Vĩ độ (Y). Vì chúng ta biết trái đất không bằng phẳng nên việc thêm giá trị Z vào đối tượng địa lý điểm thường rất hữu ích. Điều này mô tả độ cao của bạn so với mực nước biển.



1.2 Dữ liệu đường

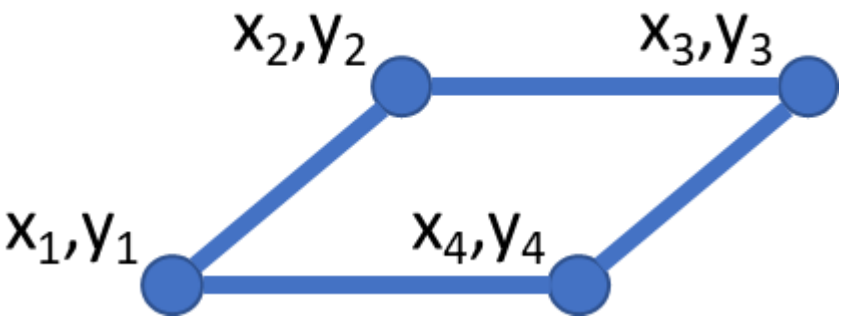
Một đường hoặc nhiều đường được sử dụng để hiển thị hình dạng của các đối tượng tuyến tính như đường, sông, đường đồng mức, lối đi bộ, đường bay, v.v. Đôi khi, chúng tôi có các quy tắc đặc biệt cho đường polyline ngoài hình học cơ bản của chúng. Ví dụ: các đường đồng mức có thể chạm nhau (ví dụ: ở mặt vách đá) nhưng không bao giờ được cắt nhau. Tương tự, các đường polylines được sử dụng để lưu trữ mạng lưới đường bộ phải được kết nối tại các giao lộ. Trong một số ứng dụng GIS, bạn có thể đặt các quy tắc đặc biệt này cho một loại đối tượng địa lý (ví dụ: đường) và GIS sẽ đảm bảo rằng các đường đa tuyến này luôn tuân thủ các quy tắc này.



1.3 Dữ liệu đa giác

Đối tượng địa lý đa giác là các khu vực khép kín như đập, đảo, ranh giới quốc gia, v.v. Giống như các đối tượng địa lý đa tuyến, đa giác được tạo từ một loạt các đỉnh được kết nối bằng một đường liên tục. Tuy nhiên, vì một đa giác luôn mô tả một khu vực kín nên đỉnh đầu tiên và đỉnh cuối cùng phải luôn ở cùng một vị trí! Đa giác thường có ranh giới hình học chung với đa giác

lân cận. Nhiều ứng dụng GIS có khả năng đảm bảo rằng ranh giới của các đa giác lân cận hoàn toàn trùng khớp.



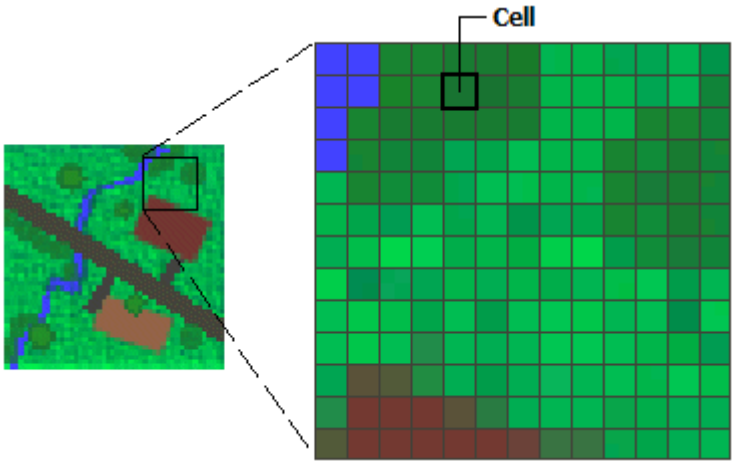
1.4 Làm việc với dữ liệu vectơ trong GemGIS

Dữ liệu vectơ là một trong những loại dữ liệu cơ bản được sử dụng trong GemGIS. Dữ liệu vectơ được xử lý bởi gói Shapely cho các hình học đơn lẻ và bởi gói PyGEOS cho các mảng hình học và các phép toán vector hóa được thực hiện trên các mảng này.
Tọa độ các đỉnh của mỗi phần tử không được liên kết với hệ quy chiếu tọa độ (CRS) tại thời điểm này. Điều này được thêm vào khi thu thập nhiều đối tượng trong GeoPandas GeoDataFrame. Đối tượng giống như Pandas DataFrame này bao gồm một cột hình học chứa các đối tượng hình học và cột dữ liệu khác nhau bằng các trường thuộc tính của tệp hình dạng. CRS thường được xác định trước khi tải các tệp hình dạng bao gồm các đối tượng hình học bằng GeoPandas hoặc có thể được chỉ định khi tạo GeoDataFrame mới. Sau đó, có thể thực hiện các phép biến đổi tọa độ trên GeoDataFrames chứa thuộc tính CRS. Các bộ dữ liệu khác nhau có cùng hệ quy chiếu tọa độ có thể được vẽ để trực quan hóa sự phân bố không gian của dữ liệu.

	formation	geometry	X	Y
0	Sand1	POINT (0.256 264.862)	0.26	264.86
1	Sand1	POINT (10.593 276.734)	10.59	276.73
2	Sand1	POINT (17.135 289.090)	17.13	289.09
3	Sand1	POINT (19.150 293.313)	19.15	293.31
4	Sand1	POINT (27.795 310.572)	27.80	310.57

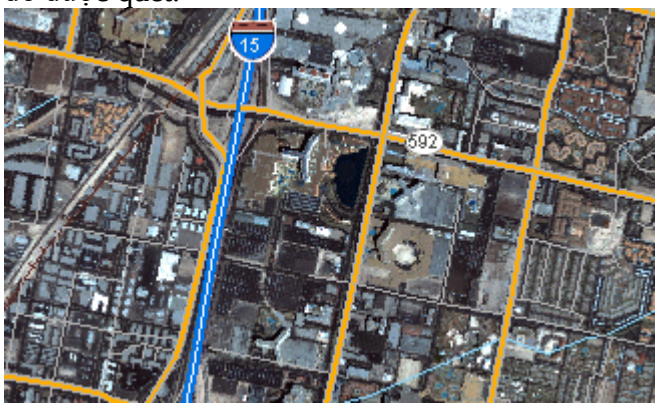
1.5 Dữ liệu raster

Ở dạng đơn giản nhất, raster bao gồm một ma trận các ô (hoặc pixel) được sắp xếp thành hàng và cột (hoặc lưới) trong đó mỗi ô chứa một giá trị biểu thị thông tin, chẳng hạn như giá trị độ cao. Raster là những bức ảnh kỹ thuật số chụp từ trên không, hình ảnh từ vệ tinh, hình ảnh kỹ thuật số hoặc thậm chí là bản đồ được quét.



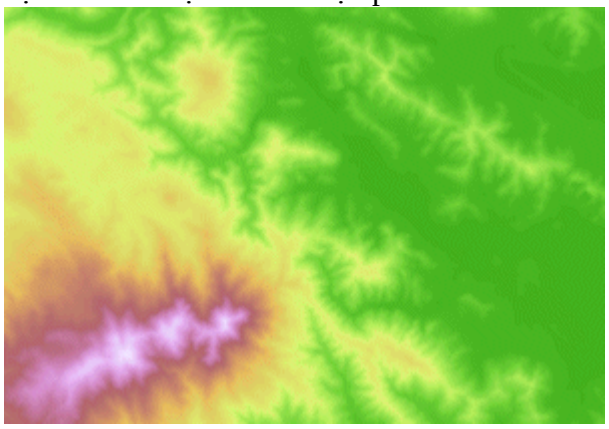
1.6 Raster làm bản đồ cơ sở

Việc sử dụng phổ biến dữ liệu raster trong GIS là hiển thị nền cho các lớp đối tượng khác. Ba nguồn chính của bản đồ nền raster là ảnh trực giao từ ảnh chụp trên không, ảnh vệ tinh và bản đồ được quét.



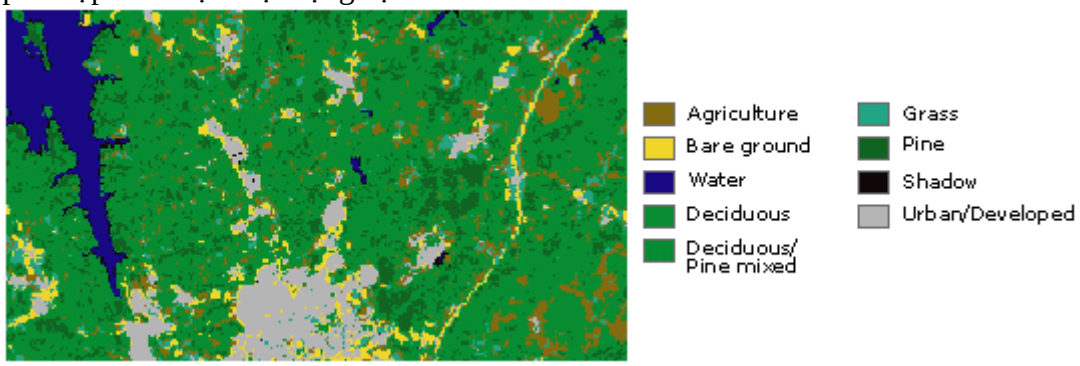
1.7 Raster dưới dạng bản đồ bề mặt

Raster rất phù hợp để biểu diễn dữ liệu thay đổi liên tục trên một cảnh quan (bề mặt). Các giá trị độ cao được đo từ bề mặt trái đất là ứng dụng phổ biến nhất của bản đồ bề mặt, nhưng các giá trị khác, chẳng hạn như lượng mưa, nhiệt độ, nồng độ và mật độ dân số, cũng có thể xác định các bề mặt có thể được phân tích theo không gian.



1.8 Raster dưới dạng bản đồ chuyên đề

Các trình quét đại diện cho dữ liệu chuyên đề có thể được lấy từ việc phân tích các dữ liệu khác. Một ứng dụng phân tích phổ biến là phân loại ảnh vệ tinh theo các loại lớp phủ mặt đất. Ví dụ: bạn có thể xử lý dữ liệu thông qua mô hình xử lý địa lý để tạo tập dữ liệu raster ánh xạ mức độ phù hợp cho một hoạt động cụ thể.

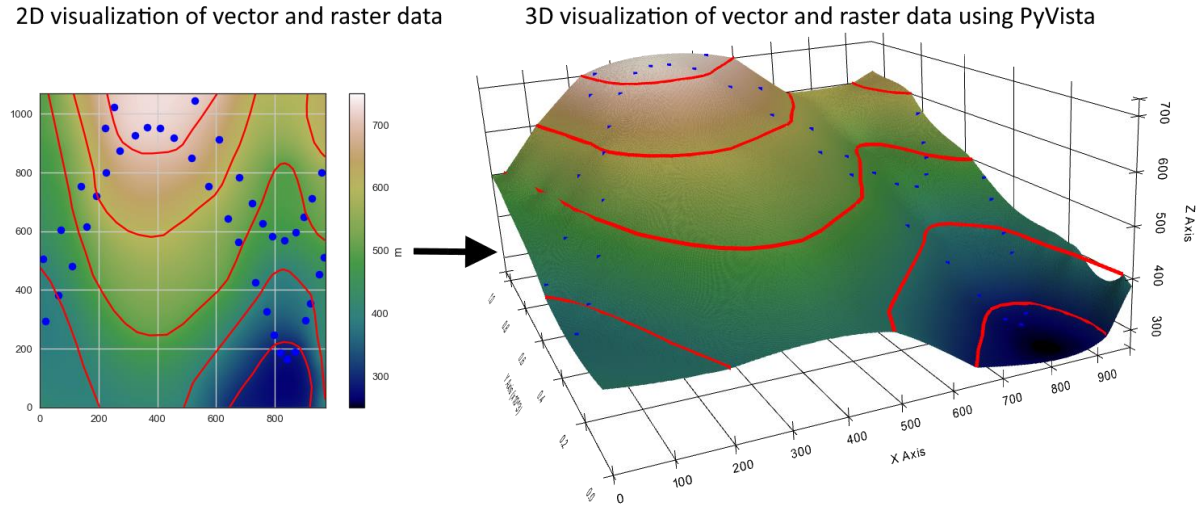


Làm việc với dữ liệu raster trong GemGIS

Dữ liệu raster là một trong những loại dữ liệu cơ bản được sử dụng trong GemGIS. Dữ liệu raster được xử lý bởi gói Rasterio và chính dữ liệu đó được NumPy xử lý.

2. TRỰC QUAN HÓA DỮ LIỆU KHÔNG GIAN VỚI PYVISTA

Dữ liệu không gian có thể được hiển thị bằng gói PyVista. Điều này bao gồm dữ liệu điểm, dữ liệu đường và raster. Dữ liệu thường sẽ được trả về dưới dạng bộ dữ liệu PolyData hoặc Lưới để người dùng có toàn quyền linh hoạt trong việc vẽ đồ thị dữ liệu bằng PyVista.



2.1 Trực quan hóa các đường đồng mức bằng PyVista

Tải dữ liệu

Các đường đồng mức được tải dưới dạng Shapely LineStrings trong GeoDataFrame.

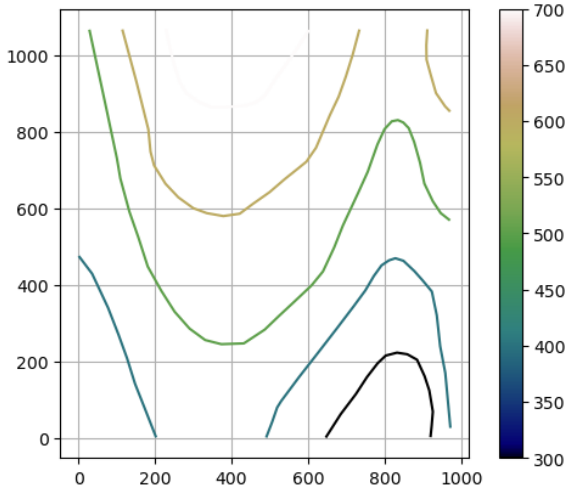
```
import pyvista as pv
import geopandas as gpd
contours = gpd.read_file(file_path + 'topo.shp')
contours.head()
```

	id	Z	geometry
0	None	400	LINESTRING (0.74088 475.44101, 35.62873 429.24...

1	None	300	LINESTRING (645.96500 0.52496, 685.14093 61.86...
2	None	400	LINESTRING (490.29223 0.52496, 505.75641 40.73...
3	None	600	LINESTRING (911.43347 1068.58451, 908.85610 10...
4	None	700	LINESTRING (228.43207 1068.58451, 239.77247 10...

Vẽ dữ liệu

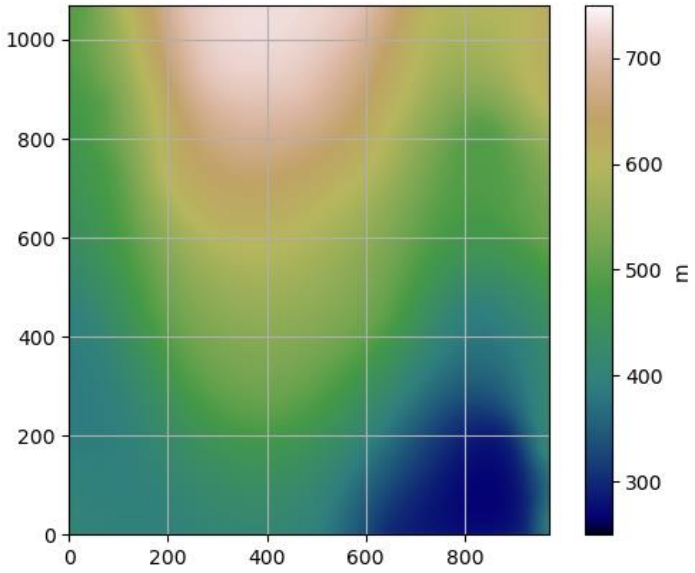
```
import matplotlib.pyplot as plt
contours.plot(aspect='equal', column='Z', cmap='gist_earth',
legend=True)
plt.grid()
```



2.2 Trực quan hóa DEM bằng PyVista

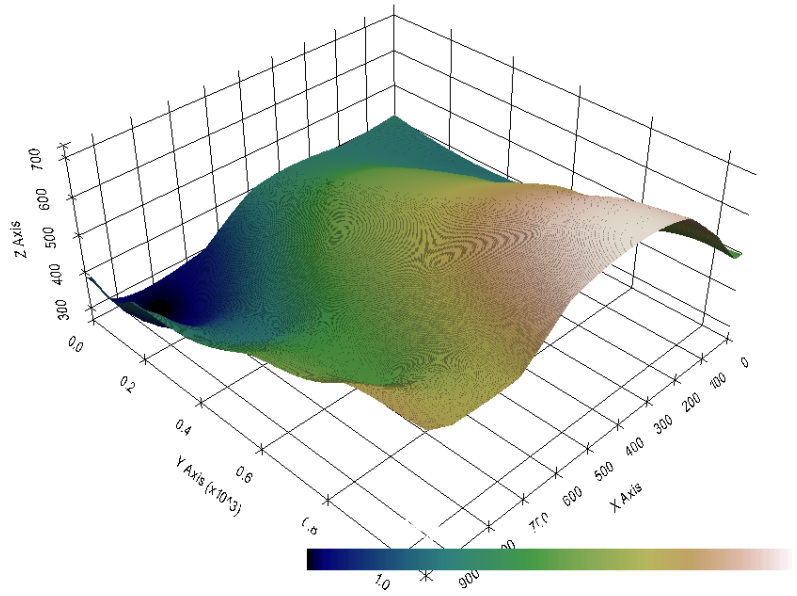
DEM được tải dưới dạng đối tượng Rasterio bằng rasterio

```
import rasterio
import gemgis as gg
import pyvista as pv
import numpy as np
dem = rasterio.open(file_path + 'topo.tif')
dem.read(1)
array([[499.90110024, 499.86421238, 499.82858152, ..., 625.37307284,
        625.78164892, 626.18920124],
       [499.53566482, 499.49887905, 499.4633306 , ..., 625.17315916,
        625.58165735, 625.98912699],
       [499.18752484, 499.15158818, 499.11692808, ..., 624.97739453,
        625.38574125, 625.79305697],
       ...,
       [411.5023835 , 411.37335931, 411.24503355, ..., 384.8252337 ,
        386.21293421, 387.56684012],
       [411.66101945, 411.5316941 , 411.40306465, ..., 384.4299191 ,
        385.80964238, 387.15718098],
       [411.82014581, 411.69052091, 411.56158939, ..., 384.04962954,
        385.42140506, 386.76248969]])
import matplotlib.pyplot as plt
im =plt.imshow(dem.read(1), cmap='gist_earth', vmin=250, vmax=750,
extent=[0,972,0,1069])
cbar = plt.colorbar(im)
cbar.set_label('m')
plt.grid()
```



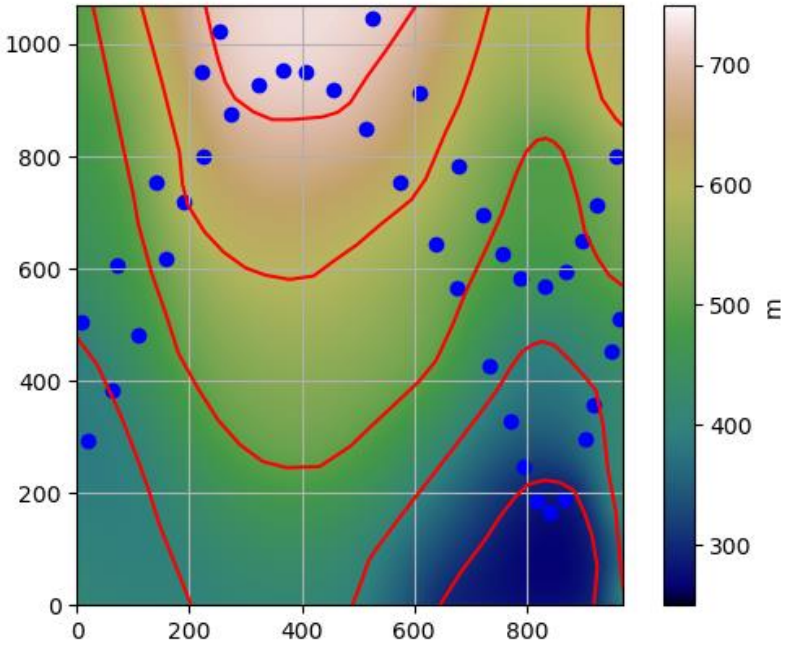
2.3 Vẽ lưới có cấu trúc bằng PyVista

```
p = pv.Plotter()
p.add_mesh(mesh=grid, scalars=grid["Elevation"], cmap='gist_earth')
p.show_grid(color='black')
p.set_background(color='white')
p.show()
```

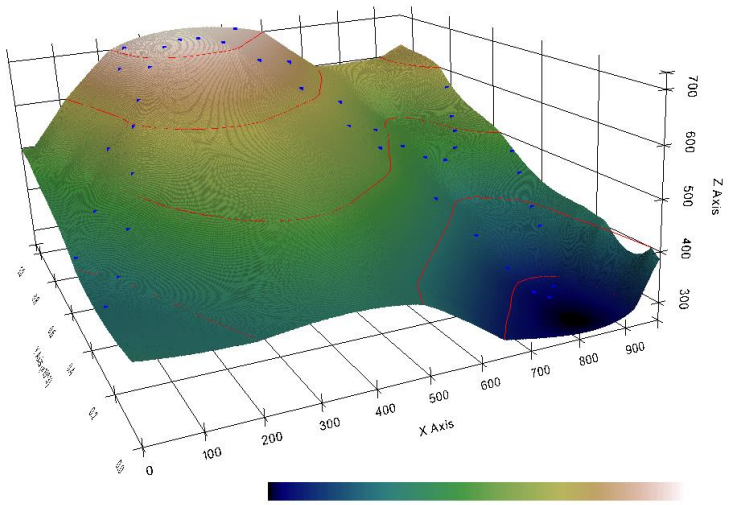



```
import matplotlib.pyplot as plt
fix, ax = plt.subplots(1,1)
points_xyz.plot(ax=ax, aspect='equal', color='blue')
contours.plot(ax=ax,aspect='equal', color='red')
im =plt.imshow(dem.read(1), cmap='gist_earth', vmin=250, vmax=750,
extent=[0,972,0,1069])
cbar = plt.colorbar(im)
cbar.set_label('m')
```

```
plt.grid()
```



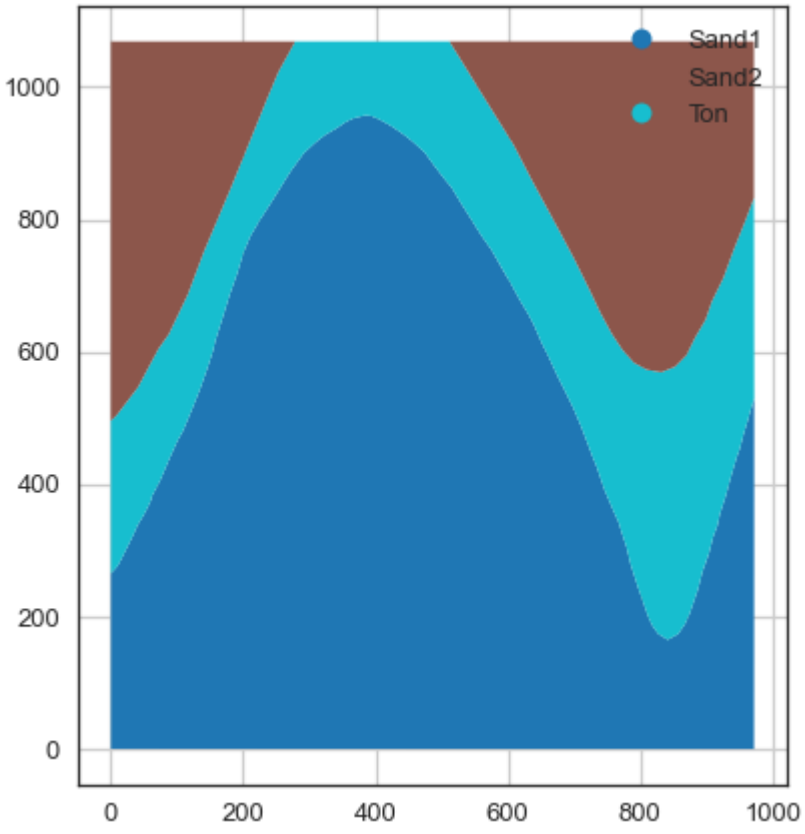

```
p = pv.Plotter()
p.add_mesh(mesh=lines, color='red')
p.add_mesh(mesh=points_mesh, color='blue')

p.add_mesh(mesh=grid, scalars=grid["Elevation"], cmap='gist_earth')
p.camera_position =[(-283.285811675846, -1597.1397046051004,
1155.542325449192),
                    (577.9371599370799, 495.3480261506809,
381.7124055285182),
                    (0.17313457304419916, 0.27814381639313923,
0.9448070898437746)]
p.set_background('white')
p.show_grid(color='black')
p.show()
```



3. PHÂN TÍCH CÁC TỆP KIỂU QGIS SANG GEMGIS

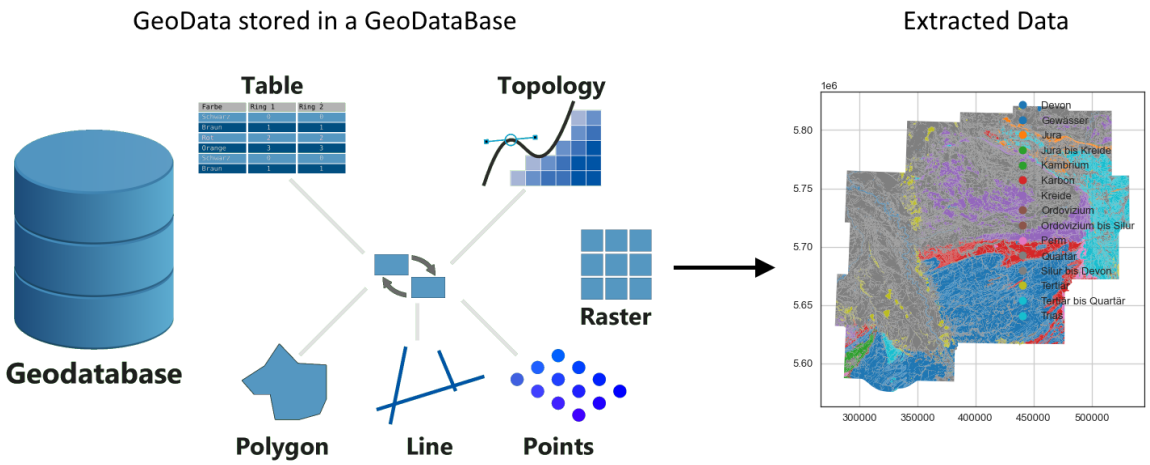
Các tệp kiểu QGIS (tệp QML) là một định dạng XML để lưu trữ kiểu dáng lớp. Tệp QML chứa tất cả thông tin mà QGIS có thể xử lý để hiển thị hình học đối tượng bao gồm định nghĩa ký hiệu, kích thước và góc quay, ghi nhãn, độ mờ và chế độ hòa trộn, v.v.

```
import gemgis as gg
file_path ='data/28_parsing_QGIS_style_files/'
gg.download_gemgis_data.download_tutorial_data(filename="28_parsing_QGIS_style_files.zip", dirpath=file_path)
import gemgis as gg
import geopandas as gpd
polygons = gpd.read_file(file_path + 'interfaces_polygons.shp')
polygons
import matplotlib.pyplot as plt
polygons.plot(column='formation', aspect='equal', legend=True)
plt.grid()
```



3.1 Mở GeoDataBases cho GemGIS

Thư mục GeoDataBase chứa dữ liệu raster hoặc vector hoặc loại định dạng khác có thể được mở bằng fiona và GeoPandas. Các lớp khác nhau có thể được liệt kê bằng fiona và được mở riêng bằng GeoPandas để trả về GeoDataFrames để xử lý tiếp.




```
import gemgis as gg
file_path ='data/30_opening_geodatabases_for_gemgis/'
gg.download_gemgis_data.download_tutorial_data(filename="30_opening_geo
databases_for_gemgis.zip", dirpath=file_path)
```

GeoDataBase đã sử dụng đã được tải xuống từ https://www.opengeodata.nrw.de/produkte/geologie/geologie/GK/ISGK100/ISGK100vektor/

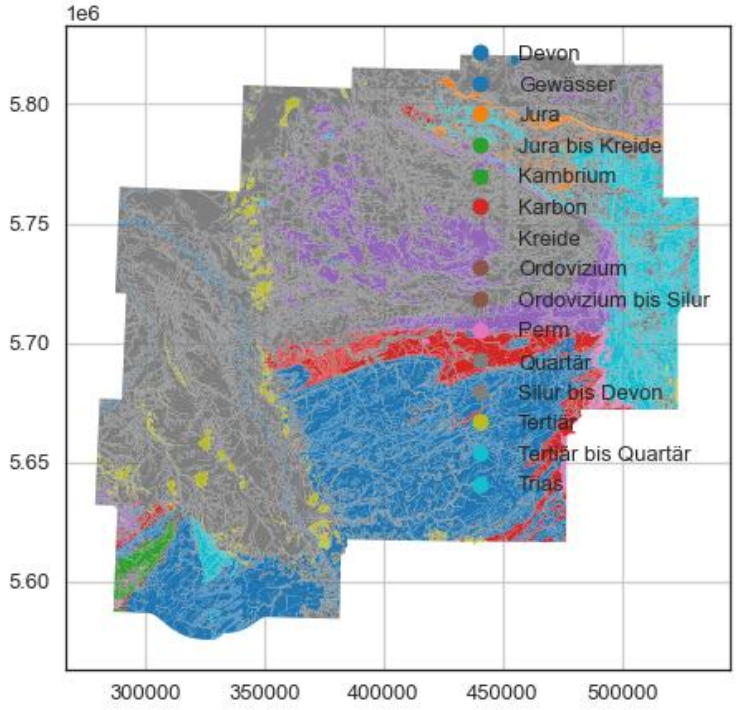
Có thể lấy danh sách các lớp khác nhau trong GeoDataBase bằng cách sử dụng hàm fionas listlayers(..). Sau đó, các lớp khác nhau trong GeoDataBase có thể được tải dưới dạng GeoDataFrames bằng GeoPandas.

```
import geopandas as gpd
import fiona
layer_list = fiona.listlayers(file_path + 'ISGK100.gdb')
layer_list
[10]:
['GK100_Tektonik', 'GK100_Hauptschichten', 'GK100_Deckschichten']
data = gpd.read_file(file_path + "ISGK100.gdb", driver='FileGDB',
layer=layer_list[0])
data.head()
```

	TYP	DATUM	SHAPE_Length	geometry
0	Überschiebung, gesichert	20140911	809.45	MULTILINESTRING ((400321.058 5653696.880, 4001...
1	Überschiebung, gesichert	20140911	695.09	MULTILINESTRING ((399433.510 5653415.030, 3992...
2	Überschiebung, gesichert	20140911	2021.76	MULTILINESTRING ((402635.674 5654612.215, 4023...
3	Überschiebung, gesichert	20140911	1016.71	MULTILINESTRING ((403501.564 5651765.422, 4037...
4	Überschiebung, gesichert	20140911	1085.76	MULTILINESTRING ((404140.075 5652650.414, 4050...

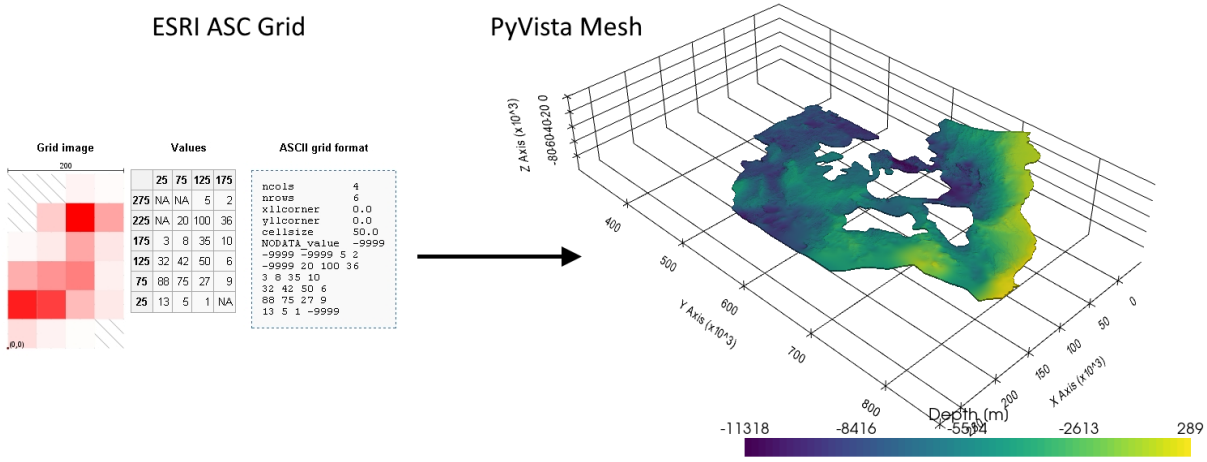
```
import matplotlib.pyplot as plt
data.plot()
plt.grid()
import matplotlib.pyplot as plt
```

```
data.plot(column='SYSTEM', legend=True)
plt.grid()
```



3.2 Mở lưới ESRI ASC và lưới ZMAP

GemGIS cũng có thể đọc các tệp ArcGIS ASC và Lưới ZMAP. Các ví dụ dữ liệu hiển thị bên dưới được lấy từ https://www.nlog.nl/en/scan-2d-seismic-interpretation-and-deep-conversion-dinantian.




```
import gemgis as gg
file_path ='data/45_opening_asc_and_zmap_grids/'
gg.download_gemgis_data.download_tutorial_data(filename="45_opening_asc_and_zmap_grids.zip", dirpath=file_path)
```

3.3 Trực quan hóa trong PyVista

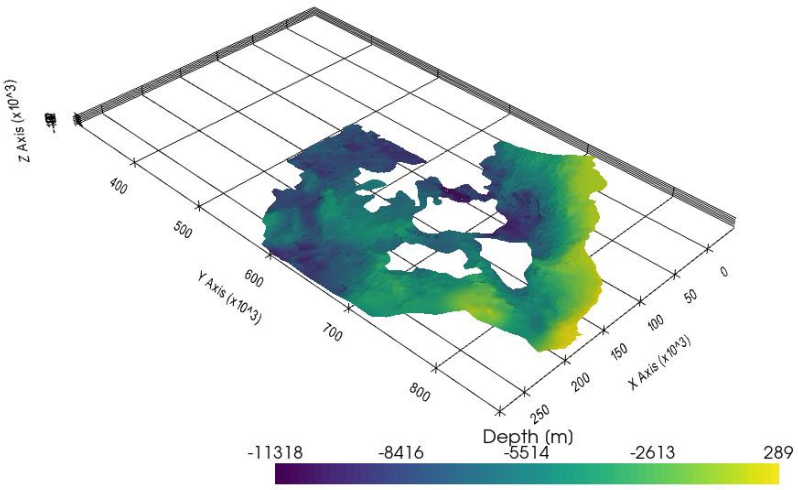
```
grid = gg.visualization.create_structured_grid_from_asc(data=data)
grid
```

Header	Data Arrays

StructuredGrid	Information
N Cells	2880012
N Points	2883540
X Bounds	-4.225e+04, 2.788e+05
Y Bounds	3.060e+05, 8.668e+05
Z Bounds	-1.132e+04, 2.887e+02
Dimensions	2244, 1285, 1
N Arrays	1

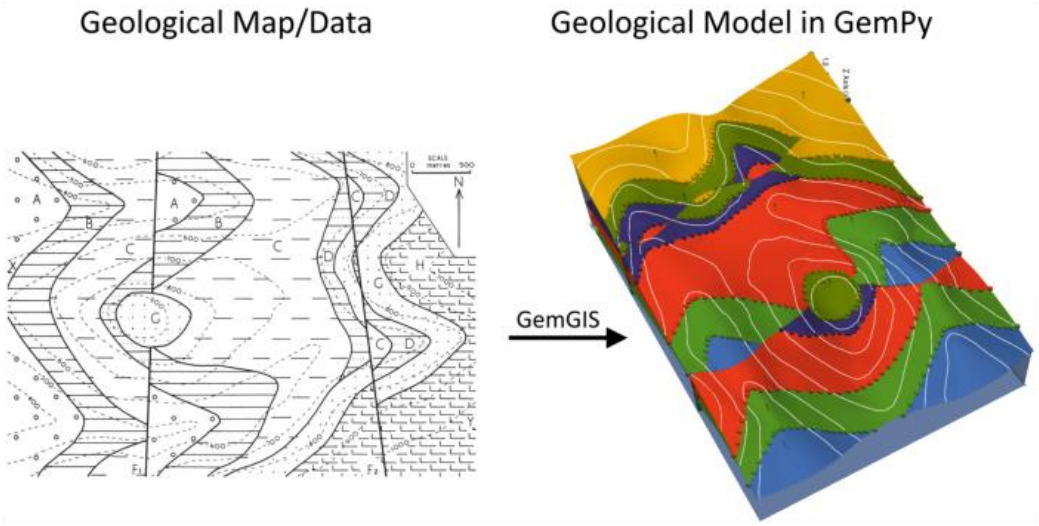
Name	Field	Type	N Comp	Min	Max
Depth [m]	Points	float64	1	-1.132e+04	2.887e+02

```
grid.save(file_path + 'top_dinant_final_tvd.vtk')
import pyvista as pv
contours = pv.read(file_path+'top_dinant_final_tvd_contours.vtk')
sargs = dict(fmt="%.0f", color='black')

p = pv.Plotter(notebook=True)

p.add_mesh(grid, scalars='Depth [m]', nan_opacity=0,
scalar_bar_args=sargs)

p.show_grid(color='black')
p.set_background(color='white')
p.show()
```



4. XÂY DỰNG MÔ HÌNH TỪ BẢN ĐỒ ĐỊA CHẤT

Chuyển đổi bản đồ địa chất bên dưới bằng GemGIS sang mô hình GemPy. Ví dụ này dựa trên dữ liệu số hóa. Khu vực này rộng 3954 m (phạm vi W-E) và cao 2738 m (phạm vi N-S). Phạm vi mô hình dọc thay đổi trong khoảng từ 0 m đến 1000 m. Mô hình thể hiện các lớp nếp uốn (màu vàng đến xanh lục nhạt) được phân tách thành tập hợp lớp thứ hai (xanh lam và tím) bởi sự không đồng nhất. Lớp màu xanh nhạt tượng trưng cho tầng đá gốc. Bản đồ đã được tham chiếu địa lý với QGIS. Ranh giới địa tầng đã được số hóa trong QGIS. Các đường thế nằm cũng được số hóa trong QGIS và sẽ được sử dụng để tính toán hướng cho mô hình GemPy. Các đường đồng mức cũng được số hóa và sẽ được nội suy bằng GemGIS để tạo địa hình cho mô hình.



4.1 Import GemGIS

```
import warnings
warnings.filterwarnings("ignore")
import gemgis as gg
```

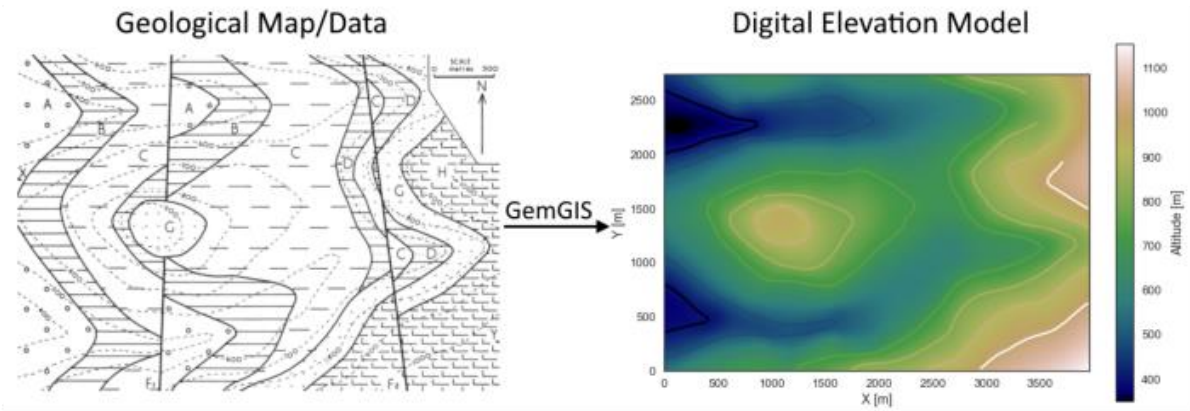
4.2 Nhập thư viện và tải dữ liệu

Tất cả các gói còn lại có thể được tải để chuẩn bị dữ liệu và xây dựng mô hình. Dữ liệu mẫu được tải xuống từ máy chủ bên ngoài bằng cách sử dụng pooch. Nó sẽ được lưu trữ trong một thư mục dữ liệu trong cùng thư mục nơi sổ ghi chép này được lưu trữ.

```
import geopandas as gpd
import rasterio
file_path = 'data/example10/'
gg.download_gemgis_data.download_tutorial_data(filename="example10_faul
ted_folded_layers
```

4.3 Tạo mô hình số độ cao từ các đường đồng mức

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
img = mpimg.imread('../images/dem_example10.png')
plt.figure(figsize=(10, 10))
imgplot = plt.imshow(img)
plt.axis('off')
plt.tight_layout()
```

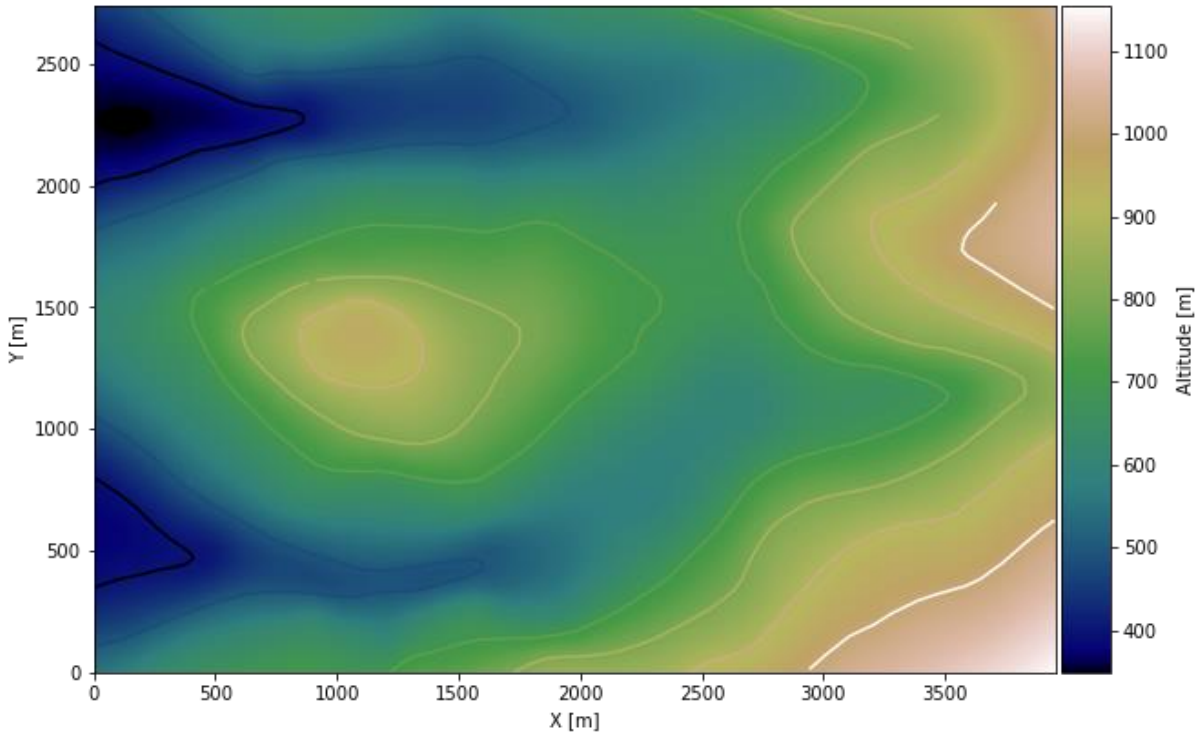



```
topo = gpd.read_file(file_path + 'topo10.shp')
topo.head()
```

	id	Z	geometry
0	None	600	LINESTRING (500.103 2733.663, 594.070 2684.564...
1	None	500	LINESTRING (217.356 2726.044, 324.445 2643.506...
2	None	400	LINESTRING (11.222 2589.327, 69.634 2556.312, ...
3	None	900	LINESTRING (1037.237 1522.677, 1083.798 1526.9...
4	None	800	LINESTRING (912.795 1610.718, 990.677 1620.030...

4.4 Nội suy các đường đồng mức

```
topo_raster = gg.vector.interpolate_raster(gdf=topo, value='Z',
method='rbf', res=10)
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

fix, ax = plt.subplots(1, figsize=(10, 10))
topo.plot(ax=ax, aspect='equal', column='Z', cmap='gist_earth')
im = plt.imshow(topo_raster, origin='lower', extent=[0, 3954, 0, 2738],
cmap='gist_earth')
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax)
cbar.set_label('Altitude [m]')
ax.set_xlabel('X [m]')
ax.set_ylabel('Y [m]')
ax.set_xlim(0, 3954)
ax.set_ylim(0, 2738)
```

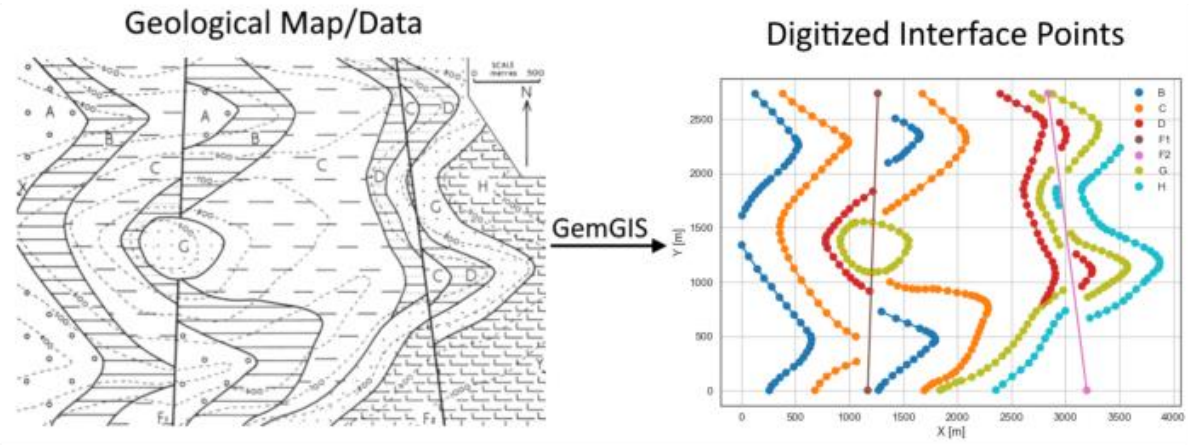


4.5 Điểm giao nhau của ranh giới địa tầng

Các điểm giao nhau sẽ được trích xuất từ LineStrings được số hóa từ bản đồ tham chiếu địa lý bằng cách sử dụng QGIS. Điều quan trọng là cung cấp tên hình thành cho mỗi ranh giới lớp. Vị trí thẳng đứng của điểm giao nhau sẽ được trích xuất từ mô hình độ cao kỹ thuật số bằng hàm GemGIS gg.vector.extract_xyz(). GeoDataFrame kết quả hiện chứa các điểm duy nhất bao gồm thông tin về đội hình tương ứng.

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
img = mpimg.imread('../images/interfaces_example10.png')
plt.figure(figsize=(10, 10))
imgplot = plt.imshow(img)
plt.axis('off')
plt.tight_layout()
interfaces = gpd.read_file(file_path + 'interfaces10.shp')
interfaces.head()
```

	id	formation	geometry
0	None	F1	LINESTRING (1263.266 2736.203, 1170.145 1.855)
1	None	F2	LINESTRING (2839.537 2736.203, 3198.049 1.855)
2	None	B	LINESTRING (128.046 2733.240, 211.007 2641.813...
3	None	B	LINESTRING (3.603 1339.823, 67.941 1234.851, 1...
4	None	C	LINESTRING (681.688 2.279, 724.862 66.616, 789...

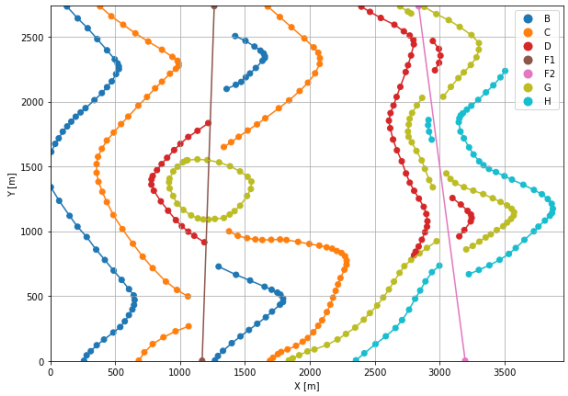


Trích xuất tọa độ Z từ Mô hình số độ cao

```
interfaces_coords = gg.vector.extract_xyz(gdf=interfaces,
dem=topo_raster)
interfaces_coords = interfaces_coords.sort_values(by='formation',
ascending=False)
interfaces_coords.head()
```

	formation	geometry	X	Y	Z
322	H	POINT (2940.276 1706.378)	2940.28	1706.38	814.55
371	H	POINT (3262.810 1592.094)	3262.81	1592.09	879.04
369	H	POINT (3188.314 1716.537)	3188.31	1716.54	888.36
368	H	POINT (3162.071 1769.023)	3162.07	1769.02	887.87
367	H	POINT (3148.526 1840.979)	3148.53	1840.98	889.10

```
fig, ax = plt.subplots(1, figsize=(10, 10))

interfaces.plot(ax=ax, column='formation', legend=True, aspect='equal')
interfaces_coords.plot(ax=ax, column='formation', legend=True,
aspect='equal')
plt.grid()
ax.set_xlabel('X [m]')
ax.set_ylabel('Y [m]')
ax.set_xlim(0, 3954)
ax.set_ylim(0, 2738)
```



4.6 Xây dựng mô hình

```
import gempy as gp
geo_model = gp.create_model('Model10')
geo_model
```

Khởi tạo dữ liệu

```
gp.init_data(geo_model, [0, 3954, 0, 2738, 0, 1000], [100, 100, 100],
             surface_points_df=interfaces_coords[interfaces_coords['Z']
!= 0],
             orientations_df=orientations,
             default_values=True)
geo_model.surfaces
```

	surface	series	order_surfaces	color	id
0	H	Default series	1	#015482	1
1	G	Default series	2	#9f0052	2
2	F2	Default series	3	#ffbe00	3
3	F1	Default series	4	#728f02	4
4	D	Default series	5	#443988	5
5	C	Default series	6	#ff3f20	6
6	B	Default series	7	#5DA629	7

```
gp.map_stack_to_surfaces(geo_model,
                         {
                          'Fault1': ('F1'),
                          'Fault2': ('F2'),
                          'Strata1': ('H', 'G'),
                          'Strata2': ('D', 'C', 'B'),
                         },
                         remove_unused_series=True)
geo_model.add_surfaces('Basement')
geo_model.set_is_fault(['Fault1', 'Fault2'])
```

```
Fault colors changed. If you do not like this behavior, set
change_color to False.
```

	order_series	BottomRelation	isActive	isFault	isFinite
Fault1	1	Fault	True	True	False
Fault2	2	Fault	True	True	False
Strata1	3	Erosion	True	False	False
Strata2	4	Erosion	True	False	False

```
geo_model.add_orientations(X=1200, Y=1350, Z=1025, surface='G',
orientation=[90,5,1])
geo_model.add_orientations(X=3500, Y=350, Z=1000, surface='G',
orientation=[90,6,1])
geo_model.add_orientations(X=3500, Y=350, Z=1000, surface='H',
orientation=[90,6,1])
```

```
geo_model.add_orientations(X=3500, Y=2000, Z=1000, surface='G',
orientation=[90,6,1])
geo_model.add_orientations(X=3500, Y=2000, Z=1000, surface='H',
orientation=[90,6,1])
geo_model.add_orientations(X=2800, Y=2000, Z=1000, surface='D',
orientation=[90,18,1])
geo_model.add_orientations(X=2800, Y=1000, Z=1000, surface='D',
orientation=[90,18,1])
```

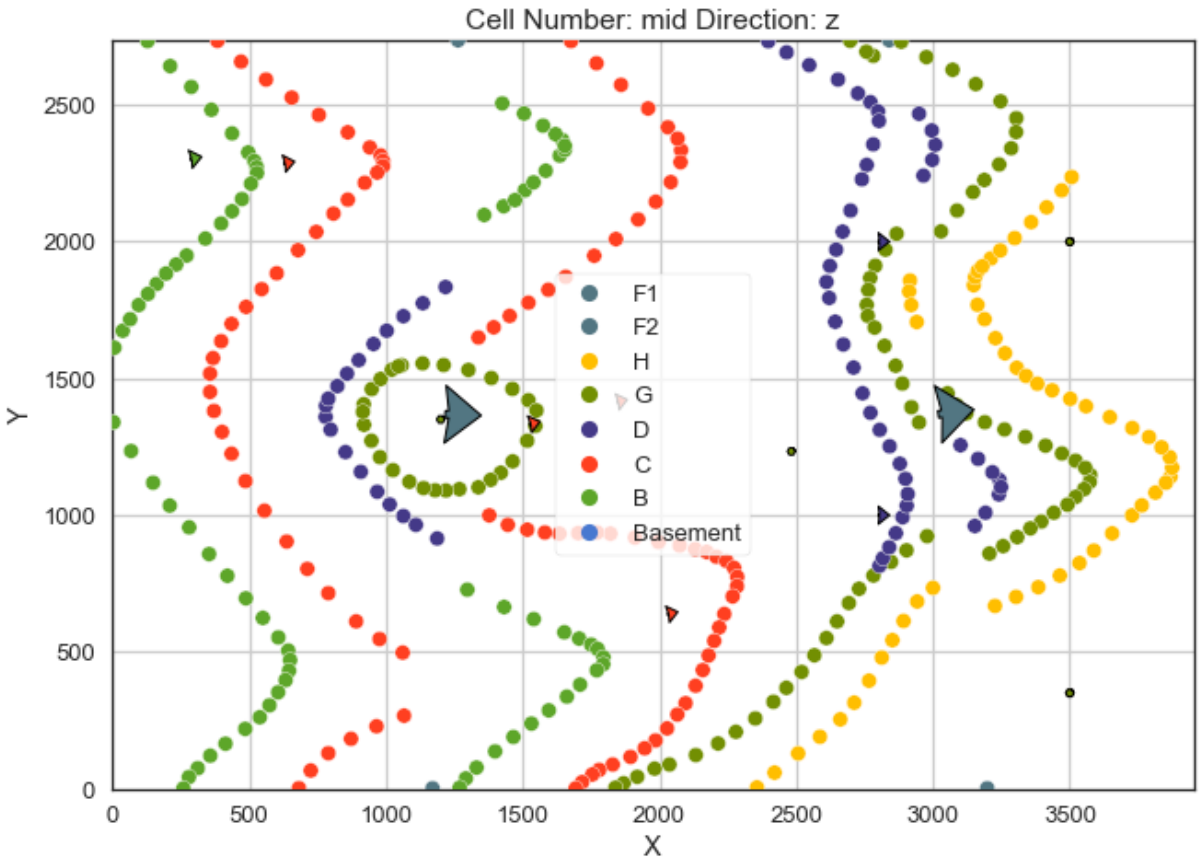
	X	Y	Z	G_x	G_y	G_z	smooth	surface
0	1216.18	1369.03	1000.00	1.00	-0.03	0.00	0.01	F1
1	3018.79	1368.82	1000.00	0.99	0.13	0.00	0.01	F2
10	3500.00	350.00	1000.00	0.10	0.00	0.99	0.01	H
12	3500.00	2000.00	1000.00	0.10	0.00	0.99	0.01	H
7	2481.98	1233.26	750.00	0.10	-0.00	0.99	0.01	G
8	1200.00	1350.00	1025.00	0.09	0.00	1.00	0.01	G
9	3500.00	350.00	1000.00	0.10	0.00	0.99	0.01	G
11	3500.00	2000.00	1000.00	0.10	0.00	0.99	0.01	G
13	2800.00	2000.00	1000.00	0.31	0.00	0.95	0.01	D
14	2800.00	1000.00	1000.00	0.31	0.00	0.95	0.01	D
2	1845.27	1413.90	650.00	0.28	0.07	0.96	0.01	C
3	1524.85	1332.63	750.00	0.29	0.07	0.95	0.01	C
4	627.09	2284.99	550.00	0.29	0.06	0.95	0.01	C
5	2029.39	637.61	650.00	0.28	0.07	0.96	0.01	C
6	284.87	2301.29	450.00	0.31	0.07	0.95	0.01	B

```
gg.utils.show_number_of_data_points(geo_model=geo_model)
```

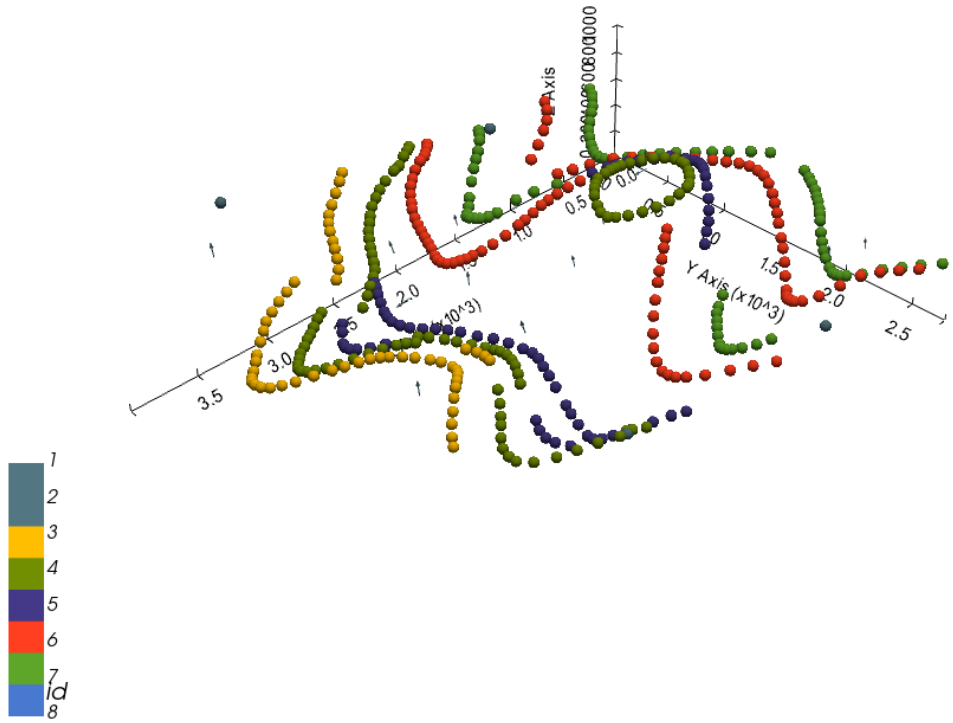
	surface	series	order_surfaces	color	id	No. of Interfaces	No. of Orientations
3	F1	Fault1	1	#527682	1	2	1
2	F2	Fault2	1	#527682	2	2	1
0	H	Strata1	1	#ffbe00	3	55	2
1	G	Strata1	2	#728f02	4	100	4
4	D	Strata2	1	#443988	5	65	2
5	C	Strata2	2	#ff3f20	6	96	4
6	B	Strata2	3	#5DA629	7	79	1
7	Basement	Strata2	4	#4878d0	8	0	0

Tải mô hình độ cao số

```
geo_model.set_topography(
    source='gdal', filepath=file_path + 'raster10.tif')
Cropped raster to geo_model.grid.extent.
depending on the size of the raster, this can take a while...
storing converted file...
Active grids: ['regular' 'topography']
[32]:
Grid Object. Values:
array([[  19.77      ,   13.69      ,    5.        ],
       [  19.77      ,   13.69      ,   15.        ],
       [  19.77      ,   13.69      ,   25.        ],
       ...,
       [3948.99493671, 2713.01824818, 1023.34295654],
       [3948.99493671, 2723.01094891, 1024.94226074],
       [3948.99493671, 2733.00364964, 1026.57104492]])
gp.plot_2d(geo_model, direction='z', show_lith=False,
show_boundaries=False)
plt.grid()
```




```
gp.plot_3d(geo_model, image=False, plotter_type='basic', notebook=True)
```




```
gp.set_interpolator(geo_model,
                    compile_theano=True,
                    theano_optimizer='fast_compile',
                    verbose=[],
                    update_kriging=False
                    )
```

```
Compiling theano function...
Level of Optimization:  fast_compile
Device:  cpu
Precision:  float64
Number of faults:  2
Compilation Done!
Kriging values:
                       values
range                 4912.31
$C_o$                574541.9
drift equations  [3, 3, 3, 3]
```

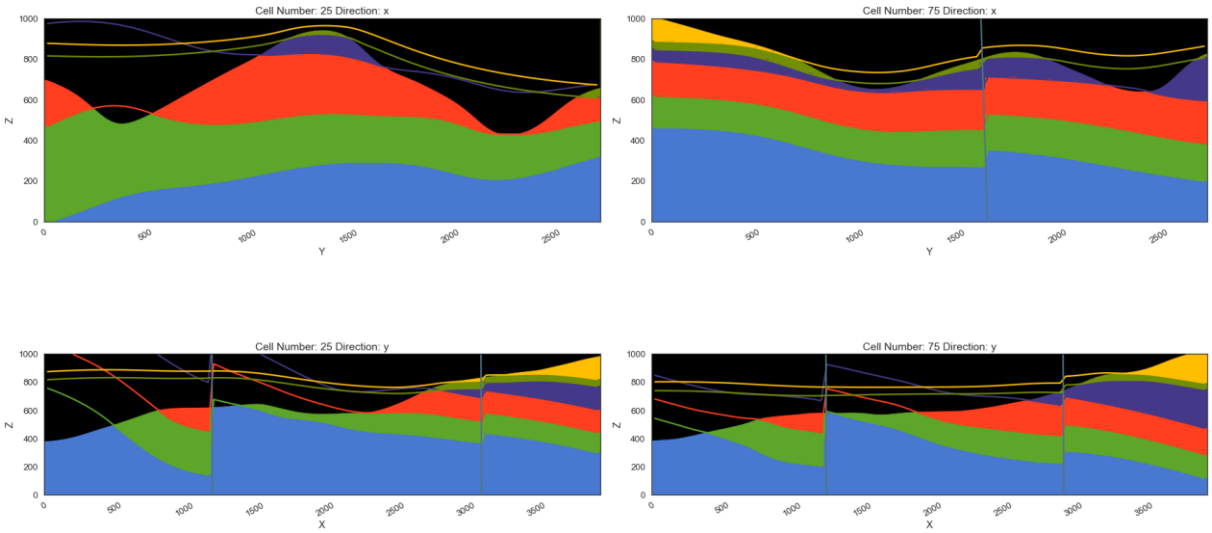
```
<gempy.core.interpolator.InterpolatorModel at 0x1da06704df0>
```

```
sol = gp.compute_model(geo_model, compute_mesh=True)
```

Vẽ mặt cắt

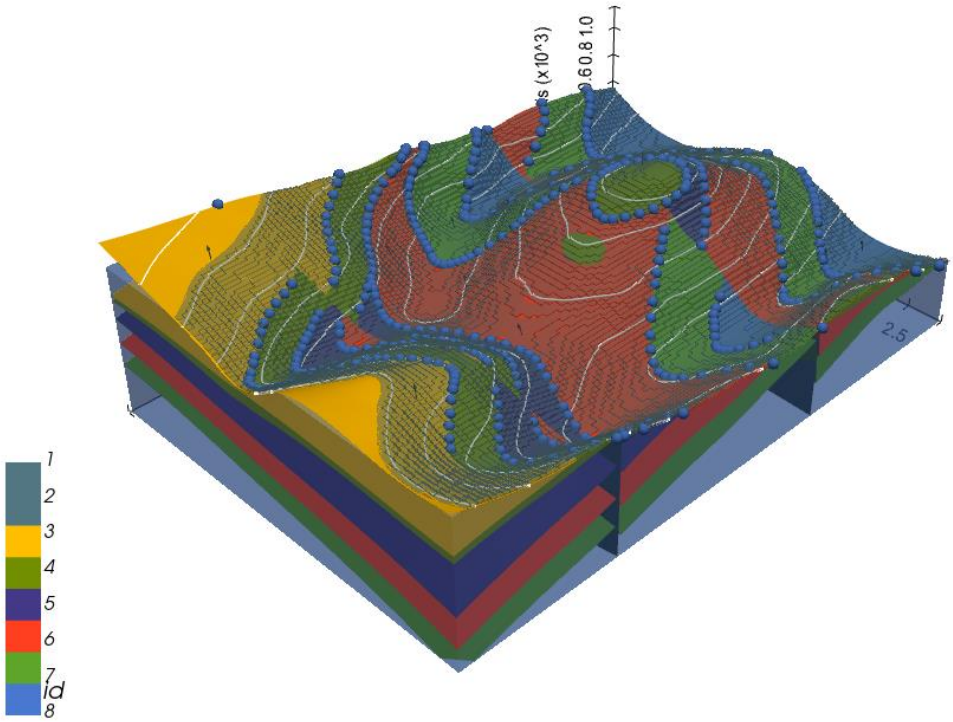
```
gp.plot_2d(geo_model, direction=['x', 'x', 'y', 'y'], cell_number=[25,
75, 25, 75], show_topography=True, show_data=False)
```

```
<gempy.plot.visualization_2d.Plot2D at 0x1da09edd640>
```



Mô hình 3D

```
gpv = gp.plot_3d(geo_model, image=False, show_topography=True,
                 plotter_type='basic', notebook=True, show_lith=True)
```



KẾT LUẬN

Vì GemGIS là một dự án mã nguồn mở và hướng đến cộng đồng nên chúng tôi cần những người để làm cho gói này trở nên tốt hơn, sử dụng đơn giản hơn cho người mới bắt đầu và mở rộng chức năng của nó cho những người dùng cao cấp hơn. Có một số cách để đóng góp cho GemGIS:

Gửi báo cáo lỗi hoặc yêu cầu các tính năng mới
Thêm mới hoặc mở rộng các hướng dẫn và ví dụ hiện có
Sửa lỗi chính tả và cải thiện tài liệu
Thêm chức năng mới vào gói

TÀI LIỆU THAM KHẢO

[1] Gillies, S., & others. (2006--). *OWSLib*. Mapbox. https://github.com/geopython/OWSLib
[2] Gillies, S., & others. (2007--). *Shapely: Manipulation and analysis of geometric objects*. toblerity.org. https://github.com/Toblerity/Shapely
[3] Gillies, S., & others. (2013--). *Rasterio: Geospatial raster i/o for Python programmers*. Mapbox. https://github.com/mapbox/rasterio
[4] Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., R'ıo, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2
[5] Jessell, M., Ogarko, V., Rose, Y. de, Lindsay, M., Joshi, R., Piechocka, A., Grose, L., Varga, M. de la, Ailleres, L., & Pirot, G. (2021). Automated geological map deconstruction for 3D model construction using *map2loop* 1.0 and *map2model* 1.0. *Geoscientific Model Development*, *14*(8), 5063–5092. https://doi.org/10.5194/gmd-14-5063-2021
[6] Jordahl, K., Bossche, J. V. den, Fleischmann, M., & et al. (2021). *Geopandas/geopandas: v0.9.0* (Version v0.9.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.4569086
[7] Sullivan, C. B., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. https://doi.org/10.21105/joss.01450
[8] The pandas development team. (2021). *Pandas-dev/pandas: pandas* (Version v1.2.3) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.4572994
[9] Varga, M. de la, Schaaf, A., & Wellmann, F. (2019). GemPy 1.0: Open-source stochastic geological modeling and inversion. *Geoscientific Model Development*, 1–32.