

# Reporte de selección auxiliatura de investigación p3925.

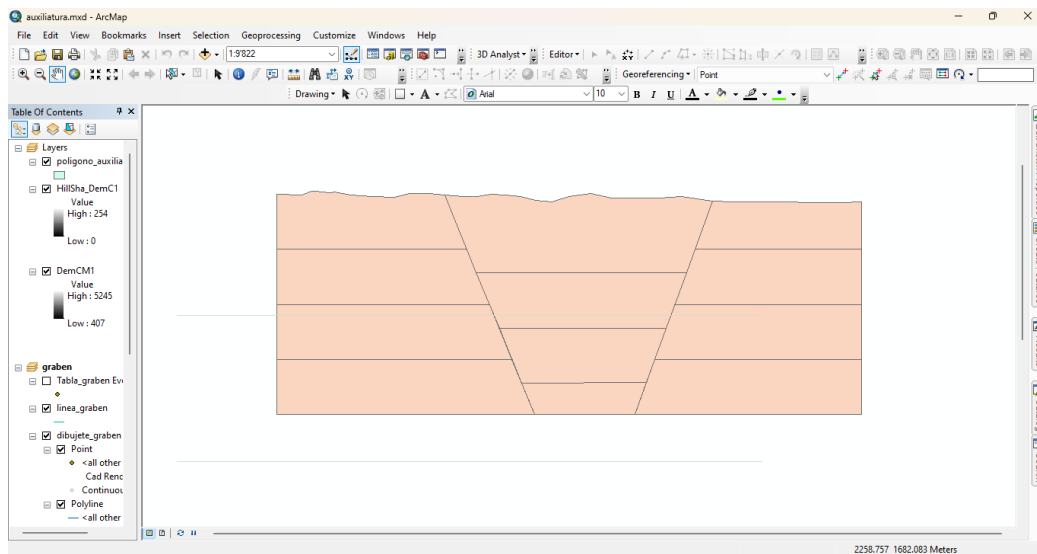
Presentado por: Gabriel Moreno Sánchez

Codigo:2210609

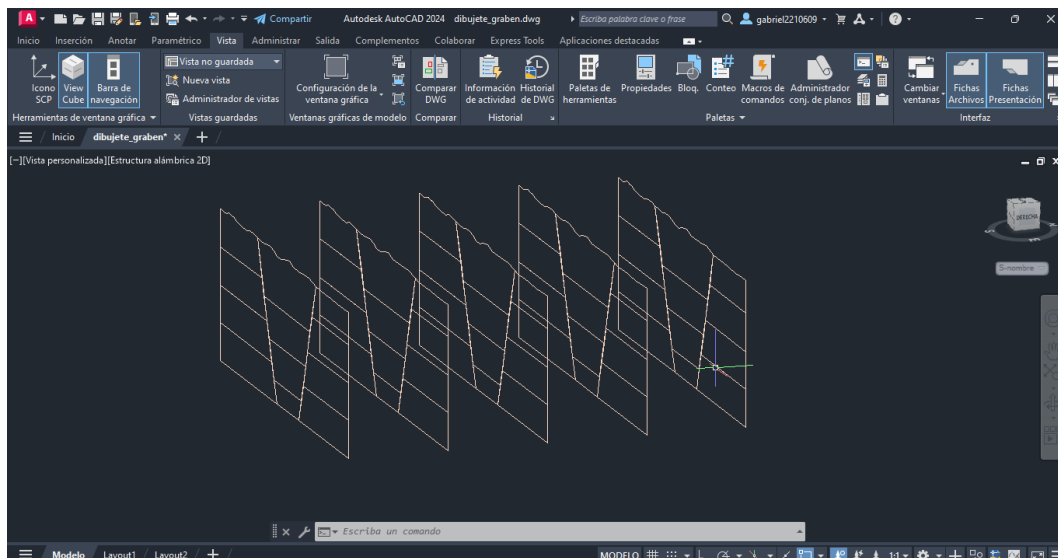
Geología (Octavo semestre)

Creación de modelo Gempy:

La creación del estilo geológico tipo “Graben”, se logró por medio de la creación de un corte en ArcGIS y su exportación a .DWG para poder trabajarlo en el Software libre AutoCAD.



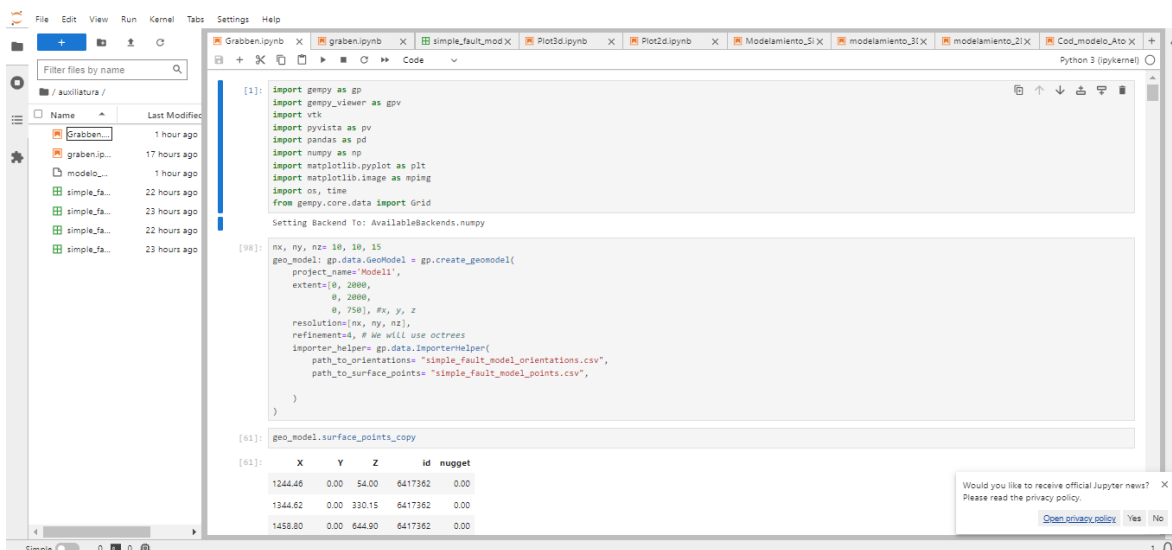
El corte en AutoCAD conserva la escala, pero no se encuentra georreferenciado, así que por medio de las herramientas “desplazamiento” y “órbita 3D” ubico y



georreferencio el corte. Después consigo que el corte abarque todo el dominio y extraigo sus coordenadas para ordenarlas en un archivo CSV.

Los puntos que componen las superficies de las capas y las fallas fueron consignados en el archivo CSV llamado “simple\_fault\_model\_points.csv” y las orientaciones de las capas y las fallas fueron consignadas en el archivo CSV llamado “simple\_fault\_model\_orientations.csv” (acudir a los anexos). Las orientaciones las tomé de cada superficie teniendo en cuenta el azimuth de buzamiento, el valor del buzamiento y la polaridad con valor de 1.

Cree un entorno con las librerías de Gempy==2024.2.0.2, Devito==4.8.11 y Python==3.10, cree un cuaderno en JupyterLab y procedí con el código creando un nuevo modelo (geo\_model), creando una malla rectangular e ingresando los archivos CSV para que pudieran ser leídos.



```
[1]: import gempy as gp
import gempy_viewer as grv
import vtk
import pyvista as pv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mimg
import os, time
from gempy.core.data import Grid

Setting Backend To: AvailableBackends.numpy

[98]: nx, ny, nz= 10, 10, 15
geo_model= gp.data.GeoModel = gp.create_geomodel(
    project_name='Model1',
    extent=[0, 2000,
            0, 2000,
            0, 750], #x, y, z
    resolution=[nx, ny, nz],
    refinement=4, # We will use octrees
    importer_helpers= gp.data.ImporterHelper(
        path_to_orientations= "simple_fault_model_orientations.csv",
        path_to_surface_points= "simple_fault_model_points.csv",
    )
)

[61]: geo_model.surface_points_copy

[61]:
```

X	Y	Z	id	nugget
1244.46	0.00	54.00	6417362	0.00
1344.62	0.00	330.15	6417362	0.00
1459.80	0.00	644.90	6417362	0.00

Would you like to receive official Jupyter news? Please read the privacy policy. [Open external website](#) Yes No

Añadí las superficies y fallas al modelo geo\_model:

```
[6]: gp.map_stack_to_surfaces(
      gempy_model=geo_model,
      mapping_object=
      {
        "Fault_Series": ('Main_Fault_Izq', 'Main_Fault_Der'),
        "Strat_Series1": ('Shale', 'Sandstone_1', 'Sandstone_2'),
      }
    )
    geo_model.structural_frame
```

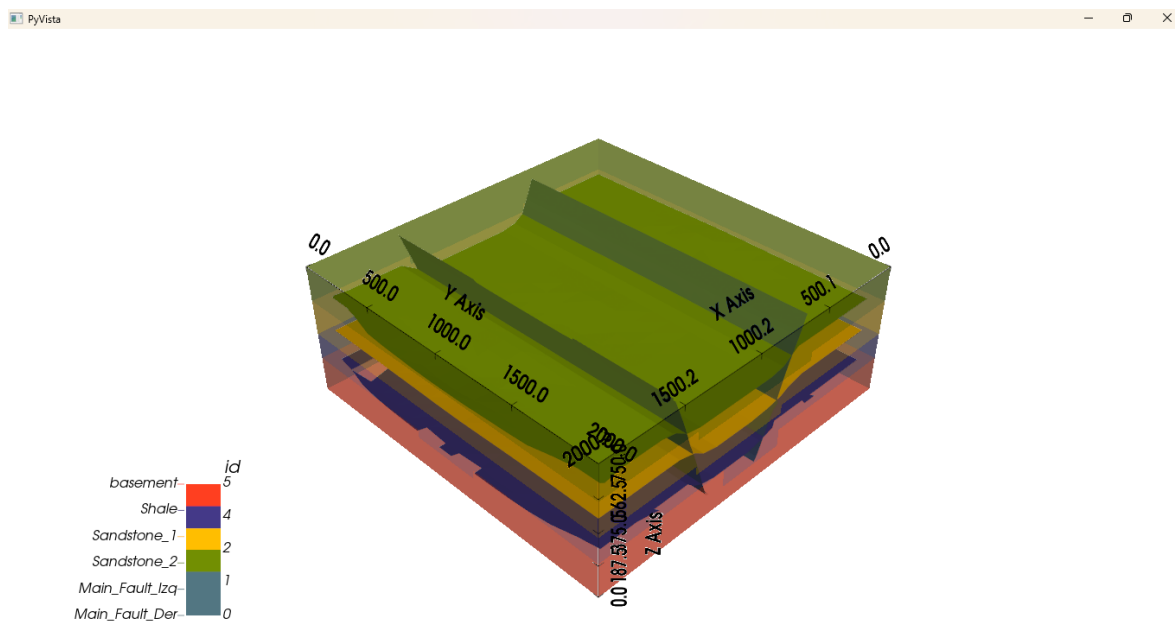
```
[6]:
```

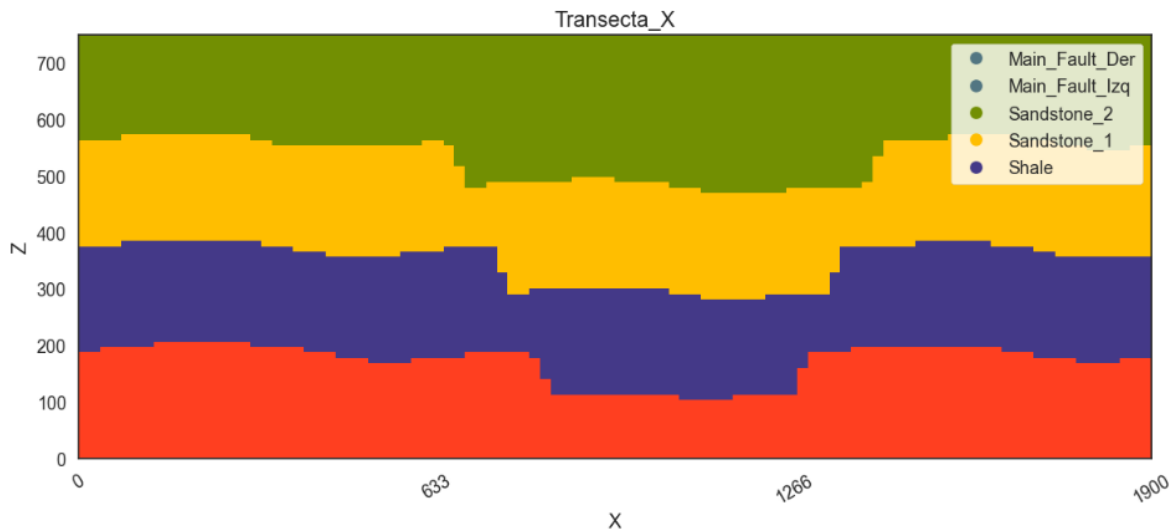
StructuralGroup	
Fault_Series	
StructuralRelation	StackRelationType:ERODE
Elements:	
StructuralElement:	
Name:	Main_Fault_Izq
StructuralElement:	
Name:	Main_Fault_Der

StructuralGroup	
Strat_Series1	
StructuralRelation	StackRelationType:ERODE
Elements:	
StructuralElement:	
Name:	

Would you like to receive official Jupyter news?  
Please read the privacy policy.  
[Open privacy policy](#) Yes No

Y Procedí a plotearlo en 3D y 2D:





Después de haber creado el modelo procedí a calcular el modelo litológico en el centro de las celdas de la malla rectangular.

```
[19]: # Obtener la malla regular
grid = geo_model.grid.regular_grid

# Obtener Las coordenadas directamente de grid.values

File Edit View Run Kernel Tabs Settings Help
Grabben.ipynb X graben.ipynb X simple_fault_model_poin X Plot3d.ipynb X Plot2d.ipynb X Modelamiento_SimPEG.i X modelamiento_3D.ipynb X modelamiento_2D.ipynb X Cod_modelo_Atocad_Vo X
Python 3 (ipykernel)

[68]: velocidades = {
0: 0,
1: 0,
2: 0,
3: 4000, # Velocidad del basamento
4: 3500, # Velocidad del shale
5: 2000, # Velocidad del sandstone_1
6: 3000, # velocidad del sandstone_2
}
print("Valores únicos en lith_block:", np.unique(lith_block))

# Crear un array de velocidades basado en el bloque litológico
velocity_model = np.zeros_like(lith_block, dtype=float)
for i, (surface, velocity) in enumerate(velocities.items()):
    mask = lith_block == i
    velocity_model[mask] = velocity
    print(f"Capa {surface}: {velocity} m/s (asignada a {np.sum(mask)} celdas)")

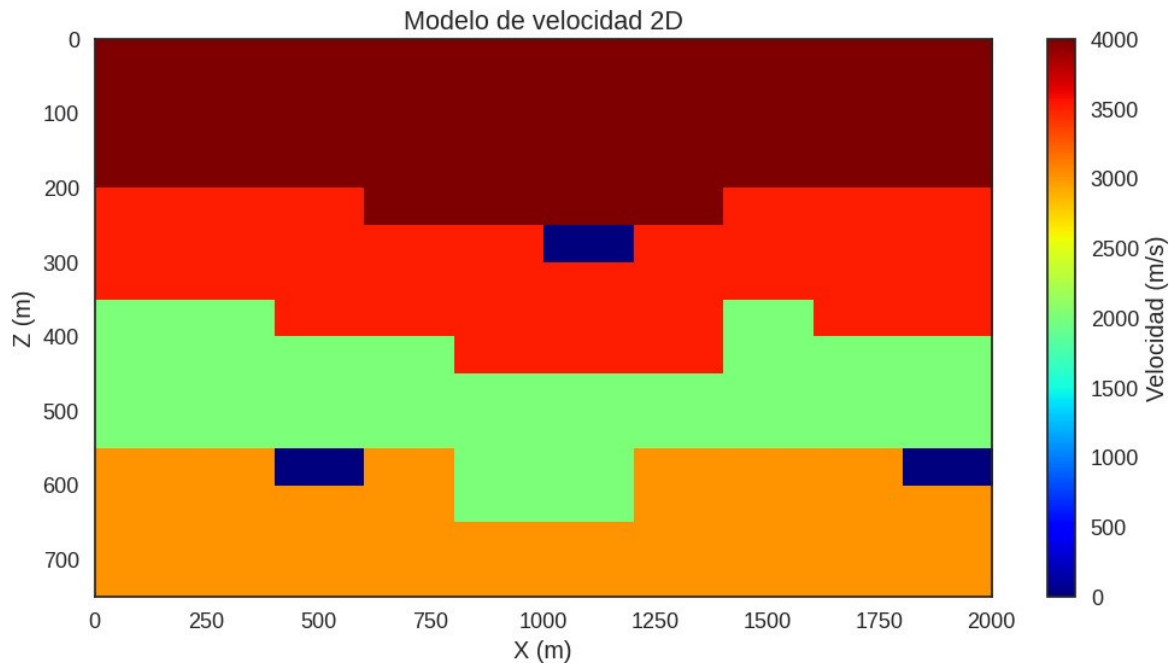
# Imprimir información adicional
print("\nInformación del modelo:")
print(geo_model.structural_frame)

# Reshape el modelo de velocidad a las dimensiones de la malla
velocity_model = velocity_model.reshape(grid.resolution)
print("Forma del modelo de velocidad:", velocity_model.shape)
print("Rango de velocidades:", velocity_model.min(), "-", velocity_model.max())

# Visualizar una sección vertical (X-Z) del modelo de velocidad
plt.figure(figsize=(12, 6))
plt.imshow(velocity_model[:, velocity_model.shape[2]//2, :],
           extent=[grid.extent[0], grid.extent[1], grid.extent[5], grid.extent[4]],
           origin='upper', aspect='auto')
plt.colorbar(label='Velocidad (m/s)')
plt.title('Sección vertical (X-Z) del modelo de velocidad')
plt.xlabel('X (m)')
plt.ylabel('Z (m)')
plt.show()

Valores únicos en lith_block: [3 4 5 6]
```

Después procedí a asignar valores de velocidad a cada capa y a calcular el modelo de velocidades y ajustarlo a la malla rectangular previamente creada.



Seguido de esto imprimí el tamaño del modelo de velocidades y generé un nuevo modelo reajustando el tamaño a una matriz 2D y lo guardé como arreglo en una sección .numpy de dimensiones (10, 15)

```
[102]: print("Forma del velocity_model:", velocity_model.shape)
print("Tamaño del velocity_model:", velocity_model.size)

Forma del velocity_model: (10, 10, 15)
Tamaño del velocity_model: 1500

[123]: # Sección en el plano X-Z
section_y = velocity_model.shape[1] // 2 # Tomar la sección del medio en Y
gempy_vp_2d = velocity_model[:, section_y, :]

# Voltear el eje Z para que la profundidad aumente hacia abajo
gempy_vp_2d = gempy_vp_2d[:, ::-1]

# Guardar esta sección 2D como un archivo .numpy
np.save("modelo_gempy.npy", gempy_vp_2d)

# Imprimir la forma del nuevo modelo 2D
print("Forma del modelo 2D:", gempy_vp_2d.shape)

Forma del modelo 2D: (10, 15)
```

Por último, definí los parámetros del modelo shape=(10,15), spacing=(1., 1.) y origen (0,0). Cargué el array donde se encontraba el modelo de velocidades reajustado llamado "modelo\_gempy.npy" y generé un modelo llamado model2 con él. Así mismo, configuré los parámetros del tiempo=0, los emisores y receptores, añadiendo la geometría de adquisición con la fuente sísmica en el centro del modelo y los receptores distribuidos a igual distancia.

```

# Definir Los parámetros del modelo
shape = gempy_vp.shape
spacing = (grid.dx, grid.dz) # Usar el espaciado de tu modelo original
origin = (grid.extent[0], grid.extent[4]) # Usar el origen de tu modelo original

# Cargar el modelo 2D
gempy_vp = np.load("modelo_gempy.npy")

# Crear el modelo de Devito
model2 = Model(vp=gempy_vp, origin=origin, shape=shape, spacing=spacing,
               nbl=40, space_order=2, bcs="damp")

# Configurar Los parámetros de la simulación
t0 = 0.
tn = 1000.
f0 = 0.008

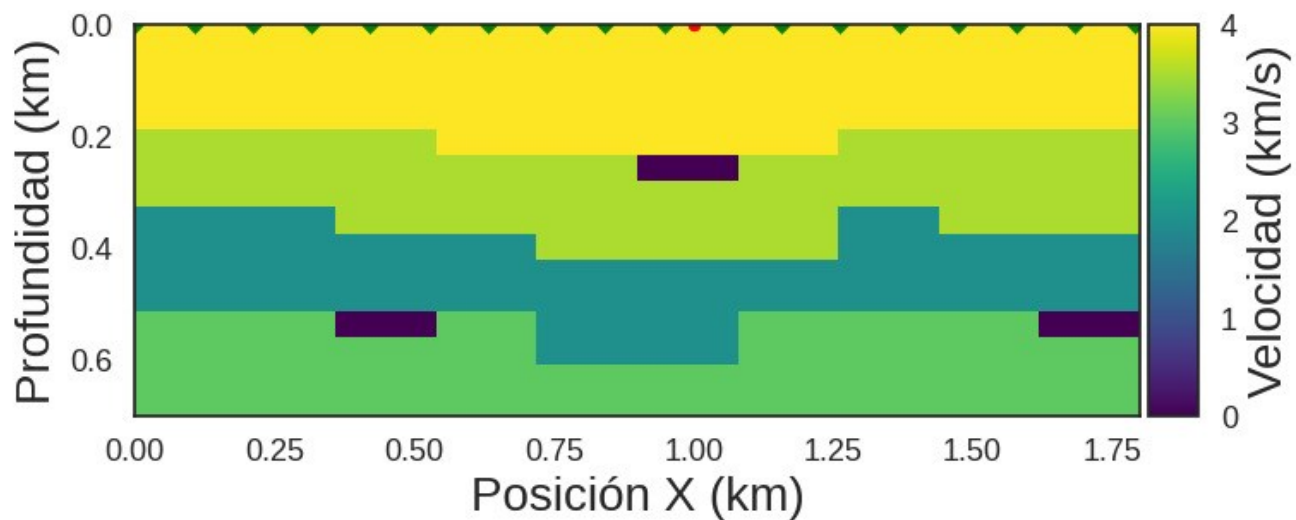
src_coordinates = np.empty((1, 2))
src_coordinates[0, :] = 1000
src_coordinates[0, 1] = 20.

rec_coordinates = np.empty((20, 2))
rec_coordinates[:, 0] = np.linspace(0, 2000, num=20)
rec_coordinates[:, 1] = 20.

geometry = AcquisitionGeometry(model2, rec_coordinates, src_coordinates, t0, tn, f0=f0, src_type='Ricker')

geometry.src.show()

```



Todo lo anterior, junto con el notebook y los demás archivos fueron anexados y también se encuentran en este repositorio:  
[https://github.com/GabrielMoreno09/Modelo\\_Graben\\_Simulaci-n\\_Devito\\_Gempy](https://github.com/GabrielMoreno09/Modelo_Graben_Simulaci-n_Devito_Gempy)

