In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

def read_gxf(file_path):
    grid_data = []
    metadata = {}
    reading_grid = False
    current_key = None

    with open(file_path, 'r') as file:
        for line in file:
            if not line.strip():
                continue


            if line.startswith("#"):
                key = line.strip().split()[0]
                if key == "#GRID":
                    reading_grid = True
                    current_key = None
                    continue
                elif not reading_grid:
                    current_key = key
                    metadata[current_key] = None
            elif current_key and not reading_grid:
                value = line.strip()
                if metadata[current_key] is None:
                    metadata[current_key] = value.split()
                else:
                    metadata[current_key].extend(value.split())
            elif reading_grid:
                grid_data.extend([float(x) for x in line.split() if x])


    print("Parsed metadata:")
    for key, value in metadata.items():
        print(f"{key}: {value}")

    try:
        rows = int(metadata.get("#ROWS", [0])[0])
        cols = int(metadata.get("#POINTS", [0])[0])
    except (ValueError, IndexError, TypeError):
        raise ValueError("Unable to find or parse metadata values for #ROWS or #POINTS, check the file format
```

```python
    grid_array = np.array(grid_data).reshape((rows, cols))


    if "#DUMMY" in metadata and metadata["#DUMMY"] is not None:
        dummy_value = float(metadata["#DUMMY"][0])
        grid_array[grid_array == dummy_value] = np.nan

    return grid_array, metadata


file_path = "/Users/royli/Desktop/az1000ag_gxf"

try:
    grid, meta = read_gxf(file_path)
    print("Metadata is parsed successfully:")
    for k, v in meta.items():
        print(f"{k}: {v}")
except ValueError as e:
    print("Parsing Failed:", e)


if 'grid' in locals():
    plt.figure(figsize=(10, 8))
    plt.imshow(grid, cmap='viridis', origin='lower', extent=[
        float(meta["#XORIGIN"][0]),
        float(meta["#XORIGIN"][0]) + int(meta["#POINTS"][0]) * int(meta["#PTSEPARATION"][0]),
        float(meta["#YORIGIN"][0]),
        float(meta["#YORIGIN"][0]) + int(meta["#ROWS"][0]) * int(meta["#RWSEPARATION"][0])
    ])
    plt.colorbar(label="Grid Value")
    plt.title("GXF Grid Visualization")
    plt.xlabel("X Coordinate (m)")
    plt.ylabel("Y Coordinate (m)")
    plt.show()



vmin, vmax = -100, 100
grid_clipped = np.clip(grid, vmin, vmax)

# Visualize HERE
plt.figure(figsize=(10, 8))
plt.imshow(grid_clipped, cmap='jet', origin='lower', extent=[
```

```python
    float(meta["#XORIGIN"][0]),
    float(meta["#XORIGIN"][0]) + int(meta["#POINTS"][0]) * int(meta["#PTSEPARATION"][0]),
    float(meta["#YORIGIN"][0]),
    float(meta["#YORIGIN"][0]) + int(meta["#ROWS"][0]) * int(meta["#RWSEPARATION"][0])
], vmin=vmin, vmax=vmax)
plt.colorbar(label="Magnetic Field (nT)")
plt.title("Clipped Magnetic Field Data")
plt.xlabel("X Coordinate (m)")
plt.ylabel("Y Coordinate (m)")
plt.savefig('/Users/royli/Desktop/MF.pdf', format='pdf')
plt.show()

from pyproj import Proj, transform


proj_lcc = Proj(proj='lcc', datum='NAD27',
                lat_1=33, lat_2=45,
                lat_0=31, lon_0=-112,
                x_0=0, y_0=0)

proj_geo = Proj(proj='latlong', datum='NAD27')
lon_min, lat_min = -113, 33.0
lon_max, lat_max = -111.5, 34.0

x_min, y_min = transform(proj_geo, proj_lcc, lon_min, lat_min)
x_max, y_max = transform(proj_geo, proj_lcc, lon_max, lat_max)
print(f"Projected extent: x_min={x_min}, x_max={x_max}, y_min={y_min}, y_max={y_max}")


# Extract grid origin and resolution from metadata
x_origin = float(meta["#XORIGIN"][0])
y_origin = float(meta["#YORIGIN"][0])
pt_separation = int(meta["#PTSEPARATION"][0])
rw_separation = int(meta["#RWSEPARATION"][0])

# Calculates the index of the clipping range in the grid
i_min = int((x_min - x_origin) / pt_separation)
i_max = int((x_max - x_origin) / pt_separation)
j_min = int((y_min - y_origin) / rw_separation)
j_max = int((y_max - y_origin) / rw_separation)

grid_subset = grid[j_min:j_max, i_min:i_max]
extent_subset = [x_min, x_max, y_min, y_max]
```

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.imshow(grid_subset, cmap='jet', origin='lower', extent=extent_subset, vmin=-100, vmax=100)
plt.colorbar(label="Magnetic Field (nT)")
plt.title("Magnetic Field Map (113°W to 111.5°W, 33°N to 34°N)")
plt.xlabel("X Coordinate (m)")
plt.ylabel("Y Coordinate (m)")
plt.savefig('/Users/royli/Desktop/MFM.pdf', format='pdf')
plt.show()


lon_phx, lat_phx = -112.0740, 33.4484

#Convert Phoenix's latitude and longitude to projection coordinates
x_phx, y_phx = transform(proj_geo, proj_lcc, lon_phx, lat_phx)
print(f"Phoenix projected coordinates: x={x_phx}, y={y_phx}")

plt.figure(figsize=(8, 6))
plt.imshow(grid_subset, cmap='jet', origin='lower', extent=extent_subset, vmin=-100, vmax=100)

# Locate Phoenix
plt.plot(x_phx, y_phx, 'ro', markersize=8, label="Phoenix")
plt.text(x_phx + 1000, y_phx + 1000, 'Phoenix', color='white', fontsize=12, weight='bold')

plt.colorbar(label="Magnetic Field (nT)")
plt.title("Magnetic Field Map with Phoenix Marked")
plt.xlabel("X Coordinate (m)")
plt.ylabel("Y Coordinate (m)")
plt.legend(loc='upper right')

plt.savefig('/Users/royli/Desktop/MFMPHX.pdf', format='pdf')
plt.show()

# Repeat
lon_smp, lat_smp = -112.0650, 33.3312
x_smp, y_smp = transform(proj_geo, proj_lcc, lon_smp, lat_smp)
print(f"South Mountain Park projected coordinates: x={x_smp}, y={y_smp}")



locations = {
    "South Mountain Park": (-112.0675, 33.3417),
    "Phoenix": (-112.0740, 33.4484),
    "White Tank Mountain": (-112.5903, 33.5696),
```

```
    "Phoenix Mountain": (-112.0306, 33.5861),
    "Estrella Mountain": (-112.3833, 33.3806)
}

projected_locations = {}
for name, (lon, lat) in locations.items():
    x, y = transform(proj_geo, proj_lcc, lon, lat)
    projected_locations[name] = (x, y)

for name, (x, y) in projected_locations.items():
    print(f"{name}: x={x}, y={y}")

plt.figure(figsize=(10, 8))
plt.imshow(grid_subset, cmap='jet', origin='lower', extent=extent_subset, vmin=-100, vmax=100)


for name, (x, y) in projected_locations.items():
    plt.plot(x, y, 'wo', markersize=8)
    plt.text(x + 1000, y + 1000, name, color='white', fontsize=12, weight='bold')


plt.colorbar(label="Magnetic Field (nT)")
plt.title("Magnetic Field Map with Key Locations Marked")
plt.xlabel("X Coordinate (m)")
plt.ylabel("Y Coordinate (m)")
plt.savefig('/Users/royli/Desktop/MFMKey.pdf', format='pdf')

# Plot final fgure
plt.show()
```