# PYTHON GRAPHING: 3D CONTOUR PLOTS

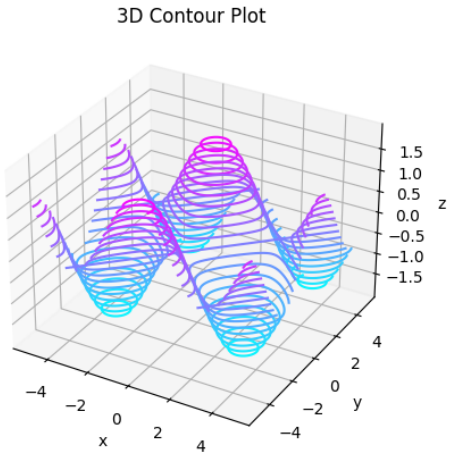
**Author:** Javier Huang
**Editor:** Jim Chen

**Date Created:** 06/10/23
**Date Edited:** 06/10/23

Version 1.0

**Abstract**

In this SOP, the principle of graphing 3D contour plots in the Python programming language is documented.



## Contents



## List of Figures

# 1 Uses of 3D Contour Graphing

A contour plot is a useful visualization technique that allows us to explore the potential relationship between three variables by representing the 3-dimensional relationship in two dimensions. It involves plotting the x-axis and y-axis variables (often referred to as predictors) on the corresponding scales, while the response values (z) are represented by contours.[1]

3D contour plots are commonly used to visualize physical phenomena including:

1. Gravitational Potential

   3D contour plots serve as a valuable tool for visualizing the gravitational potential in a given region of space.

2. Electric and Magnetic Fields

   By graphically depicting field strengths at different points in space, 3D contour plots facilitate the visualization of field patterns and the identification of areas featuring stronger or weaker fields.

3. Temperature Distributions

   3D contour plots are constructed by mapping temperature values onto a three-dimensional grid, which allow for the visualization of temperature variations and gradients, thereby aiding in the analysis of heat transfer processes.

# 2 Installing the Python packages

1. Open the Integrated Development Environment (IDE). This SOP will focus on using Visual Studio Code (VSCode).

2. Go to the Terminal section of VSCode (Fig. 1) or open the Windows Command Prompt (Fig. 2).

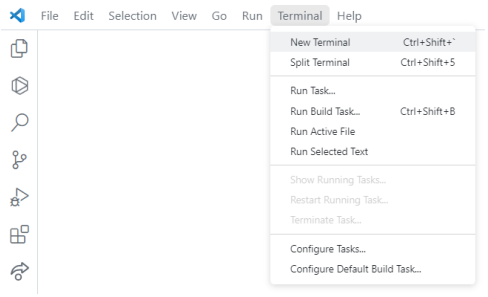



**Figure 1**: VSCode Terminal

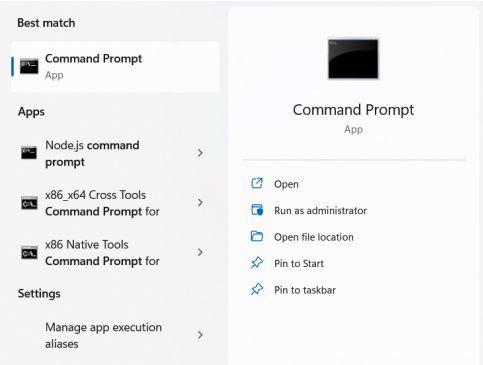



**Figure 2:** Command Prompt

3. The graphing requires two packages, NumPy and MatPlotLib. To install them, type in the following commands separately to either the VSCode Terminal or the Command Prompt Terminal

```
pip install numpy
```

```
pip install matplotlib
```

# 3 Graphing a 3D Contour Plot with Examples

## 3.1 Creating a Generic Function Plot

1. Create a new Python file, and import the libraries

```
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt
```

2. Generate the data for the contour plot. You can use NumPy to create a grid of x, y, and z values. In this example, we'll create a grid ranging from -5 to 5 for both x and y, and use the function $\sin(\sqrt{x^2+y^2})$ for the z

```
# Define the function to be plotted
def f(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))

# Generate x and y values
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)

# Create a grid of x and y values
X, Y = np.meshgrid(x, y)

# Evaluate the function for each combination of x and y
Z = f(X, Y)
```

3. Plot the contour plot by passing the x, y, and z values, along with the number of contour levels (20) and the desired colormap

```
# Create a new figure
fig = plt.figure()
```

```
# Create a 3D axes object
ax = plt.axes(projection='3d')

# Create a 3D contour plot
ax.contour3D(X, Y, Z, 20, cmap='jet')
```

4. Add labels and customize the plot if desired

```
# Set labels for x, y, and z axes
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
```

5. Display the plot

```
# Display the plot
plt.show()
```

6. Finally, execute the code to see the contour plot open in a separate window

**Complete code for graphing a 3D contour plot in Python:**

```
import numpy as np
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

# Define the function to be plotted
def f(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))

# Generate x and y values
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)

# Create a grid of x and y values
X, Y = np.meshgrid(x, y)

# Evaluate the function for each combination of x and y
Z = f(X, Y)

# Create a new figure
fig = plt.figure()

# Create a 3D axes object
ax = plt.axes(projection='3d')

# Create a 3D contour plot
ax.contour3D(X, Y, Z, 20, cmap='jet')

# Set labels for x, y, and z axes
ax.set_xlabel('x')
ax.set_ylabel('y')
```

```
ax.set_zlabel('z')

# Set the title of the plot
ax.set_title('3D Contour Plot')

# Display the plot
plt.show()
```

**The output of the program is displayed in Fig. 3**

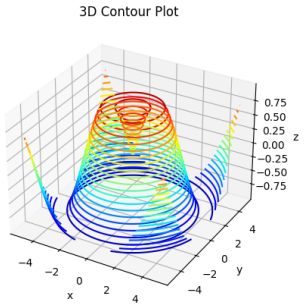



**Figure 3:** Final Generic 3D Contour Plot

## 3.2 Creating a Physics Related Plot

The following example displays the gravitational potential of a point mass.
**Steps 1, 3, 4, 5, and 6** are the same as in subsection 3.1.
Replace **Step 2** with:

```
# Gravitational constant
G = 6.67430e-11

# Define the gravitational potential function for a point mass
def gravitational_potential(x, y, mass):
    r = np.sqrt(x ** 2 + y ** 2)  # Distance from the origin
    return -G * mass / r

# Generate x, y, and z values
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)

# Create a grid of x, y, and z values
X, Y = np.meshgrid(x, y)

# Define the mass of the point mass
mass = 1

# Evaluate the gravitational potential function for each combination of x, y, and z
Z = gravitational_potential(X, Y, mass)
```

**Complete code for creating a physics related 3D contour plot in Python:**

```
import numpy as np
from mpl_toolkits import mplot3d
import matplotlib.pyplot as plt

# Gravitational constant
G = 6.67430e-11

# Define the gravitational potential function for a point mass
def gravitational_potential(x, y, mass):
    r = np.sqrt(x ** 2 + y ** 2)  # Distance from the origin
    return -G * mass / r

# Generate x, y, and z values
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)

# Create a grid of x, y, and z values
X, Y = np.meshgrid(x, y)

# Define the mass of the point mass
mass = 1

# Evaluate the gravitational potential function for each combination of x, y, and z
Z = gravitational_potential(X, Y, mass)

# Create a new figure
fig = plt.figure()

# Create a 3D axes object
ax = plt.axes(projection='3d')

# Create a 3D contour plot
ax.contour3D(X, Y, Z, 50, cmap='jet')

# Set labels for x, y, and z axes
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('potential')

# Set the title of the plot
ax.set_title('3D Contour Plot - Gravitational Potential')

# Display the plot
plt.show()
```

**The output of the program is displayed in Fig. 4**

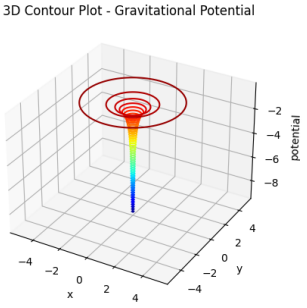



**Figure 4:** Final Physics 3D Contour Plot

## REFERENCES

[1] Contour plots and 3d surface plots. https://support.minitab.com/en-us/minitab/21/help-and-how-to/statistical-modeling/using-fitted-models/supporting-topics/graphs/contour-plots-and-3d-surface-plots/.