Implementing Contourf plot in Python Programming Language

This tutorial introduces the contourf plot concept in matplotlib using Python. The contourf function utilizes the current colormaps to fill the gaps between plot levels. The primary purpose of the contourf function is to fill color between the lowest level and the level above it.

The first color in the contourf function fills the space between the lowest level and the level above it with the specified color. The final color value returned by the contourf function represents Z-axis values that are greater than the plot's highest level.

Let’s get started with the implementation right away!

## Implementation in Python Programming

We'll start by demonstrating a contour plot using a simple f (x, y) function of sine and cosine values and then generate the contourf plot to differentiate between the two plots.

### Contour Plot Implementation

The `plt.contour` function can be utilized to generate a contour plot. A grid of x values, a grid of y values, and a grid of z values are required. The x and y values represent placements on the plot, whereas the contour levels represent the z values. But even before that, we must generate the plotting data. Using the `np.meshgrid` function, which generates a two-dimensional grid from one-dimensional arrays, is one of the most straightforward ways to prepare such data.

Look at the code snippet below.

import numpy as np

def fun(x,y):

   return np.sin(x) \*\* 40 + np.cos(20 + y \* x) \* np.cos(x)

x = np.linspace(0, 10, 50)

y = np.linspace(0, 10, 40)

X, Y = np.meshgrid(x, y)

Z = fun(X, Y)

Now let's have a look at a standard line-only contour plot using the code snippet below.

import matplotlib.pyplot as plt

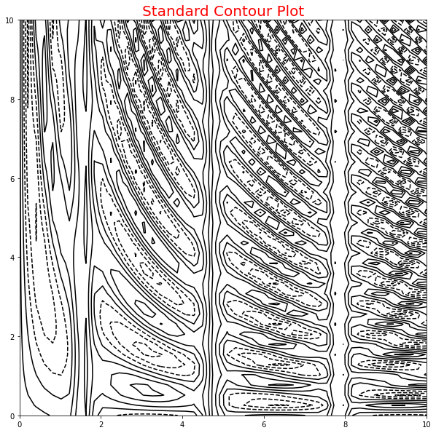
plt.figure(figsize=(10,10))

plt.contour(X, Y, Z, colors='black')

plt.title("Standard Contour Plot",fontsize=20,color="red")

plt.show()

The output of the code looks like the plot shown below.



Notice that when using a single color, negative values are represented by dashed lines and positive values by solid lines by default. Alternately, the lines can be color-coded by using the cmap option to define a colormap.

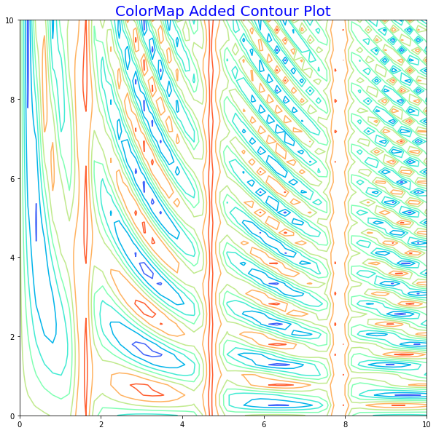
plt.figure(figsize=(10,10))

plt.contour(X, Y, Z, cmap='rainbow')

plt.title("ColorMap Added Contour Plot",fontsize=20,color="Blue")

plt.show()

The output of the above code is displayed below.



Now let us try to implement contourf plot in the next section.

### Contourf Plot Implementation

Our final illustration looks great, but the spaces between the lines may be a little distracting. This can be altered by switching to a filled contour plot using the `plt.contourf()` method (**note the f at the end**), which employs a syntax very identical to `plt.contour()`.

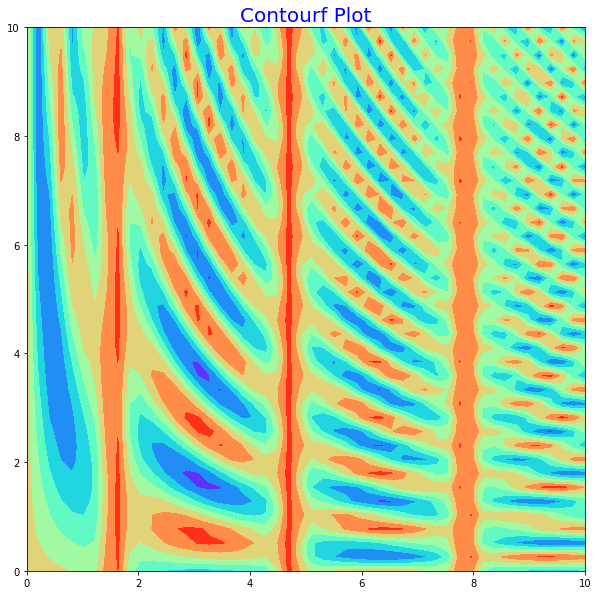
Look at the code snippet below.

plt.figure(figsize=(10,10))

plt.contourf(X, Y, Z, cmap='rainbow')

plt.title("Contourf Plot",fontsize=20,color="Blue")

plt.show()



One may additionally add colorbar using the following code snippet, which clarifies that the black parts are "peaks" and the red regions are "valleys."

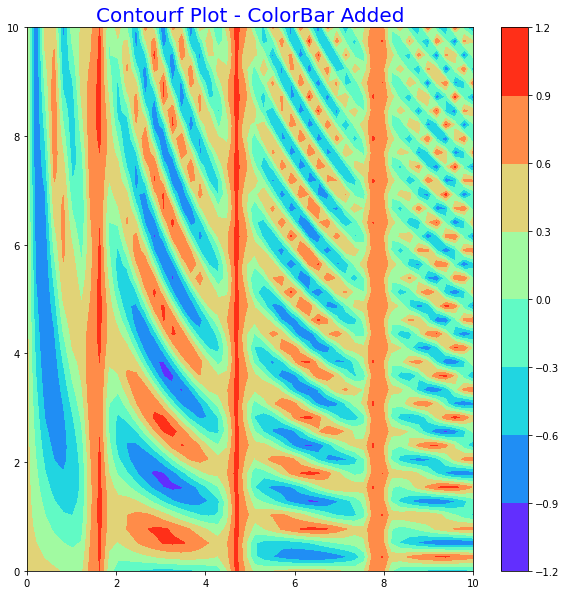
plt.figure(figsize=(10,10))

plt.contourf(X, Y, Z, cmap='rainbow')

plt.title("Contourf Plot - ColorBar Added",fontsize=20,color="Blue")

plt.colorbar()

plt.show()



This plot has the potential flaw of being somewhat "splotchy." In other words, the colour stages are discrete as opposed to continuous, which is not always preferred. This might be rectified by setting the number of contours to an extremely high value; however, this would result in a plot that is inefficient.

Matplotlib must draw a new polygon for each level step. Use the `plt.imshow()` function, which translates a two-dimensional data grid as an image, for a more efficient solution.

plt.figure(figsize=(10,10))

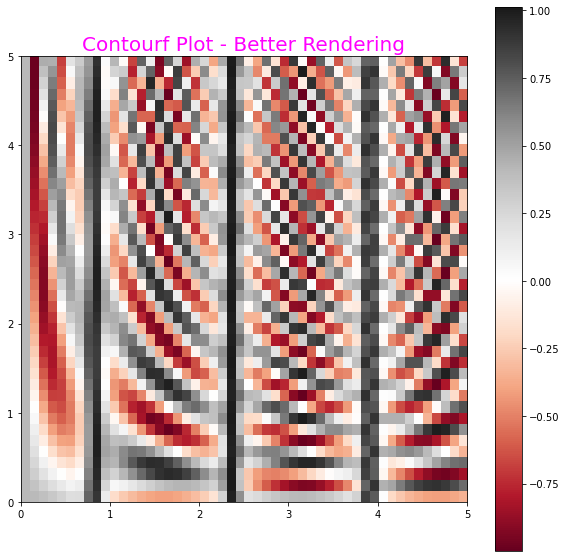
plt.imshow(Z, extent=[0, 5, 0, 5], origin='lower',cmap='RdGy')

plt.colorbar()

plt.axis(aspect='image')

plt.title("Contourf Plot - Better Rendering",fontsize=20,color="Magenta")

plt.show()



Lastly, it can be advantageous to mix contour plots and contourf plots using the following code snippet.

plt.figure(figsize=(10,10))

contours = plt.contour(X, Y, Z, 3, colors='black')

plt.clabel(contours, inline=True, fontsize=8)

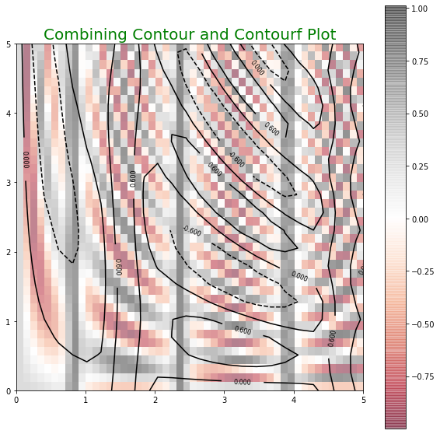
plt.imshow(Z, extent=[0, 5, 0, 5], origin='lower',

           cmap='RdGy', alpha=0.5)

plt.colorbar()

plt.title("Combining Contour and Contourf Plot",fontsize=20,color="Green")

plt.show()



## Conclusion

In this tutorial, you learned about contourf plot and how to implement it in Python programming language. If you liked this tutorial, check out the following tutorials as well:

Happy Learning!