Project Synopsis

Ahmad Saeed Zaidi
Reg no. 2023073

AI 33
FCSE

AI201L
https://github.com/AhmadSaeedZaidi

u2023073@giki.edu.pk
ahmadsaeedzaidi@gmail.com

Project Introduction

The goal of this project is to develop a user-friendly tool that allows individuals to generate 2D and 3D surface plots and scatter plots based on mathematical equations they input, without requiring extensive knowledge of mathematics or programming. By focusing on simplicity and intuitive interactions, this tool will enable users to visualize mathematical surfaces and functions effortlessly, even if they are not well-versed in complex plotting techniques or mathematical syntax.

Libraries to be Used:

Several Python libraries will be utilized to streamline the functionality and interactivity of the tool:

- **SymPy**: For symbolic computation and to convert user-input equations into mathematical expressions that can be evaluated.
- **NumPy**: To generate numerical data points by evaluating mathematical expressions on grids or specific intervals.
- **Plotly**: A powerful library for interactive plotting, which will render 2D and 3D scatter plots and surface plots based on the generated data.

  Maybe:

- **Tkinter (or PyQt/Streamlit)**: For creating the graphical user interface (GUI) that will allow users to input their equations and visualize the results seamlessly.

GUI Options

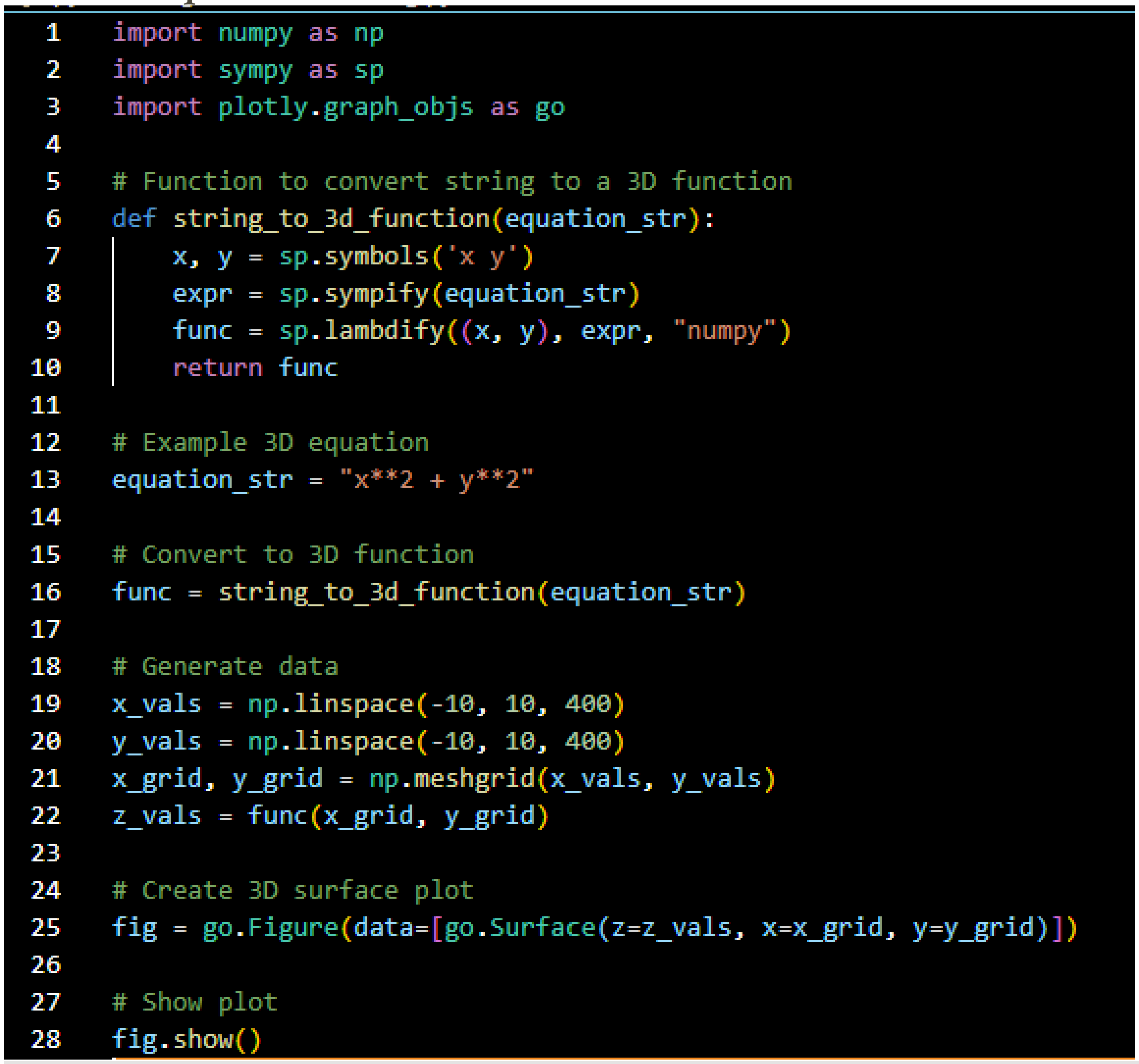
Several GUI frameworks are being considered to handle user interaction:

- **Tkinter**: A lightweight, built-in Python GUI library that provides basic functionality for input fields, buttons, and plots.
- **PyQt**: A more feature-rich option, offering greater flexibility and a modern look for desktop applications, ideal for more complex layouts.
- **Streamlit (most likely)**: A web-based, interactive Python library that could enable users to input their equations and view the plots in a browser environment, simplifying deployment.

Interaction Between Components:

The graphical interface will serve as the front-end where users can input mathematical equations in string form. Once submitted, the equation will be processed by **SymPy**, converting it into a symbolic expression. This expression will then be evaluated using **NumPy** over a specified range or grid. The resulting data points will be passed to **Plotly**, which will dynamically generate the appropriate 2D or 3D plot based on user settings. The GUI will also allow users to adjust plot settings, such as the domain and range of variables, to customize the visualization further.

Code Sample:



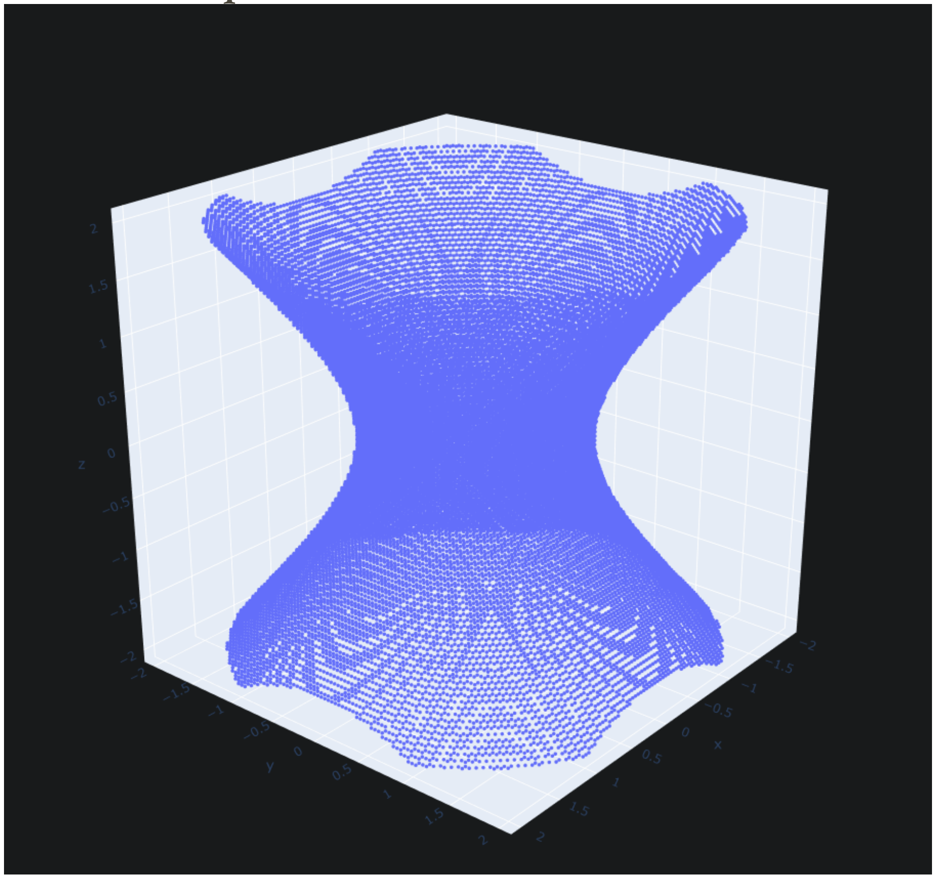
```python
import numpy as np
import sympy as sp
import plotly.graph_objs as go

# Function to convert string to a 3D function
def string_to_3d_function(equation_str):
    x, y = sp.symbols('x y')
    expr = sp.sympify(equation_str)
    func = sp.lambdify((x, y), expr, "numpy")
    return func

# Example 3D equation
equation_str = "x**2 + y**2"

# Convert to 3D function
func = string_to_3d_function(equation_str)

# Generate data
x_vals = np.linspace(-10, 10, 400)
y_vals = np.linspace(-10, 10, 400)
x_grid, y_grid = np.meshgrid(x_vals, y_vals)
z_vals = func(x_grid, y_grid)

# Create 3D surface plot
fig = go.Figure(data=[go.Surface(z=z_vals, x=x_grid, y=y_grid)])

# Show plot
fig.show()
```

Above is the simplest case of the code, an explicitly defined 3d function. Sympy can innately handle this with the lambdify function, that does most of the semantic analysis for us. But this doesn't work for many cases, one which I will show below.

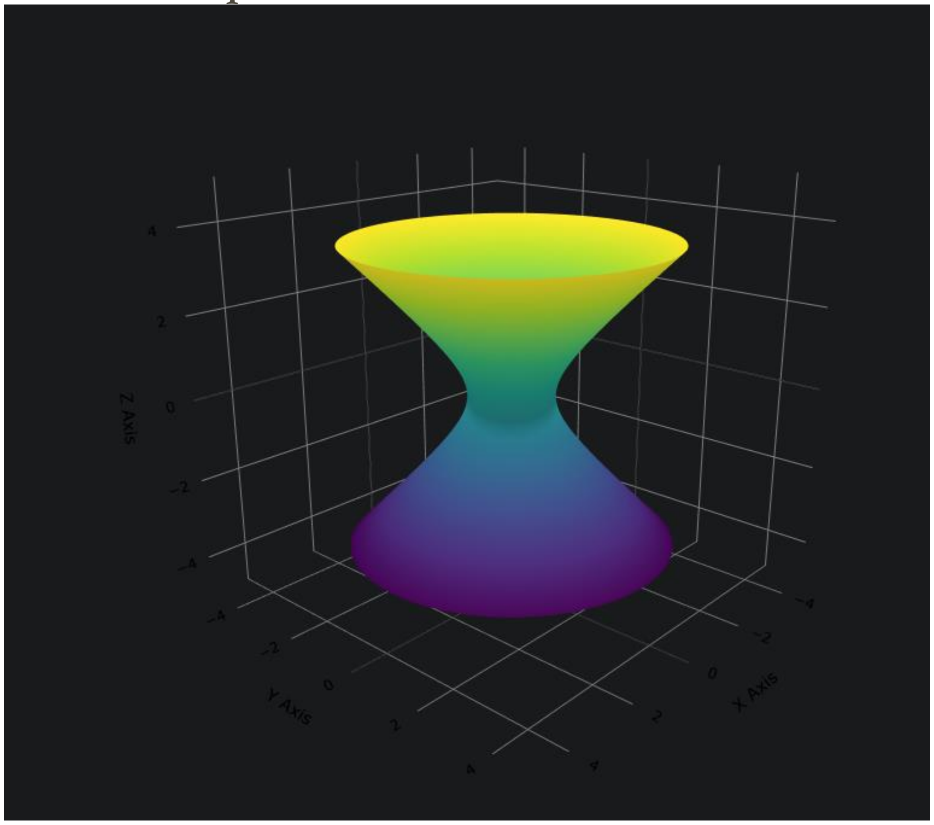
Slowly, I plan to add better pre-processing methods, more error handling, better output formatting, and better semantic analysis. Perhaps even the use of an LLM API.

Currently it is difficult to plot a perfect continuous surface sheet for implicit functions, but that is the goal.

Current Output:



Desired Output:



Documentation and Version Control:

All project files and updates will be systematically uploaded to a GitHub repository. This repository will serve as a central hub for version control and collaboration. Each iteration of the project will be documented with detailed commit messages that describe the changes made in each update. These commit messages will include helpful comments to explain modifications, improvements, bug fixes, or new features added in each version. This approach ensures transparency, maintains a history of the project's evolution, and facilitates easy collaboration or future maintenance.

https://github.com/AhmadSaeedZaidi