Mathematical Plotting Application

Ahmad Saeed Zaidi
Reg no. 2023073

AI 33
FCSE

AI201L
https://github.com/AhmadSaeedZaidi

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

Abstract

The goal of this project was to develop a user-friendly tool that allows individuals to generate 2D and 3D surface 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 Used:

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

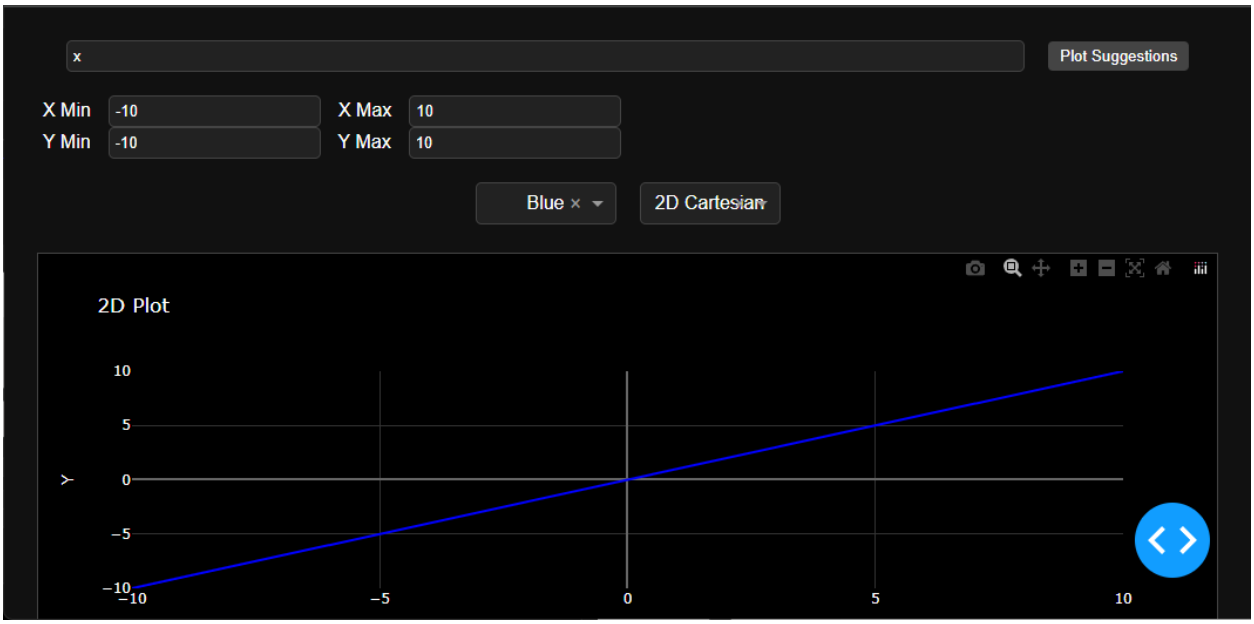
- **Dash:** For creating the web-based interface and handling interactive callbacks between the UI components and plotting functionality.
- **SymPy:** For symbolic computation and converting user-input equations into mathematical expressions that could be evaluated.
- **NumPy:** To generate numerical data points by evaluating mathematical expressions on grids and specific intervals.
- **Plotly:** A powerful library for interactive plotting, which rendered 2D and 3D plots including explicit functions, implicit curves, parametric curves, and surface plots.

Additional tools:

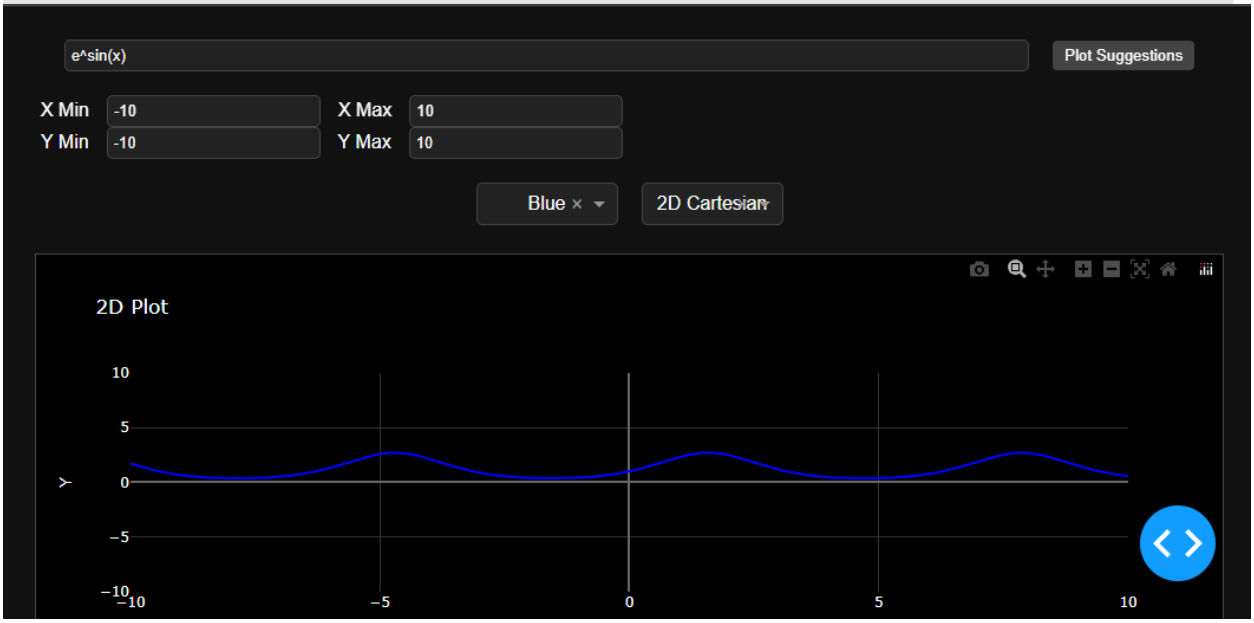
- **CSS:** For styling the dark-themed interface and creating responsive layouts.
- **Git:** For version control and managing the project's source code.

BlackBox Testing:

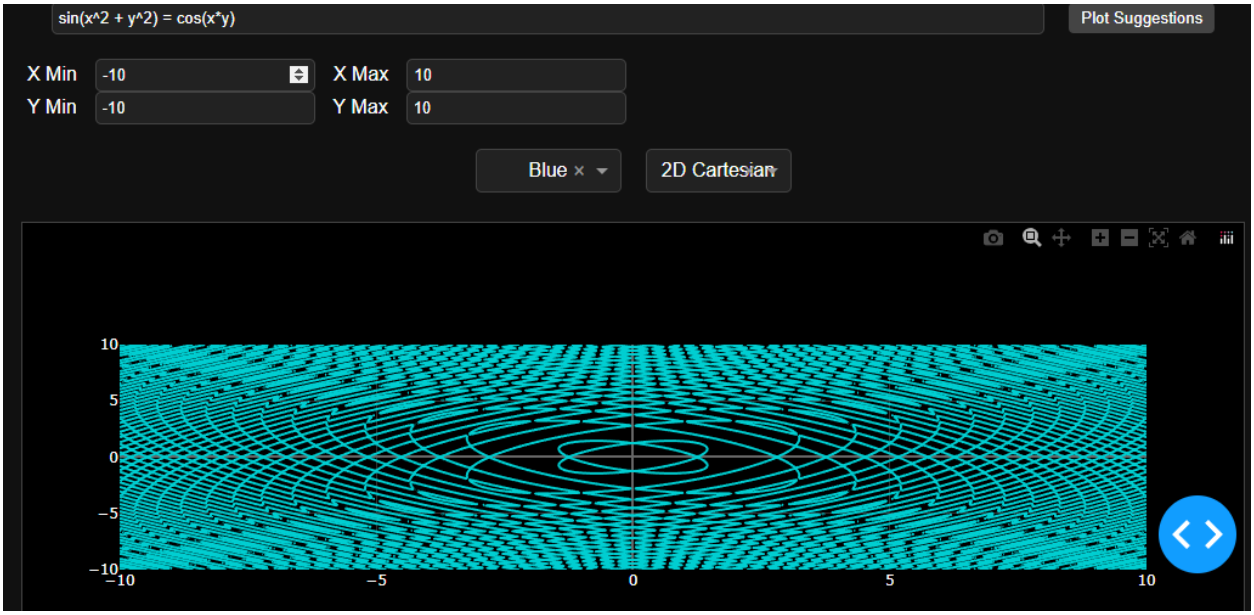
2D Graphing: Simplest case: y = x



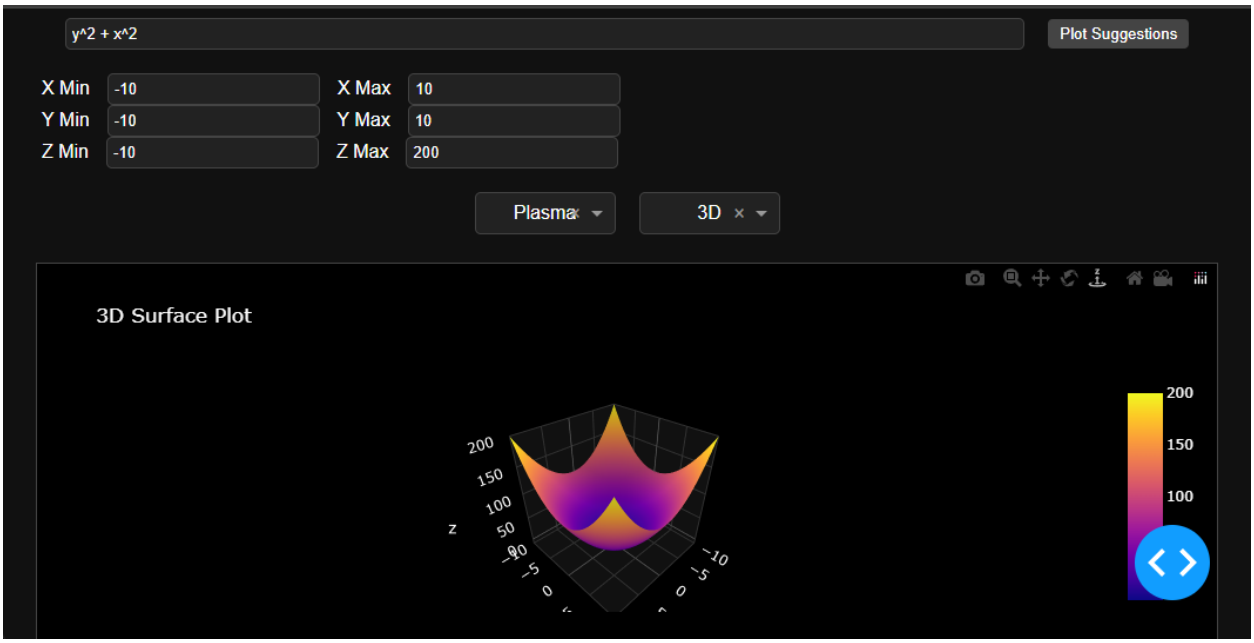
2D Graphing: more complex case: y = $e^{\sin(x)}$



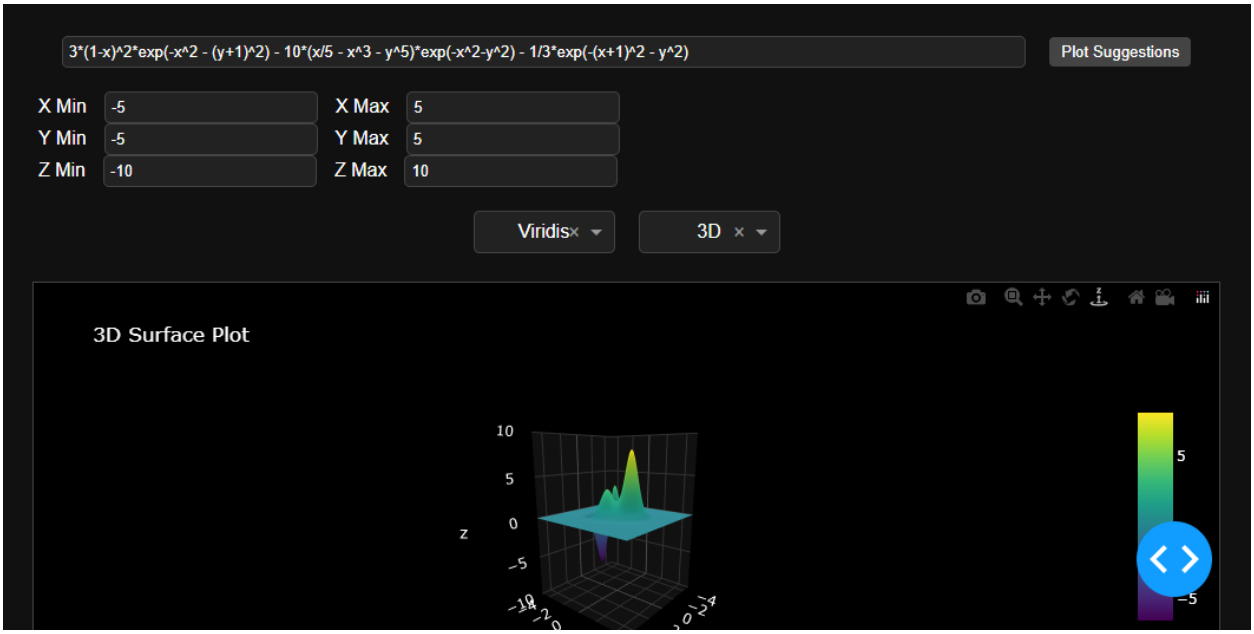
2D Graphing: Implicit Case: $\sin(x^2 + y^2) = \cos(x*y)$



3D Graphing: Explicit Case: Paraboloid



3D Graphing: Extreme Case: Peaks Function



Boundaries and limitations of the project

There are several notable limitations. The tool is restricted to real-valued equations and cannot process complex numbers or differential equations. In 2D mode, equations are limited to two variables (x,y), while 3D mode supports three variables (x,y,z). However 3d equations are only supported in the form z = f(x,y). Performance considerations necessitate a maximum grid resolution for surface plotting, particularly noticeable in complex 3D visualizations. The parser, while robust for standard mathematical notation, may struggle with more exotic expressions or unconventional syntax. Error handling, while functional, provides basic feedback without detailed mathematical context. Additionally, the system cannot simultaneously plot multiple equations or solve systems of equations.

Regardless I think the program achieves it's initial goal for providing a simple to use plotting application, that accepts a decent variety of notation.

Tools and Technologies Used:

The project was developed using Python 3.12 as the core programming language, running on a standard Windows development environment, VS Code IDE. The web application was built using the Dash framework, which provided the foundation for creating an interactive user interface in the browser. Git was employed for version control, allowing systematic tracking of changes and development progress. The development process was significantly enhanced using Claude 3.5 AI, which assisted in code generation, debugging, and making architectural decisions.

Design Approach:

The application's architecture was built around several key Python libraries:

Dash: Handles the web interface and user interactions

Plotly: Powers the interactive plotting capabilities

SymPy: Manages mathematical expression parsing and symbolic computation

NumPy: Performs numerical computations and data generation for plotting

The codebase is organized into logical directories:

```
project/
├── d2/             # 2D plotting functions
├── d3/             # 3D plotting functions
├── preprocessing/  # Input validation and processing
├── utils/          # Utility functions
└── assets/         # CSS and static files
```

Findings:

Key Achievements:

• Successfully developed an interactive mathematical visualization tool with real-time plotting capabilities

• Implemented comprehensive equation parsing supporting multiple input formats (e.g., "2x", "sin(x)")

• Created four distinct plotting modes: 2D explicit, 2D implicit, 2D parametric, and 3D surfaces

• Developed an intuitive dark-themed interface with eye-friendly colour schemes and gradient options

• Integrated interactive features including zoom, pan, and 3D rotation controls

• Built a plot suggestion system to showcase various mathematical patterns and functions

• Established robust error handling with real-time feedback for invalid expressions

• Achieved seamless visualization of complex mathematical expressions including the Peaks function and implicit relations

These results demonstrate the tool's effectiveness in bridging the gap between mathematical expressions and their visual representations.

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/AI201-Project