System Architecture and Purpose

Group 21

March 2025

Overview Text Rendering Architecture

Matplotlib provides a powerful text rendering system that supports **Standard text rendering using *Text objects*** and **Mathematical expressions rendering using *MathText* or *LaTeX*** (usetex=True). It uses multiple backends (Agg, PDF, SVG, etc.), but our modifications primarily impact Agg-based rendering (**backend_agg.py**), which is a very commonly used rendering engine.

Matplotlib's Text Rendering Pipeline

Figure 1 gives a simple overview of matplotlib's text rendering pipeline, which contains some key methods used when (only) rendering text objects.

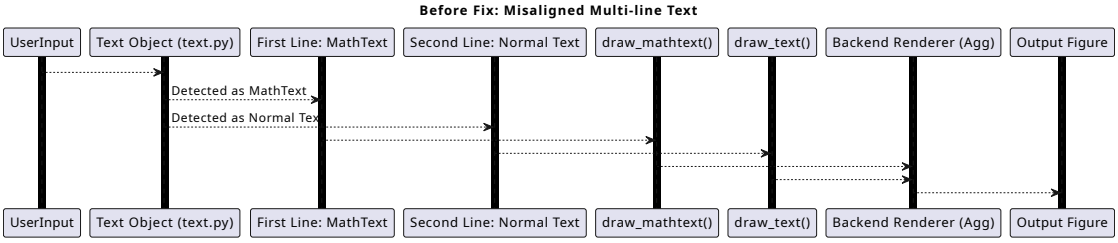

Figure 1: Pipeline before fix

- It begins with **Text Object** creation (e.g., ax.text(), plt.text() or Text()). Users could define text objects with **normal text or MathText**.
- The main text processing is in **text.py**. The Text class determines text properties such as font, alignment, and whether the text contains mathematical expressions. _get_layout() computes the text layout, including character positions and spacing.
- The next key step is rendering, which uses **draw()** as a key method. If the text contains MathText, it is processed via draw_mathtext(). Otherwise, it is rendered as standard text using draw_text().
- In this process, **Text Object** cached the metrics (e.g., size and spacing between characters) to accelerate rendering.
- The last step is backend rendering, different backends are called for different platforms (e.g., macos or windows) and different format (e.g., pdf or svg).

Issues in the Original System and Analysis

Before our fix, Matplotlib incorrectly handled multi-line text containing both MathText and normal text, causing misalignment of vertical elements (e.g., ——). The issue originated from different spacing rules between MathText and standard text. MathText had different glyph positioning and kerning compared to normal text. Also, character metrics are cached when the Text object first met the character; it has an influence on the alignment.

Code Changes in Context

Figure 2 shows the pipeline after our fix.

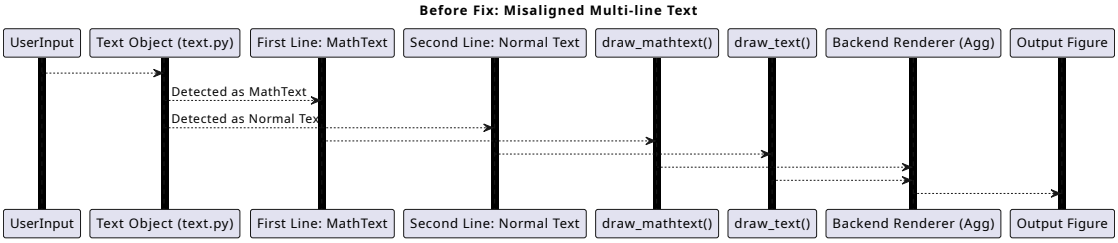

Figure 2: Pipeline after fix

We introduced changes in:

- New Attribute **self.exist_math** in **Text Class**. It tracks whether a text object contains MathText and unifies rendering decisions. It extends Text state management; requires synchronization with content updates.
- Refactored draw() in text.py. Before rendering, we check for MathText in the input. If self.exist_math = True, we ensure the entire text block is processed as MathText.
- Updated _get_layout() and draw_mathtext(). It Forces mixed text to be treated as math text, simplifying layout logic at the cost of suboptimal regular text handling.

Limitations and Future Refactoring

While it's not a perfect/solid solution for this issue, there are still limitations. Also, we proposed future refactoring.

- Hard-coded conditional branches and global flag may conflict with dynamic content updates.
- Cross platforms problem. Different platforms may use different backends, we need to resolve it in the future.
- For the future refactoring, we think a solid solution is modifying the calculation of MathText and normal text metrics (e.g., position and space between characters, called kerning in matplotlib). Also, we should take caching mechanism into consideration.