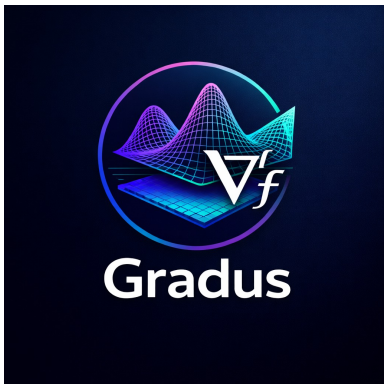
Università Commerciale
Luigi Bocconi

Software Engineering Project

Bocconi University

GRADUS

Derivative symbolic calculator



Authors:
Ceria Gregorio
Bacci Daniele

January 10, 2026

Introduction

We have developed Gradus, a web application for derivatives calculation and functions analysis designed to support learning. The idea for this project came from a simple conversation between colleagues, during which we noticed that many existing web applications for mathematical calculators or related tools do not provide adequate support in these areas. For this reason, we have designed an application that includes features we found to be lacking elsewhere. These tools are not intended solely to compute and display results, but also to provide detailed visualizations.

Gradus is implemented in Python. Specifically, Gradus was developed using three fundamental libraries: SymPy for symbolic computation, NumPy for numerical evaluation, and Matplotlib for data visualization. To provide an interactive user experience, we used Streamlit as the web framework.

Project components

Our app is structured with a two-column interface. On the left side, the user can choose whether to analyze a univariate or multivariate function and can also find a brief help section that explains the input format and some available options. The right side contains the fields where input can be entered. Based on the selection on the left, the layout on the right updates accordingly.

For the univariate case, users can enter the function, select the order of the derivative to compute, choose to display the graph, and decide to evaluate the result at a specific point. For the multivariate case, users can enter the function, choose to compute the gradient, the Hessian matrix, or both, select the desired graph, and decide to evaluate the result at a given point. Once the input submission is completed by the user, the app generates the requested symbolic and numerical results and, when enabled, the corresponding plot based on the selected mode.

Challenges

During development, we identified several practical issues that arise only after extensive testing of the application across diverse functions and settings. Most of these were not critical bugs but rather edge cases where the default behavior of the libraries did not meet reasonable user expectations for a mathematical analysis tool.

First, we encountered issues during the parsing phase, such as missing function arguments, unclosed parentheses, malformed operator sequences, and divisions by zero that can occur before evaluation. We also mitigated ambiguity and incompatibilities, such as unsupported functions in SymPy, the difference between ^ and ** for powers, interpreting "e" as either Euler's constant or a variable, and number-format differences.

A subsequent issue involved plotting functions with vertical asymptotes, such as tan(x). Matplotlib tends to connect consecutive points across discontinuities, resulting in vertical lines that misrepresent the plot. To resolve this, we detected sign changes in the data, inserted NaN values to break the line, and employed high-resolution sampling to maintain smooth curves.

Another issue arose when computing derivatives of absolute-value expressions. In particular, SimPy can return complex expressions unless the variable is explicitly assumed to be real. Since our tool targets real-valued calculus, we addressed this by temporarily enforcing real assumptions during differentiation.

We also encounter a problem with domain validation, especially for even roots or logarithm functions. We added custom domain checks to prevent users from obtaining plots or numeric evaluations that include invalid values. Specifically, when the app detects a potential domain restriction, it warns the user and suggests the correct domain, printing it on the graph as well.

Finally, in the multivariable context, we also observed numeric overflow during evaluation. Functions such as exp(x·y) grow rapidly, and even moderate input values can exceed floating-point limits, resulting in inf or NaN. To address this, we implemented threshold-based guards and improved error reporting. As a result, users receive explicit messages indicating that the evaluation is outside the representable numeric range, rather than encountering ambiguous graphs or invalid numbers.

Considerations and final remarks

As currently implemented, Gradus focuses exclusively on derivative analysis. The application presents only final results and does not display the step-by-step process. Incorporating this feature would enhance its educational value by clarifying which differentiation rules are applied at each stage.

Although domain restrictions are identified for standard cases, edge cases involving piecewise or highly composed functions may not be properly handled. Moreover, the plotting module required some modifications to accurately display functions with vertical asymptotes. Furthermore, very complex expressions may cause noticeable delays in computation.

In the short term, the primary improvements planned for future versions include the following:

- Step-by-step differentiation showing each rule applied (ex: chain rule, product rule, quotient rule) with explanations;
- A comparison mode that enables simultaneous plotting of multiple functions;
- A redesign of the interface to ensure mobile responsiveness.

A potential direction for future development involves expanding Gradus toward a comprehensive learning assistant for mathematical analysis. Beyond computational features, the application could incorporate a dedicated section to support students in understanding and constructing proofs of fundamental theorems. This module would guide users through the logical steps of a proof, offer interactive exercises in which they fill in missing justifications, and provide hints when they encounter difficulties. Such enhancements would transform Gradus from a calculator into a comprehensive study companion for theoretical mathematics.