Interactive MATLAB GUI for exploring extreme values in bivariate functions

M. Y. Tufail

Department of Mathematics, NED University of Engineering & Technology, University Road, Karachi, Pakistan, e-mail: tufail@neduet.edu.pk

S. Gul

Department of Mathematics, NED University of Engineering & Technology, University Road, Karachi, Pakistan, e-mail: sagul@neduet.edu.pk

S. A. A. Hashmi

Department of Mechanical Engineering, NED University of Engineering & Technology, University Road, Karachi, Pakistan, e-mail: alipc239@gmail.com

Received 6 May 2025; accepted 4 July 2025

This paper presents the development of a graphical user interface (GUI) in MATLAB designed to assist in the identification and classification of extreme values and saddle points of functions of two variables. Aimed at supporting mathematics and physics education for university students and instructors, the tool integrates symbolic differentiation and the second derivative test to provide an interactive learning experience. Through three representative examples, the GUI demonstrates its ability to detect multiple critical points and visually convey their nature, enhancing conceptual understanding. While the method is effective in most cases, it is limited when the discriminant D equals zero, rendering the second derivative test inconclusive. Furthermore, the current implementation supports only two-variable functions; adapting it to higher dimensions would require further development. Overall, the GUI serves as a valuable educational aid for teaching and learning bivariate calculus concepts.

Keywords: GUI; teaching mathematics and physics; extreme values; bivariate functions.

DOI: https://doi.org/10.31349/RevMexFisE.23.010212

1. Introduction

Over the past several decades, the incorporation of Graphical User Interfaces (GUIs) into educational frameworks has brought about a significant transformation in the manner by which students and educators interact with digital learning platforms and computational tools [1-8]. These interfaces have evolved into critical components of modern educational technology, promoting accessibility, interactivity, and a deeper engagement with complex subject matter. Ivan Sutherland is widely regarded as the seminal figure in this field. In 1963, he developed *Sketchpad*, the first interactive computer graphics system [9]. This pioneering innovation not only introduced the foundational concept of graphical interaction between humans and machines but also laid the groundwork for future developments in computer-aided design, simulation, and visualization across diverse scientific domains.

GUIs function as essential mediators between users–such as university students, academic instructors, and researchers–and sophisticated software environments [10-12]. They enable seamless and intuitive transitions across different applications, facilitating a more fluid and productive interaction with digital tools. In the context of higher education, GUIs serve as invaluable instruments for enriching the teaching and learning process, particularly in disciplines characterized by high levels of abstraction and complexity. Mathematics and physics, for example, often involve the exploration of theoretical models and intricate relationships that are difficult to grasp through symbolic notation alone. By offering dynamic visual representations–such as graphs, interactive plots, and real-time simulations–GUIs help translate abstract mathematical and physical concepts into tangible, comprehensible forms. This not only aids in cognitive processing but also fosters a more active and participatory learning environment.

The broader impact of computational technologies on scientific inquiry and academic instruction cannot be overstated. Within this technological landscape, GUIs have emerged as powerful pedagogical assets. Their ability to combine computational power with visual clarity enhances the presentation, exploration, and analysis of mathematical structures and physical systems. The user-friendly nature of GUI-based platforms encourages greater experimentation, promotes critical thinking, and supports conceptual understanding by enabling learners to manipulate variables, observe outcomes, and draw conclusions through direct interaction with visual data.

The importance of graphical user interfaces (GUIs) becomes particularly pronounced during natural disasters and global crises, such as the recent COVID-19 pandemic. This unprecedented health emergency accelerated the shift from traditional classroom-based instruction to remote, distance learning modalities [13-15]. Undoubtedly, this transition

emerged as both a legitimate and essential necessity for learners and educators worldwide. GUIs played a crucial role in this transition by enabling educators to deliver content in interactive formats and allowing students to engage with materials despite geographical and logistical constraints. By facilitating access to complex mathematical models and simulations, GUIs ensured that the quality of instruction could be maintained, and in some cases even enhanced, during a time of unprecedented global disruption.

In the field of physics, the concept of extreme values of functions of two variables is integral to the analysis of numerous phenomena. For instance, in thermodynamics, the optimization of heat transfer involves the determination of maximum or minimum values of heat flux and temperature distributions, subject to specific boundary conditions. Similarly, in optics, Fermat's principle posits that light travels along the path that extremizes travel time; solving this problem requires finding the minimum or stationary value of a time-dependent function of multiple variables. In structural mechanics, the minimization of potential energy allows for the determination of equilibrium shapes of elastic bodies–such as the characteristic minimal surface formed by a soap film stretched between two rings. Moreover, in analytical mechanics, the principle of least action, which underpins Lagrangian mechanics, involves minimizing an action integral. Solving such problems often entails the application of the calculus of variations to functions of two or more variables, making the identification of critical points a fundamental task in theoretical physics.

Traditionally, the analytical determination of maxima, minima, and saddle points for functions of two variables has relied on manual computation techniques. While mathematically rigorous, this process can be labor-intensive and prone to computational errors, particularly when dealing with complex or nonlinear functions. In contrast, the use of modern computational tools–such as MATLAB–has revolutionized this process. These tools offer efficient algorithms for solving optimization problems, as well as powerful visualization capabilities that allow users to graph surfaces, contour plots, and critical points in real time. Such visualizations not only enhance the accuracy of analysis but also contribute to a deeper and more intuitive understanding of the underlying mathematical structures.

Considering the points discussed above, we introduce a custom-designed MATLAB algorithm developed for this study. This tool is tailored to support the analysis of extreme values and saddle points in bivariate functions, and it includes visualization features to aid interpretation. While the primary users are expected to be university-level science instructors and students, the algorithm holds value for educators in other fields as well. Importantly, this GUI was developed in direct response to the challenges faced by students in our local academic context, particularly at institutions in Karachi, where traditional methods of teaching bivariate calculus often fall short in conveying the abstract nature of critical point analysis. By streamlining complex calculations, it reduces the potential for manual errors and effectively bridges theoretical concepts with hands-on application, thereby promoting deeper understanding. This context-specific educational tool reflects a novel contribution to teaching practice by integrating well-established mathematical methods into a user-friendly, visual platform customized for enhanced student engagement and conceptual learning.

This study is guided by the underlying hypothesis that an interactive, visually driven computational tool can enhance students' conceptual understanding of critical points and extrema in bivariate functions. The objective is not solely the development of a functional GUI, but also the creation of an educational resource tailored to address specific learning challenges faced by students–particularly within the context of undergraduate science education in Karachi. By translating abstract mathematical principles into interactive visual representations, the proposed interface aims to foster deeper comprehension and active engagement in learning environments where traditional instructional methods often fall short.

2. Theory

We provide essential definitions–for the illustration of extreme values–in this section.

Definition 1. *(Critical point) Consider a function f of two variables that is defined on $\Omega \subset \mathbb{R}^2$ as below:*

$$f : \Omega \to \mathbb{R}.$$

This function will have a critical point at (x_o, y_o), if its first-order partial derivatives (with respect to x and y) are zero or at least one of the partial derivatives is undefined. Symbolically, $f_x = \partial f/\partial x = 0$, $f_y = \partial f/\partial y = 0$ or $f_x \notin \mathbb{R}$ or $f_y \notin \mathbb{R}$.

Definition 2. *(Relative maximum) Let $f(x, y)$ be a function defined on a neighborhood of a point (x_0, y_0). We say that f has a relative maximum at (x_0, y_0) if $\exists \delta > 0$ such that $f(x, y) \leq f(x_0, y_0)$,$\forall (x, y)$ within the δ-neighborhood of (x_0, y_0). i.e., within some open disk centered at (x_0, y_0), the value of the function never exceeds $f(x_0, y_0)$.*

Definition 3. *(Relative minimum) Let $f(x, y)$ be a function defined on a neighborhood of a point (x_0, y_0). We say that f has a relative maximum at (x_0, y_0) if $\exists \delta > 0$ such that $f(x, y) \geq f(x_0, y_0)$,$\forall (x, y)$ within the δ-neighborhood of (x_0, y_0). i.e., the function is not smaller than $f(x_0, y_0)$in some neighborhood around the point.*

Definition 4. *(Saddle point) A point (x_0, y_0) is a saddle point of f if (i) it is a critical point: $\partial f/\partial x = 0, \partial f/\partial y = 0$ and (ii) it is neither maximum nor minimum, but in every neighborhood of (x_0, y_0), the function takes on both values greater than and less than $f(x_0, y_0)$.*

Figure 1 illustrates the pictorial representation of relative maximum, relative minimum and saddle point.

Definition 5. *(Second derivative test for a bivariate function) Let $f(x, y)$ be a function with continuous second partial derivatives. Suppose (x_0, y_0) is a critical point of f, i.e.,*

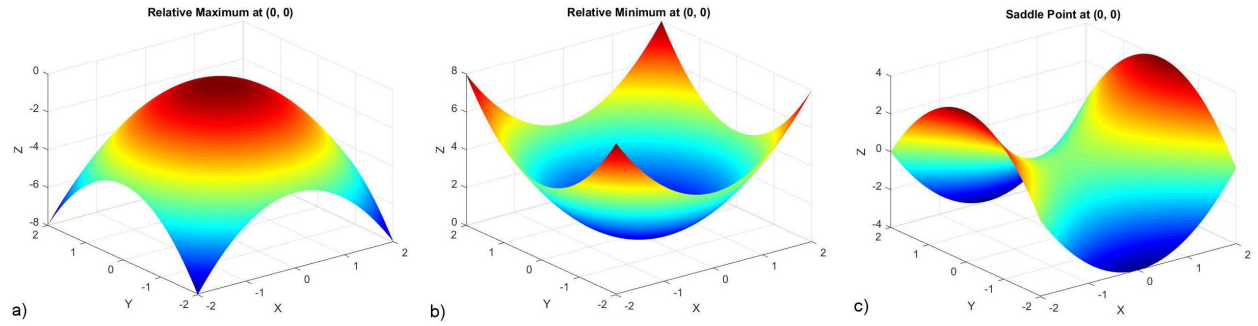

FIGURE 1. a) Depiction of relative maximum. b) Depiction of relative minimum. c) Depiction of saddle points.

$$f_x(x_0, y_0) = 0 \quad \text{and} \quad f_y(x_0, y_0) = 0.$$

Now define the discriminant D at (x_0, y_0) as:

$$D = f_{xx}(x_0, y_0) \cdot f_{yy}(x_0, y_0) - (f_{xy}(x_0, y_0))^2,$$

Decision

(i) If $D > 0$ and $f_{xx}(x_0, y_0) > 0$ then f has a relative minimum at (x_0, y_0).

(ii) If $D > 0$ and $f_{xx}(x_0, y_0) < 0$ then f has a relative maximum at (x_0, y_0).

(iii) If $D < 0$ then f has a saddle point at (x_0, y_0)
(iv) If $D = 0$ then no conclusion can be drawn.

2.1. Design principles behind the GUI

The development of the GUI was informed by constructivist learning theory, which advocates for active student engagement and knowledge construction through interactive experiences. By integrating dynamic visual and computational elements, the interface aims to facilitate a deeper conceptual understanding of extreme values in bivariate functions. From a software design perspective, the GUI was structured to be modular and user-friendly, adhering to established usability heuristics and leveraging MATLAB's capabilities for interactive visualization and computation.

3. Experiments

In this section, we present three illustrative experiments utilizing the MATLAB algorithm (refer to Listing 1 for implementation details). These experiments are designed to enhance the understanding of key concepts in bivariate calculus, specifically relative maxima, relative minima, and saddle points. It is important to note that the underlying MATLAB code is based on the application of the second derivative test (see Definition 5).

LISTING 1. MATLAB code for critical point analysis.

```
% This MatLab code will explore all possible
% options for the computation of extreme values
% of a function of two variables.
clear % Clear Workspace
clc % Clear command window
close all % Close all figures
syms x y
a=input('Enter function f(x,y) (e.g, x^2+y^2) = ',
    's'); % Ask user to input function
a =strrep(a, '^', '.^');  % Replacing ^ with .^
a =strrep(a, '*', '.*');  % Replacing * with .*
a =strrep(a, '/', './');  % Replacing / with ./
f =str2func(['@(x, y) ' a]) % Converting the
    string to function handle
fx=diff(f(x,y),x) % Derivative of function w.r.t x
fxx=diff(fx,x) % Derivative of fx w.r.t x
fy=diff(f(x,y),y) % Derivative of function w.r.t y
fyy=diff(fy,y) % Derivative of fy w.r.t y
fxy=diff(fy,x) % Derivative of fy w.r.t x
% Critical Points %
eq1=fx==0; % Assigning fx equal to zero
eq2=fy==0; % Assigning fy equal to zero
[X,Y]=solve(eq1,eq2,[x,y]); % Solve eq1 and eq2
    simultaneously
S=double(X); % Conversion of X into Numeric data
T=double(Y); % Conversion of Y into Numeric data
x_o=S(imag(S)==0); % Filter imaginary number
y_o=T(imag(T)==0); % Filter imaginary number

% Initializing
X_o=zeros(length(y_o),1);
Y_o=X_o;

% Value of critical point

if isempty(x_o) || isempty(y_o)
    disp('Critical points are complex.')
else
if length(x_o)>length(y_o)
for j1=1:length(y_o)
    X_o(j1)=x_o(j1);
    Y_o(j1)=y_o(j1);
end
elseif length(y_o)>length(x_o)
    for j2=1:length(x_o)
    Y_o(j2)=y_o(j2);
    X_o(j2)=x_o(j2);
    end
else
    X_o=x_o;
    Y_o=y_o;
end

% Plotting %
hold on
grid on
g=max(abs(X_o)); % Finding maximum of x co-
    ordinate of critical points
```

```
h=max(abs(Y_o)); % Finding maximum of y co-
    ordinate of critical points
j=max(g,h); % Finding maximum co-ordinate
j=j+15; % adding 15 to maximum co-ordinate
[X,Y]=meshgrid(-j:0.1:j,-j:0.1:j); % generate 2-D
    grids of coordinates
z=f(X,Y); % Evaluating value of function using X
    and Y values
z(imag(z)~=0)=NaN; % Assigning imaginary number to
     NaN (if exist)
contour3(X,Y,z,50,'Color', [0.8 0.8 0.8]) %
    Generate 3-D contours
xlabel('X-AXIS','FontSize',15)
ylabel('Y-AXIS','FontSize',15)
title('CONTOURS OF f(x,y)','FontSize',15)
surf(X,Y,z,'EdgeColor','none') % generate 3-D
    surface without edge color
xlabel('X-AXIS','FontSize',15)
ylabel('Y-AXIS','FontSize',15)
zlabel('Z-AXIS','FontSize',15)
title('GRAPH OF f(x,y)','FontSize',15)
view(45,30);
colorbar;
for k=1:length(X_o)
    plot3(X_o(k),Y_o(k),f(X_o(k),Y_o(k)),'o','
        MarkerSize',1,'linewidth',5,'
        markeredgecolor','k') % Plot Critical
        pints in 3-D space
end
hold off
% Discriminant (D) %


for l=1:length(X_o)
    fprintf('Critical Point = f(%5.3f,%5.3f).\n',
        X_o(l),Y_o(l))
d=fxx*fyy-fxy^2;
DD=@(x,y) d;
D(l) = double(subs(d, {x, y}, {X_o(l), Y_o(l)}));
    % First substitute value in discriminant (d)
    and then convert it into numeric data

FXX=@(s,t) fxx ;
Fxx(l)=double(subs(fxx, {x, y}, {X_o(l), Y_o(l)}))
    ; % First substitute value in fxx and then
    convert it into numeric data


if D(l)>0 && Fxx(l)>0
    fprintf('f(%5.3f,%5.3f) is Relative Minima.\n'
        ,X_o(l),Y_o(l))
    Minimum_Value=f(X_o(l),Y_o(l)) % Minimum Value
         of function
elseif D(l)<0
    fprintf('f(%5.3f,%5.3f) is Saddle Point.\n',
        X_o(l),Y_o(l)) % Print Saddle point
elseif D(l)>0 && Fxx(l)<0
     fprintf('f(%5.3f,%5.3f) is Relative Maxima.\n
         ',X_o(l),Y_o(l))
     Maximum_value=f(X_o(l),Y_o(l)) % Maximum
         value of function
else
    disp('No conclusion can be given because D=0.'
        )
end
end
end
```

3.1. Computation of relative maximum

In the first experiment, we present the implementation of a MATLAB script for the function $f(x, y) = 4xy - x^4 - y^4$. The function attains its maximum value of 2 at the points $(1, 1)$ and $(-1, -1)$, while exhibiting a saddle point at the origin $(0, 0)$. These characteristics are illustrated in Fig. 2.

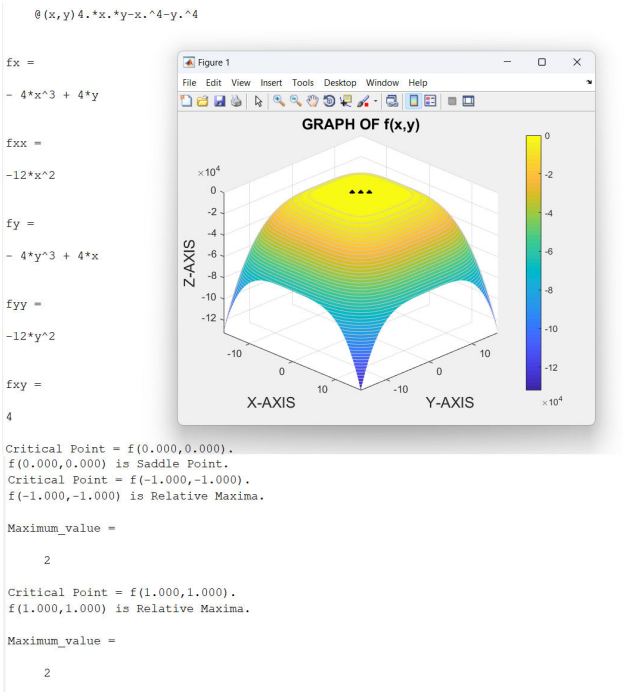

FIGURE 2. Visualization of the results obtained from the MATLAB-based implementation for the function $f(x, y) = 4xy - x^4 - y^4$. The plot highlights the presence of a saddle point at the origin and two relative maxima at $(1, 1)$ and $(-1, -1)$.

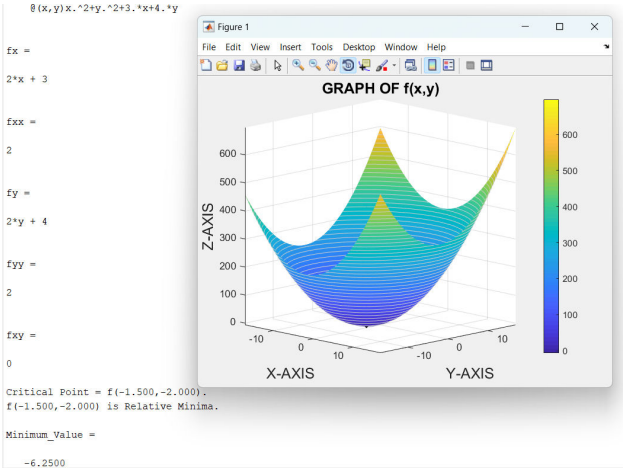

FIGURE 3. Implementation of MATLAB code for the relative minimum.

3.2. Computation of relative minimum

Figure 3 displays the outcome of applying the MATLAB algorithm to the function $f(x, y) = x^2 + y^2 + 3x + 4y$. The algorithm accurately identifies the function's minimum value of -6.25, occurring at the point $(-1.5, -2)$.

3.3. Computation of saddle point

The final experiment in our study involves the function $f(x, y) = x^2 - y^2 - 3x + 5y + 3$. The MATLAB algorithm

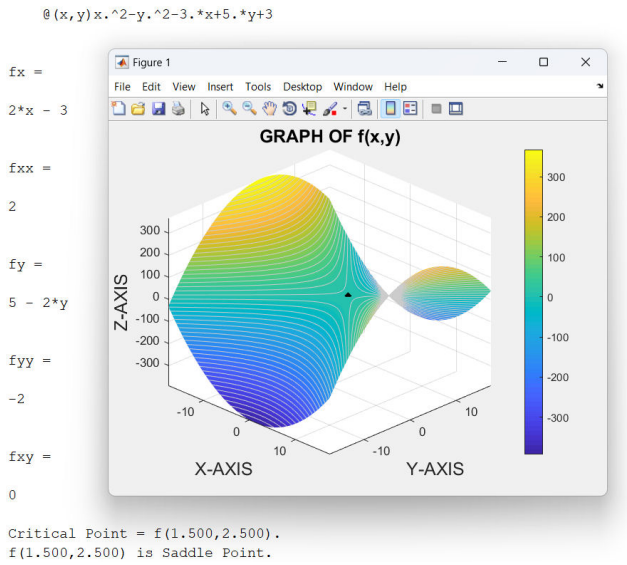

FIGURE 4. Depiction of a saddle point for a function of two variables using MATLAB.

we developed effectively identifies the saddle point located at $(1.5, 2.5)$. This result is illustrated in Fig. 4.

4. Discussion and limitation

While the proposed MATLAB-based algorithm demonstrates strong performance in identifying extreme values and saddle points for functions of two variables, there are important considerations and limitations to acknowledge.

First, the algorithm relies on the second derivative test, which involves computing the discriminant $D = f_{xx}f_{yy} - (f_{xy})^2$. This test is effective in most cases; however, it becomes inconclusive when $D = 0$. In such cases, the test fails to determine the nature of the critical point, and further analysis–either symbolic or graphical–is required to draw conclusions. This limitation is inherent to the mathematical method itself rather than the implementation, but it affects the algorithm's reliability in borderline scenarios.

Second, the current implementation is specifically tailored for functions of two variables. Extending the algorithm to handle functions of three or more variables would require significant redesign, particularly in the computation of the Hessian matrix and the classification of critical points in higher dimensions. As such, its applicability is limited to a specific class of optimization problems.

Despite these limitations, the algorithm provides a reliable and efficient tool for analyzing functions of two variables, particularly in cases where multiple critical points need to be identified and classified. With further development, the approach could be generalized to handle higher-dimensional problems and incorporate fallback strategies for inconclusive cases.

4.1. Comparison with related GUI tools

The present GUI distinguishes itself from a diverse range of earlier tools, many of which were designed for domain-specific scientific or engineering applications. For instance, ORTEP-3 for Windows by Farrugia [1], EXPGUI by Toby [2], and the CCP4 GUI suite by Potterton *et al.* [5] provided graphical interfaces to support crystallographic visualization and refinement workflows. While powerful within their respective domains, these tools were not intended for instructional use in mathematics, nor do they offer features for real-time exploration of bivariate functions.

Similarly, Gabedit, developed by Allouche [6], is a GUI tailored to computational chemistry packages, emphasizing molecular visualization and quantum chemical modeling rather than mathematical pedagogy. The work of Roßberger and von Luck [7] explored innovative physics-based GUI prototyping methods for tabletop environments, but this was aimed at interface design methodology rather than educational content delivery.

Closer in spirit to the current work are MATLAB-based educational tools, such as the trigonometry teaching GUI by Mulyawati *et al.* [10], and the engineering system model by Petrov *et al.* [11]. These contributions underscore the pedagogical potential of GUI development in technical disciplines. However, both tools target specific subfields (trigonometry and hydraulic systems, respectively), and neither integrates symbolic computation, graphical visualization, and classification of critical points into a cohesive instructional interface.

Gul and Tufail [12] developed a graphical user interface (GUI) within Excel for the visualization of conic sections, while Tufail and Gul [8] introduced an Excel-based platform for exploring the complex plane. Although these tools are designed with educational purposes in mind and promote interactive learning, their scope remains confined to specific mathematical topics and does not extend to the more general problem of understanding extrema in functions of two variables.

In contrast, the present GUI is explicitly designed for undergraduate instruction in bivariate calculus. It combines symbolic differentiation, visual representation, and second-derivative classification into a unified, user-friendly interface that supports exploratory learning and conceptual reinforcement.

4.2. Pedagogical considerations and future evaluation

While the primary emphasis of this study is on the design and functionality of the GUI, its pedagogical potential served as a central motivating factor in its development. The interface was specifically conceived to address persistent difficulties encountered by undergraduate students in comprehending the abstract concept of critical points in functions of two variables. Preliminary, anecdotal observations from informal instructional settings indicate that the GUI may con-

tribute to increase student engagement and improved conceptual understanding. Nevertheless, a systematic evaluation of its educational impact–such as a quasi-experimental study comparing learning outcomes between students utilizing the GUI and those receiving conventional instruction–remains an important avenue for future investigation. Such an evaluation could incorporate pre- and post-assessments, attitudinal surveys, and observational measures to rigorously assess the tool's effectiveness in enhancing learning.

5. Conclusion

In this study, we have presented the implementation of a MATLAB-based algorithm designed to analyze and identify extreme values and saddle points of functions of two variables. To demonstrate its effectiveness, we explored three distinct examples in Secs. 3.1, 3.2, and 3.3.

A notable strength of the proposed algorithm is its ability to detect multiple critical points within a single function, rather than limiting the search to a single solution. For instance, in Sec. 3.1, the algorithm successfully identified two relative maxima and one saddle point, showcasing its robustness and comprehensive analytical capability.

Beyond its computational effectiveness, the GUI also offers a meaningful pedagogical advancement. By providing an interactive visual environment tailored for students struggling with abstract bivariate concepts, this tool serves as a bridge between symbolic mathematics and intuitive understanding. Its use in classroom and remote settings demonstrates its adaptability and potential to enrich physics and mathematics instruction in resource-constrained and high-variability educational contexts.

While this study primarily focuses on the computational development and performance of the GUI, its broader significance lies in its potential to enhance students' conceptual understanding of critical points in bivariate calculus. By combining interactive visualization with mathematical rigor, the tool aims to bridge the gap between abstract theory and intuitive learning. Future work will involve formal pedagogical assessment to empirically evaluate the GUI's impact on student learning outcomes and guide further refinement of the interface.

Supplementary video demonstrations of the algorithm's implementation and performance can be accessed through the following links:

https://drive.google.com/file/d/11FUEbqmL39R85izUnEVKbr4UW571k5hR/view?usp=drive_link

1. L. J. Farrugia, ORTEP-3 for Windows-a version of ORTEPIII with a Graphical User Interface (GUI), *J. Appl. Cryst.* **30** (1997) 565, https://doi.org/10.1107/S0021889897003117
2. B. H. Toby, EXPGUI, a graphical user interface for GSAS, *J. Appl. Cryst.* **34** (2001) 210, https://doi.org/10.1107/S0021889801002242
3. B. Buchberger, Mathematica: A system for doing mathematics by computer?, In A. Miola, ed., Design and Implementation of Symbolic Computation Systems (Springer Berlin Heidelberg, Berlin, Heidelberg, 1993) pp. 1-1, https://doi.org/10.1007/BFb0013163
4. A. Oulasvirta *et al*., Combinatorial optimization of graphical user interface designs, *Proc. IEEE.* **108** (2020) 434, https://doi.org/10.1109/JPROC.2020.2969687
5. E. Potterton *et al*., A graphical user interface to the CCP4 program suite, *Acta Cryst. D: Struct. Biol.* **59** (2003) 1131, https://doi.org/10.1107/s0907444903008126
6. A.-R. Allouche, Gabedit, A graphical user interface for computational chemistry softwares, *J. Comput. Chem.* **32** (2011) 174, https://doi.org/10.1002/jcc.21600
7. P. Roßberger and K. von Luck, Iterative design of tabletop GUIs using physics simulation, In H. Wandke, S. Kain, and D. Struve, eds., Mensch & Computer 2009 (Oldenbourg Verlag, 2009) pp. 203-212, https://dblp.org/rec/conf/mc/RossbergerL09
8. M. Y. Tufail and S. Gul, GUI of Complex plane on Excel spreadsheets, *Rev. Mex. Fis. E* **22** (2025) 010208, https://doi.org/10.31349/RevMexFisE.22.010208
9. I. E. Sutherland, Sketch pad a man-machine graphical communication system, In Proceedings of the SHARE Design Automation Workshop, DAC '64 (Association for Computing Machinery, New York, NY, USA, 1964) p. 6.329-6.346, https://doi.org/10.1177/0037549764002005.
10. C. Mulyawati *et al*., Teaching media development of mathematic in the materials trigonometry sum and two angles difference by using GUI Matlab, *Jurnal Natural* **17** (2017) 69, https://doi.org/10.1145/800265.810742
11. O. Petrov *et al*., Mathematical Modeling of the Operating Process in LS Hydraulic Drive Using MatLab GUI Tools, In V. Ivanov, et al., eds., Advances in Design, Simulation and Manufacturing III (Springer International Publishing, Cham, 2020) pp. 52-62, https://doi.org/10.1007/978-3-030-50491-5_6
12. S. Gul and M. Y. Tufail, GUI for conic sections: parabola, ellipse and hyperbola, *Rev. Mex. Fis. E* **21** (2024) 010203, https://doi.org/10.31349/RevMexFisE.21.010203
13. M. Ciotti *et al*., The COVID-19 pandemic, *Crit. Rev. Clin. Lab. Sci.* **57** (2020) 365, https://doi.org/10.1080/10408363.2020.1783198
14. S. J. Daniel, Education and the COVID-19 pandemic, *Prospects* **49** (2020) 91, https://doi.org/10.1007/s11125-020-09464-3
15. S. Pokhrel and R. Chhetri, A literature review on impact of COVID-19 pandemic on teaching and learning, *High Educ. Future* **8** (2021) 133, https://doi.org/10.1177/2347631120983481