

AI-Driven Python Code Generation for Solving Specific Mathematics Problems

Hongwei.wang
hongwei.wang22@student.xjtlu.edu.cn

March 11, 2025

Abstract

In daily learning, we encounter various challenges across different areas of mathematics. However, with the assistance of Artificial Intelligence (AI) and computers as tools, we can leverage artificial intelligence and coding to facilitate and enhance our mathematical learning in everyday life. For certain specific problems, AI-generated assistance in program code can help us implement corresponding mathematical concepts through Python, or simplify mathematical computations. This process will lead to a deeper understanding of our mathematical learning while also saving us time.

Keywords: Artificial Intelligence, AI-Generated Code, Mathematical Computations

1 Introduction

This report presents the assistance code and program results corresponding to the common difficulties encountered in several major courses during the third year of study.

2 Vector Field Theory Visualization(MTH207)

In vector field theory, we can utilize AI-generated code to provide visualization tools for the gradient, curl, and divergence, and assist us in their calculation.

2.1 Scalar Fields Visualization

We can ask AI to provide a Python code for visualizing a scalar field. Then AI provides an interactive 3D scalar field visualization tool. Users can input a 3D scalar field equation (e.g., $f(x,y,z)$), and the program will automatically compute and generate a visualization of the scalar field, displaying the isosurfaces of the field. Additionally, users can select any coordinate point, and the program will calculate and display the gradient vector at that point in real-time. This helps users better understand the local variations and gradient information of the scalar field. This visualization method is suitable for exploring and analyzing scalar fields and their derivatives in three-dimensional space.

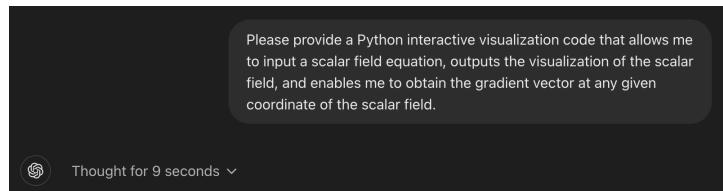
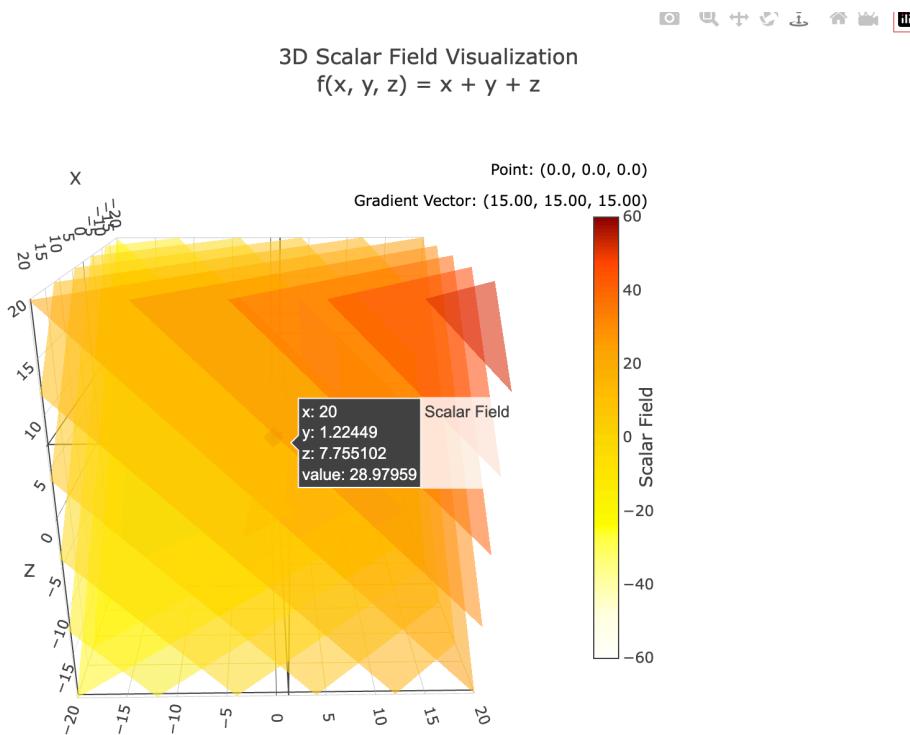


Figure 1: Input Requirements

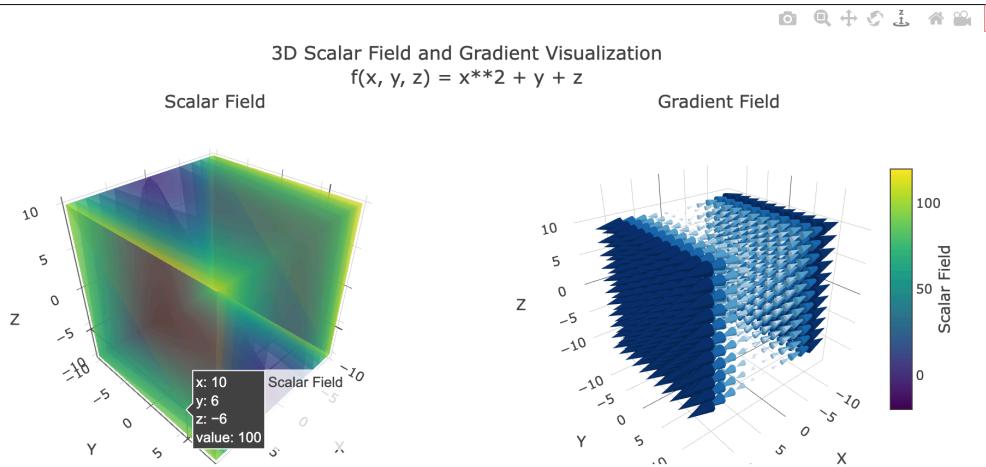
Figure 2: Visualization of Scalar Field: $x + y + z$

2.2 Gradient Fields Visualization

In the context of scalar field code, we can modify the code such that, in addition to outputting the scalar field visualization, it also outputs the corresponding vector field. We can reduce the time complexity of the code, enhance its aesthetics, or implement additional features through our own programming skills or by seeking assistance from AI.

```

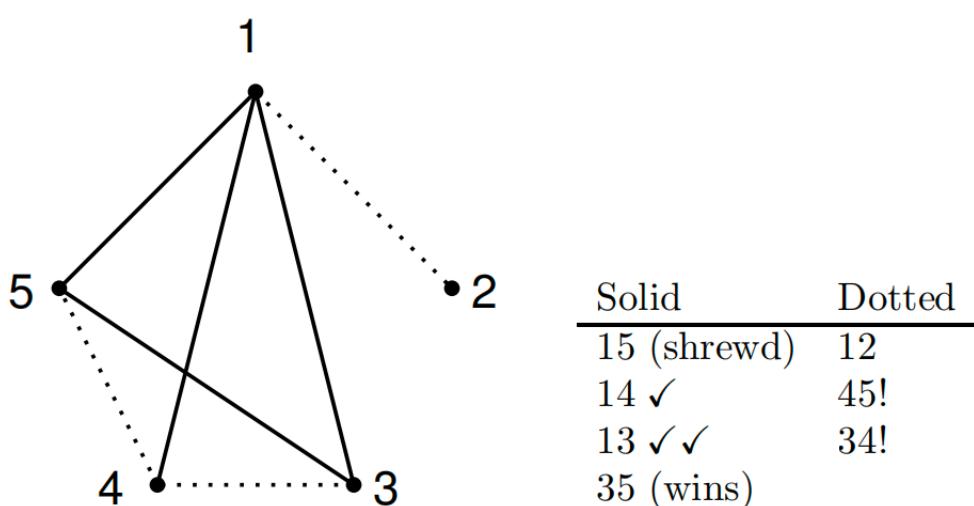
import sys
# Define
>>> def get_scalar_function():
>>>     pass
>>> def parse_function(func_str):
>>>     pass
>>> def create_grid(bounds=(-10, 10), step=1):
>>>     pass
>>> def compute_scalar_field(func_lambda, X, Y, Z):
>>>     pass
>>> def compute_gradient_sympy(func_sympy, X, Y, Z):
>>>     pass
>>> def visualize(X, Y, Z, F, grad_x, grad_y, grad_z, func_str):
>>>     fig = make_subplots(1, 2)
>>>     fig.add_trace(go.Surface(x=X, y=Y, z=Z, surface_count=8, opacity=0.6,
>>>                            showscale=True, colorbar=dict(
>>>                                title='Scalar Field',
>>>                                titlefont='right',
>>>                                thickness=20,
>>>                                len=0.75)))
>>>     fig.add_trace(go.Scatter3d(x=X, y=Y, z=Z, mode='lines',
>>>                               line=dict(color='red', width=2)))
>>>     fig.update_layout(title=func_str)
>>>     fig.show()
>>> def main():
>>>     pass
    
```

Figure 3: Scalar Field $x^2 + y + z$ and Its Gradient Field

3 Finding Optimal Strategy In Combinatorial Game Theory (MTH229)

3.1 Graph Achievement Game

Graph Achievement Game refers to the use of graph theory in strategic games, where a graph represents the possible states and transitions in the game. In this context, nodes represent game states, and edges represent actions or moves that change the state. The "achievement" is the goal that players aim to reach by navigating through the graph. In the example below, Player 1 connects two points with a solid line, and then Player 2 connects two unconnected points with a dashed line. The first player to form a triangle using their own connected lines wins the game.

Figure 4: Graph Achievement Game for K_3 in K_5 , player one (Solid) has a forced win strategy. Erickson, M. (2009). *Pearls of Discrete Mathematics*. p166, CRC Press.

3.2 Graph Avoidance Game

Graph Avoidance is the inverse rule of Graph Achievement. In the example above, the first player to form a triangle using their own connected lines loses the game. We might instinctively think that because Player 1 has a winning strategy in the previous example, where they are the first to construct a triangle, Player 1 would inevitably lose in this game, meaning Player 2 has a winning strategy. However, this is not the case, as the reversal of the rules necessitates a strategic adjustment.

We ask the AI to provide a code that allows us, as Player 1, to play against a computer acting as Player 2. If we are able to win, it would imply that Player 2 does not have a guaranteed winning strategy, thereby validating our hypothesis.

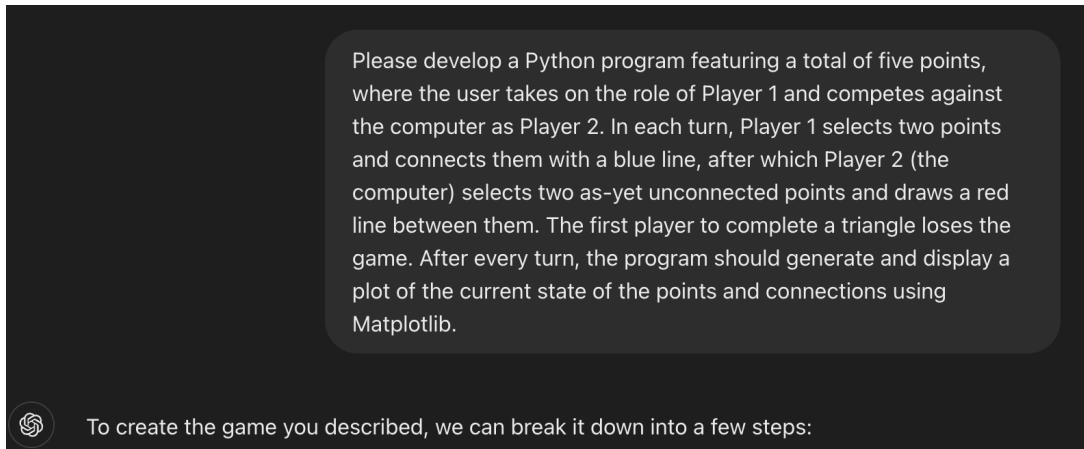


Figure 5: Input Requirements

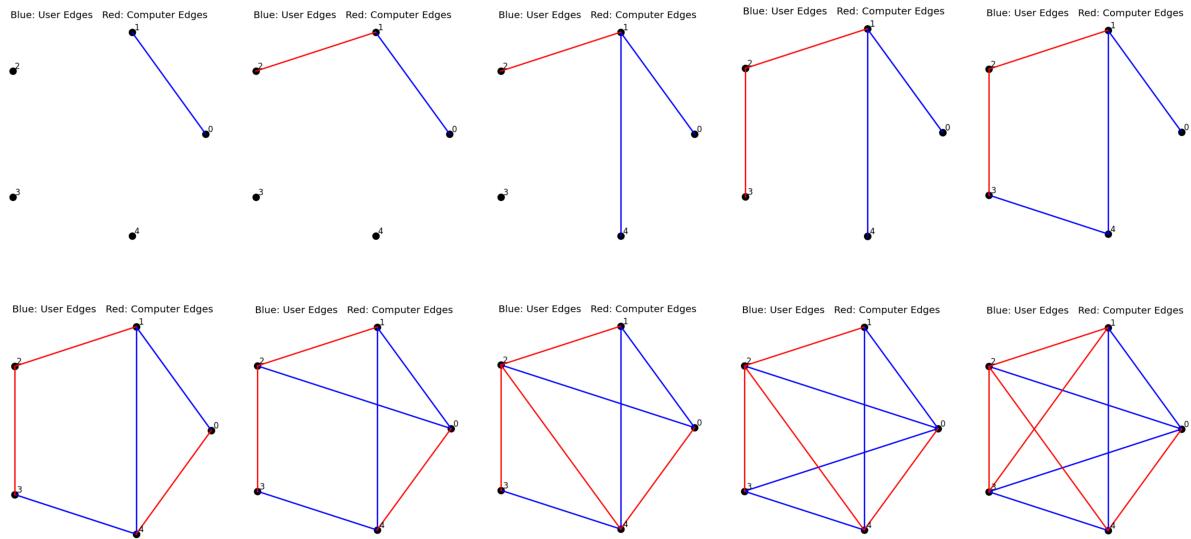


Figure 6: In a single round of the game, under the player's configuration, the computer is forced to draw a triangle, leading to failure.

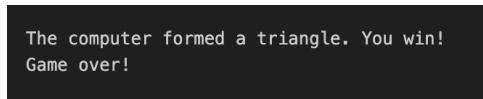


Figure 7: AI lose in this game

You win Player2 (AI) as Player 1, can it be concluded that Player 2 does not have a winning strategy?

NO!!!!!!

In this game, what strategy does computer follows?

In the current implementation of the game where the user (Player 1) and the computer (Player 2) take turns drawing edges between points on a plane, the computer follows a **random strategy**. Here's how the current implementation works and the strategy used by the computer:

Current Strategy (Random Move):

- Player 1 (Human):** The human player selects two points (which are not yet connected) and connects them with a blue line.
- Player 2 (Computer):** The computer randomly selects two unconnected points and connects them with a red line.

Figure 8: AI always pick two vertices randomly

3.3 Machine Learning In Game Theory

If Player 2 has a winning strategy, we can find it using machine learning techniques. The computer can play against itself randomly for several rounds, and then, based on the outcomes of all the games, we check whether Player 2 can always win regardless of any move made by Player 1. If Player 2 has a guaranteed winning strategy, this approach will train an AI that only makes one mistake, and subsequently, it will be undefeatable.

Please provide a Python code program based on an existing program, incorporating machine learning, where the computer plays against itself for 10,000 rounds to find the optimal strategy. Then, using the optimal strategy it has learned, the computer will play as Player 2 against a human.

Thought about computer learning optimal strategy for a couple of seconds ↴

Below is an example of how you might modify the given game code to include a rudimentary reinforcement learning component. In this scenario:

1. We use Q-Learning to train the computer to find a near-optimal strategy by having the computer play against itself for 10,000 training rounds.
2. We store Q-values in a dictionary keyed by states and actions.
3. A "state" is represented by which edges have been taken so far, along with the current player's turn. Since there are only 10 possible edges among 5 points, a game state can be represented by a 10-bit binary pattern (each bit representing whether a particular edge is chosen or not) plus the current player.
4. After training, we run the game in "human vs. computer" mode, where the computer (Player 2) uses the learned Q-values to select moves that maximize its winning chances.

Figure 9: We use machine learning to help AI find the winning strategy for player 2

After training, the AI has made significant progress in its strategy. We found that if neither player has a winning strategy, and both players follow their optimal strategies

during the game, the result will inevitably be a draw.

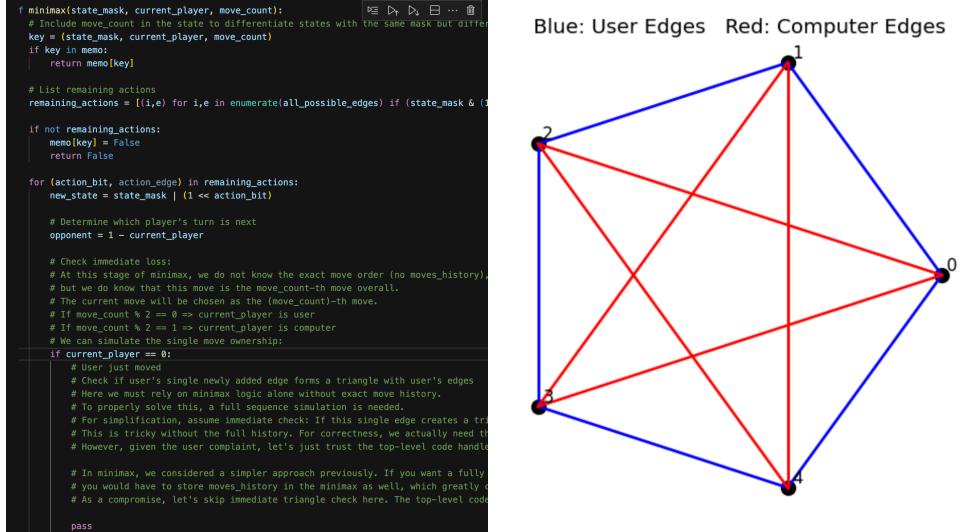


Figure 10: The best strategy for two players is to draw

4 Python’s Proofs for Complex Functions(MTH219)

In the theory of complex functions, understanding is challenging for humans because complex functions inherently exist in four dimensions. We aim to ask AI to provide an intuitive Python program demonstration for a specific complex function to enhance our understanding.

4.1 Domain Coloring of Complex Functions

Complex functions exists in four-dimensional space therotically, but we can perceive the shape of complex functions through the modulus of complex numbers, revealing the projection of complex functions in lower dimensions. We use colors to scalarize and visualize the output.

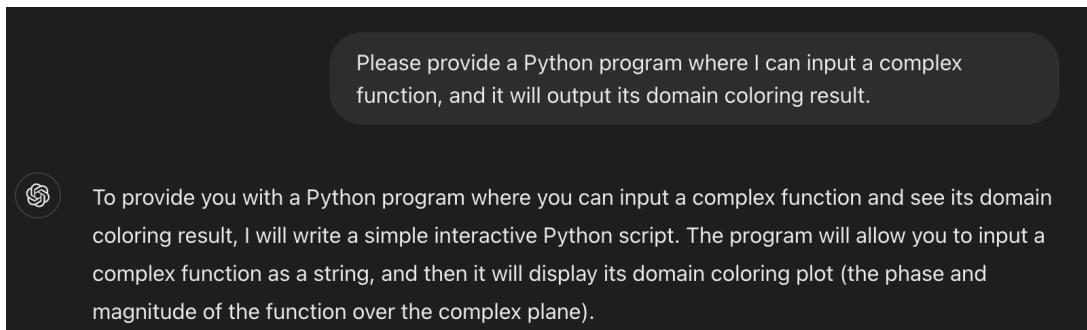


Figure 11: Input Requirements

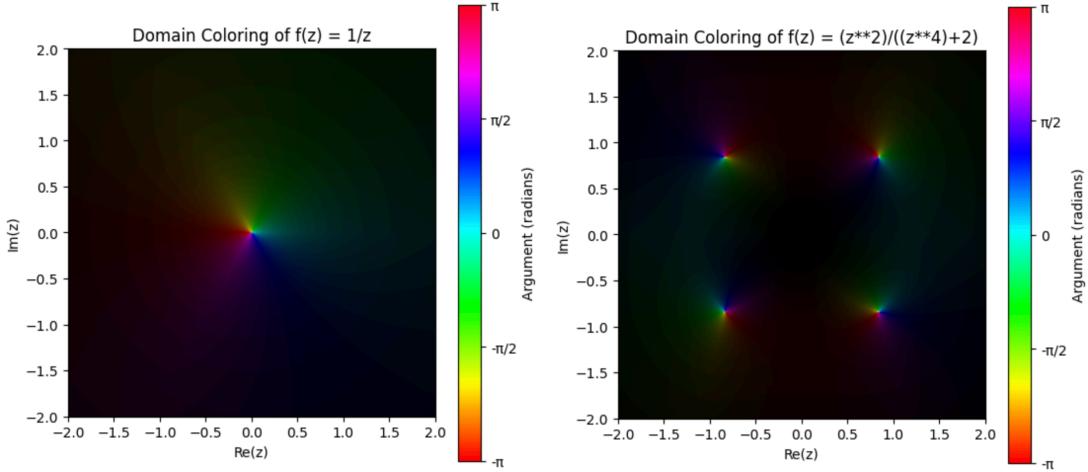


Figure 12: Domain Coloring for $f(z) = \frac{1}{z}$ and $f(z) = \frac{z^2}{z^4+2}$, in such way we can easily find out the locations of their singularities. In this visualization, the hue of the color represents the phase (argument) of the complex function, while the brightness (value) represents the magnitude (modulus) of the function.

4.2 A Coloring Picture for Cauchy-Goursat Theorem

Let's consider a rather bold idea. Since the integral of an analytic function over any closed path is zero, it follows that if we use color mapping to represent the complex function, the average color along any closed path is fixed, and corresponds to the color mapping of zero!

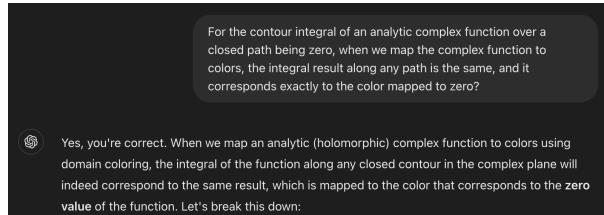


Figure 13: Propose a conjecture

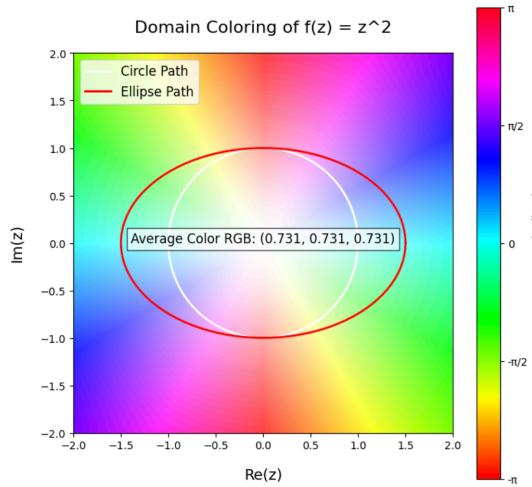


Figure 14: The average RGB value is the same for different paths.

For different path selections, the average RGB value along the path is constant and corresponds to the color mapped to zero. AI provides us with a code that allows us to adjust the selected path and compute the average RGB value, enabling us to continually test and verify this fact.

5 Conclusion

As students of mathematics, we are often required to deal with abstract and complex mathematical concepts. Amidst these intricate ideas and numerous computations, computers offer us valuable shortcuts and opportunities. We do not need highly specialized programming knowledge; instead, we can simply request the AI to generate the necessary programs and then make minor adjustments. Through this technical means, we can deepen our understanding of mathematical knowledge and assist in the proof of conjectures. With the increasing prevalence of AI in our daily lives, it is essential to use it judiciously, to pose thoughtful questions, and to leverage AI to enhance our mathematical learning.

If you wish to obtain the code script used to generate the output in this text, please contact [Hongwei.wang22@student.xjtu.edu.cn].

Thanks for reading : >