Graph-It: Design and implementation of a Graph Theory toolbox and visualization application

Diane Y. Du, Landon Fox, Jimson Huang, Logan Leavitt,
Mohit Kumbhat, and Frederick C. Harris, Jr.

University of Nevada, Reno, NV USA
mkumbhat@unr.edu
fred.harris@cse.unr.edu

Abstract

Graph-It is a Graph Theory toolbox and graph visualization desktop application for Windows, macOS, and Linux for students, teachers, and researchers, since creating graphs by hand and running graph algorithms repeatedly on the graphs can be time consuming. Users can create graphs, modify graphs, save graphs, and run various graph algorithms on the graphs and view the results. The application is built using Unity, utilizing C# for both the front end and back end. Graph-It aims to provide users an easy-to-use and efficient application for creating and running various algorithms on multiple graphs.

Keywords: C#, graph algorithms, Graph Theory, Unity, visualization

1 Introduction

Creating graphs by hand and running graph algorithms repeatedly on graphs can be time consuming. Graph-It is a Graph Theory toolbox and graph visualization application that provides users an environment to construct their own graphs and run various graph algorithms. The intended audience for Graph-It includes those in education and academic positions. Graph Theory is a mathematical study of graphs, which are structures to model relations between objects, and often requires visualization. For any student studying Graph Theory or its applications, the ability to easily create and manipulate graphs, run and display algorithms, and to store graphs to files, would provide a beneficial learning environment. Moreover, professors would also benefit from an interactive environment that can illustrate concepts live in lecture. As for researchers in fields that utilize Graph Theory, large graphs can be imported into the application and computations can be exported so that conjectures and hypotheses can be tested.

Graph-It provides the users various methods of creating graphs without requiring coding knowledge. Similarly, Graph-It allows users to save graphs as PNG images and export as CSV files. Additionally, Graph-It displays graph information, such as order and size, to the user and updates as the user modifies the graph. Graph-It provides various graph algorithms for users to run on the graph, including Dijkstra's algorithm, Depth First Search, Max Matching, and more. After algorithms are completed, users can view the results of the algorithms. Furthermore, users

can view each step of the algorithm, allowing users to learn the about the algorithm and to verify and see how the algorithm obtains its result.

The rest of the paper is structured as follows: Section 2 provides a review of similar Graph Theory software. Section 3 provides the engineering specification and design of the project, including the requirements, use cases, and traceability. Section 4 provides the implementation details, along with technology used. Conclusion and Future Work are discussed in Section 5.

2 Related Work

One application that can be used for graph visualization is Tikzcd-Editor, which has a simple user interface and clean graph visualization [5]. A second related application is Gephi, which provides users the means to visualize large and complex graphs, with applications in biology [1]. Lastly, MATLAB [3], Mathematica [9], and SageMath [7] are programs that allow users to create graphs and run graph algorithms through code.

Tikzcd-Editor: Tikzcd-Editor is a web application allowing users to create and visualize commutative diagrams and output them as LaTeX markup compatible with the Tikz-cd package [5]. Tikzcd-Editor allows various customization options for their diagrams that are useful for mathematicians. The simplicity and ease of diagram creation as well as LaTeX integration were features desired for Graph-It. As a visualization and LaTeX markup tool, Tikzcd-Editor does not provide any interface for graph theoretic computations, a major focus of Graph-It.

Gephi: Gephi is a program which also aims to visualize graphs [1]. Some features of Gephi, which the team has aimed to replicate in Graph-It, are the vast array of customization options, impressive graphics and animations, and support for large graphs (up to thousands of edges). However, Gephi focuses more narrowly on the biological applications of graphs rather than Graph Theory itself which is Graph-It's intended focus. Thus, Graph-It differs from Gephi as it includes more general-purpose graph algorithms.

MATLAB: MATLAB is an application equipped with a multi-paradigm programming language and visualization tools aimed at scientists and engineers [3]. Specifically, MATLAB provides support for modern methods in control systems, machine learning, signal processing, and much more; as a result, MATLAB provides various libraries for graph and network theory. For example, MATLAB allows users to create, modify, and visualize graphs in addition to performing common and custom graph computations using its programming language. A major goal of Graph-It was to provide computational tools similar to MATLAB, yet in a manner that is interactive and intuitive by a larger audience that does not require programming experience.

Mathematica: Mathematica is a technical computing program paired with the Wolfram programming language with libraries that allow applications in mathematics, symbolic computation, data science, and more [9]. Moreover, Mathematica provides vast computations and visualizations for Graph Theory and related subjects that satisfies most users' needs, yet still requires programming knowledge, limiting its audience.

SageMath: SageMath is a computer algebra system resembling the Python programming language that features support for many branches of mathematics such as algebra, geometry, and Graph Theory [7]. SageMath has built-in libraries that are uniquely aimed as mathematicians;

furthermore, SageMath well equipped to provide most computations a student or researcher may require. However, SageMath also limits its audience with programming experience and lacks tools to aid student understanding.

3 Software Design

3.1 Requirements Specification

The following subsections list out the functional and non-functional requirements used during the development of this project [6]. Per Sommerville, functional requirements outlines the specific services the program should provide, or features to be included. Non-functional requirements apply to the project as a whole and impose various constraints. Feedback from educators in computer science and mathematics were used to develop these requirements.

3.1.1 Functional Requirements

1. Graph-It shall allow a user to create a graph.
2. Graph-It shall visually display a graph.
3. Graph-It shall allow the user to create vertices.
4. Graph-It shall allow the user to create edges to connect between the vertices.
5. Graph-It shall allow the user to select collections of edges and vertices.
6. Graph-It shall allow the user to move vertices around.
7. Graph-It shall allow the user to remove vertices and edges.
8. Graph-It shall display a toolbar for the user with options the user may select to manipulate an existing graph.
9. Graph-It shall allow the user to import a graph from a file.
10. Graph-It shall allow the user to save a graph to a file.
11. Graph-It shall display mathematical information about a graph such as number of vertices, number of edges, and basic graph classification.
12. Graph-It shall allow the user to run an algorithm to construct a minimum spanning tree on a graph.
13. Graph-It shall allow the user to run pathfinding algorithms on graphs.
14. Graph-It shall run an algorithm to check if a graph is acyclic.
15. Graph-It shall run an algorithm to check if a graph is bipartite.
16. Graph-It shall allow the user to add labels to vertices and edges.
17. Graph-It shall allow the user to zoom in and out on a graph.
18. Graph-It shall allow the user to save pictures of a graph.
19. Graph-It shall allow the user to pan around a graph.
20. Graph-It shall allow the user to modify the direction of a directed edge.
21. Graph-It shall allow the user to enter LaTeX labels.
22. Graph-It shall allow the user to select a subgraph.

23. Graph-It shall allow users to create graphs from presets.
24. Graph-It shall allow the user to change the visualization style of vertices.
25. Graph-It shall run an algorithm to check if a graph is planar.
26. Graph-It shall allow the user to run an algorithm to construct a maximal matching.
27. Graph-It shall allow a user to be able to start running an algorithm on a graph, and stop running an algorithm at any point.
28. Graph-It shall run estimates for computationally intensive algorithms.
29. Graph-It shall allow the user to undo and redo operations.
30. Graph-It shall allow the user to create a new graph from a selection.
31. Graph-It shall run algorithms in separate background threads.
32. Graph-It shall allow algorithms to share results to increase computation speed.
33. Graph-It shall allow the user to enumerate through algorithms.
34. Graph-It shall allow the user to use custom vertex sprites.
35. Graph-It shall allow the user to manipulate the user interface.
36. Graph-It shall allow the user to copy and paste between different graphs.
37. Graph-It shall incorporate a colorblind mode and a screen reader.
38. Graph-It shall allow users to import custom algorithms that can be run and displayed on the graph.

3.1.2 Non-Functional Requirements

1. Graph-It will run on Windows, macOS, and Linux.
2. Graph-It will be implemented in Unity Engine.
3. Graph-It will have an intuitive user interface for graph creation/manipulation.
4. Graph-It will display graphs in a visually appealing way.
5. Graph-It will include algorithms useful for graph theorists.
6. Graph-It will support industry standard file formats for graph input and output.
7. Graph-It will efficiently run graph algorithms.
8. Graph-It will support large graphs without unreasonable performance degradation.
9. Graph-It will be able to be used to educate students on graph algorithms.
10. Graph-It will run on the browser as a web app.
11. Graph-It shall be optimized for minimum RAM usage.
12. Graph-It shall include some fun and visually appealing graph algorithms for demonstration purposes.

3.2 Use Cases

Use cases show the interactions between actors and the system [6]. Figure 1 is an illustrated depiction of these interactions, consisting of various actors connected to the actions each actor is allowed to perform. The rest of this section provides details on each use case.

1. **StartEmpty**: Upon the start of the program, the user can choose to create a null graph, and add vertices and edges to it manually.
2. **StartLoadFromTemplate**: Upon the start of the program, the user can choose to load an existing template of commonly used graphs.
3. **StartLoadFromFile**: Upon the start of the program, the user can choose to load a saved graph from a supported file format from the disk.
4. **CustomizeGraphLayout**: The user can drag vertices around the graph to organize the graph as the user sees fit. When dragging a vertex, no other vertices will be affected. Two vertices cannot occupy the exact same location.
5. **ModifyGraph**: The user can add vertices and edges to a graph, as well delete any vertices and edges from the graph. The graph should be responsive to any changes to the structure of the graph and the mathematical information should update in real time.
6. **CalculateGraphInfo**: The program will calculate mathematical information about the graph in the background in real time so that it is visible to the user when needed.

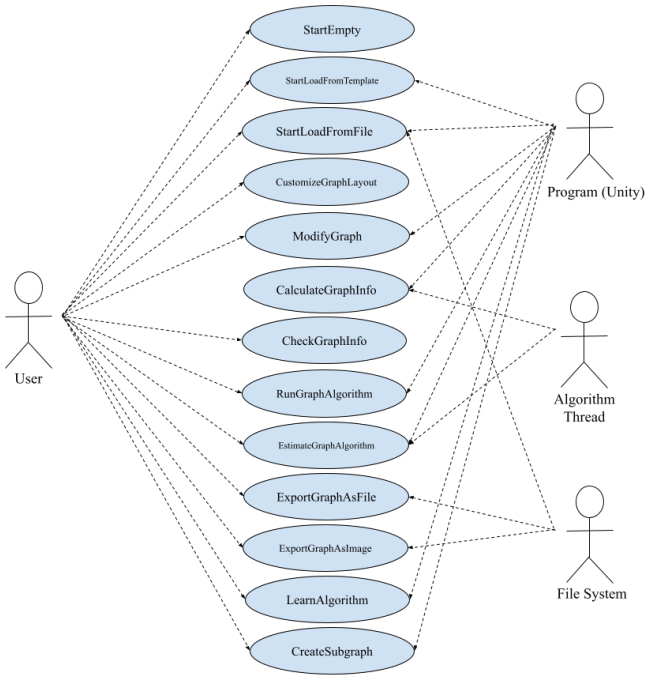

Figure 1: Use case diagram displaying use cases for Graph-It and their interactions with different actors.

7. **CheckGraphInfo**: The user can use the selection panel on the left side of the UI to check certain mathematical information about the graph.

8. **RunGraphAlgorithm**: The user can request certain graph algorithms to be run from the selection panel. If needed, the program will ask the user to specify initial information for the algorithm. The program will run the algorithm in the background and display the result(s). The user can choose to cancel the graph algorithm.

9. **EstimateGraphAlgorithm**: The user can request certain graph algorithms to be run from the selection panel. If the algorithm is computationally intensive or the graph size exceeds a certain threshold, the program will first run an estimation algorithm and display the result of the estimate. The actual algorithm is run in the background and will replace the result of the estimate once it is complete.

10. **ExportGraphAsFile**: The user can choose to export the graph into the file system as a supported file type, which allows the graph to be reimported to the program.

11. **ExportGraphAsImage**: The user can choose to export the graph into the file system as an image.

12. **LearnAlgorithm**: The user can choose to iterate through a supported graph algorithm to learn about its functionality. The program will step through the algorithm and display the result during each iteration.

13. **CreateSubgraph**: The user can choose to select a certain number of vertices in the graph, then press a button to create a subgraph. Then a new graph containing the selected vertices will be opened in another tab and exist separately from the first graph.

3.3 Traceability Matrix

Traceability describes the relationships between requirements and system design [6]. A traceability matrix, as seen in Figure 2 is a graphical record of such relationships. If a functional requirement is needed to implement a use case, their intersection on the matrix is colored black. Traceability analysis helps establish a hierarchy of priority for the functional requirements and helps keep development to an organized fashion.

4 Implementation

Graph-It is implemented following the Model-View-Controller (MVC) architectural pattern. The main technology used to implement Graph-It are the Unity Engine, Unity Standalone File Browser, and the CodeCogs Equation Editor. We have tested the current version of Graph-It under both Windows, Linux, and macOS. A screenshot can be seen in Figure 3.

4.1 Architectural Pattern

The high-level architectural pattern of Graph-It is a MVC model. In a MVC model, system components are separated into three parts. The model components manage the data, the view components manages how the data is presented to the user, and the controller components manages user interactions [6]. Figure 4 shows the high-level MCV architectural pattern that exists within the application.

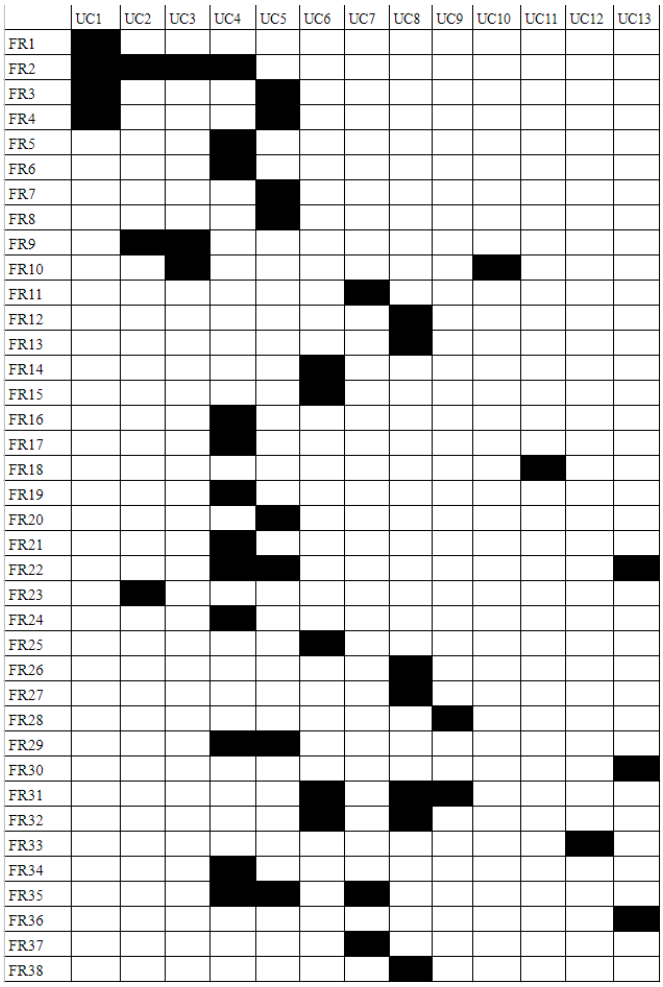

Figure 2: Traceability matrix; FR refers to functional requirement and UC refers to use case.

Model: The Model component consists of the data structures and data for the vertex, edge, graph information, and information needed to perform algorithms. The Model component provides the data needed by the View component to display the graphs and algorithm computations for the View component to display the results.

View: The View component consists of graph display algorithms, means of processing graph information, graph algorithms, and user menus. The View component has processes to determine the graph qualities, such as the type of graph, to display on the graph information panel. Graph algorithms in the View component consist of processes to interpret the data obtained from the graph algorithms and change that data in a viewable form. User menus consist of the processes and information required to display user menus. The View component sends the data and necessary displays to present to the user. The View component also sends user events to the Controller to process and perform the necessary actions. Additionally, the View component requests and retrieves the data needed to display the user requests from the Model component.

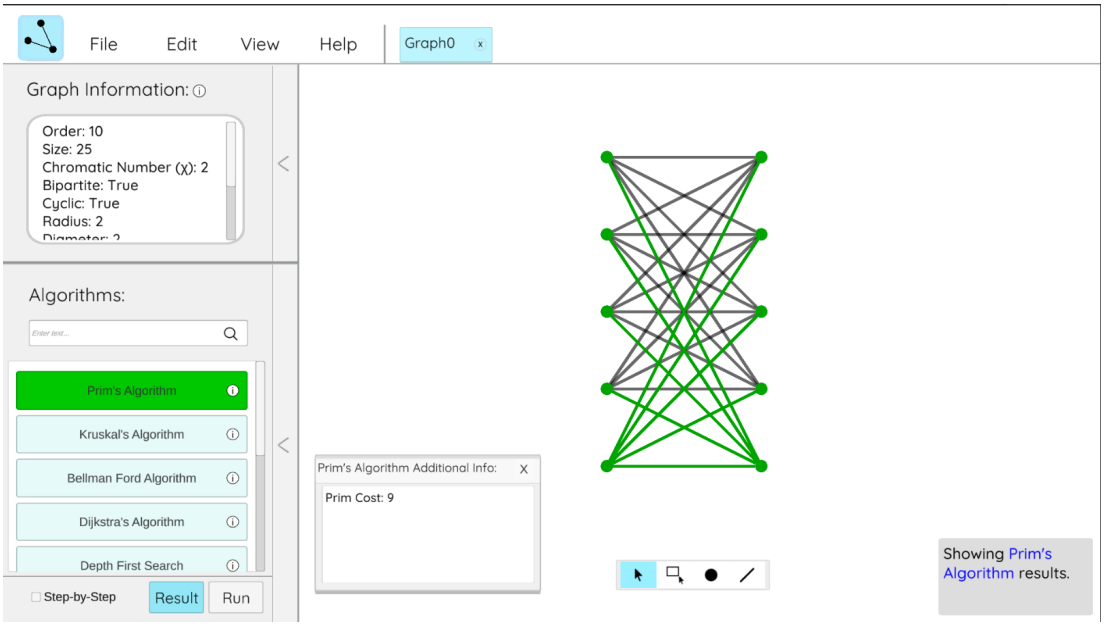

Figure 3: A screenshot of Graph-It

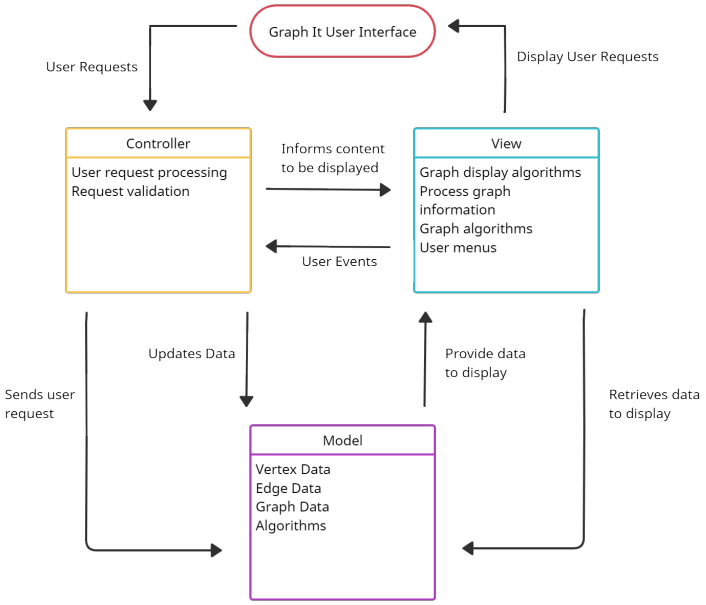

Figure 4: Architectural Pattern of Graph-It with the Model-View-Controller pattern to illustrate the system-level interactions between the Model, View, and Controller components.

Controller: The Controller component processes the user interaction, termed as user requests, and validates whether the interaction yields a valid operation or not. Then, the Controller component passes information to the View component to let the View component determine what to display to the user. For example, if the user attempts to perform an invalid interaction, the Controller component determines that the interaction is invalid and sends the information to the View component, and the View component can determine the appropriate error message to display to the user. The Controller component also passes the user interaction requests to the Model component for the data for data processing. Additionally, if the user interaction causes changes to the data, such as when a new vertex is created, the appropriate data is updated in the Model component.

4.2 Technology Used

Unity Engine: Graph-It is implemented in Unity (Version 2020.3.22f1), a popular game engine. As a game engine, Unity includes sophisticated and efficient tools for visualization, which is a core element of the program [8]. Additionally, Unity programs are cross-platform and can be compiled for Windows, macOS, and Linux. Unity uses the C# programming language, developed by Microsoft. C# is an object-oriented language and has similar syntax to Java, allowing for multiple components of the program to be developed simultaneously.

Unity Standalone File Browser: Unity Standalone File Browser is an external plugin for Unity which provides a wrapper to access the native operating system file browsers for Windows, macOS, and Linux [2]. This plugin was used to implement the file import and file/image export functionalities, allowing the user to use their OS's native file browser to select a location for saving or a file for importing.

CodeCogs Equation Editor: CodeCogs is an website which renders LaTeX equations when given its LaTeX markup [4]. CodeCogs provides a free URL-based API for creating rendered LaTeX images, which was used for the LaTeX labelling feature of Graph-It. LaTeX was chosen because of the mathematical purpose of the project.

5 Conclusions and Future Work

5.1 Conclusions

Graph-It was created for the purpose of visualizing and performing computations on graphs in a convenient and intuitive environment for students and academics without requiring programming knowledge. MATLAB, Mathematica, and SageMath, although having many graph algorithms, require programming knowledge to utilize. Gephi can accommodate large graphs and has great visualization, which Graph-It will aim to accommodate in future works. Tikzcd-Editor has many graph customization methods for graph visualization, which Graph-It also seeks to include in future works. Graph-It also has additional features such as graph presets, a tab system for graphs, a multi-threaded algorithm system, and various others features which were added to aid the general user experience.

5.2 Future Work

Web Application: To improve the accessibility, Graph-It could be ported as a web application. This would make Graph-It accessible from more platforms, including ones currently unsupported. Given that Graph-It is made in Unity, most of the existing code could be reused when compiling for WebGL. However, testing and optimization would have to be done to ensure the performance of Graph-It in a browser environment provides a satisfactory, smooth user experience. In addition, testing would need to be done to see whether UnityStandaloneFileBrowser [2] is usable in the Web environment.

Additional Algorithms: Since Graph Theory is a broad field, there are a wide variety of algorithms that can be applied to graphs. The algorithms currently implemented are mainly mathematical properties and general algorithms. Since there are many algorithms that could be implemented, it would be beneficial to focus on allowing the ability to create custom algorithms. Work would have to be done on deciding the interface for custom algorithms. Through the interface, additional algorithms would be able to be added by the user for their own applications.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under grant numbers OIA-1301726, OIA-2019609, and OIA-2148788 Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, pages 361–362, 2009. https://ojs.aaai.org/index.php/ICWSM/article/view/13937, (Last Accessed: 7/26/2022).

[2] Gokhan Gokce and Ricardo Rodrigues. UnityStandaloneFileBrowser: A native file browser for unity standalone platforms, nov 2018. https://github.com/gkngkc/UnityStandaloneFileBrowser, (Last Accessed: 7/26/2022).

[3] MathWorks Inc. MATLAB - MathWorks, 2022. https://www.mathworks.com/products/matlab.html, (Last Accessed: 7/26/2022).

[4] Zyba Limited. CodeCogs: Online LaTeX equation editor - create, integrate, download, 2022. https://editor.codecogs.com/, (Last Accessed: 7/26/2022).

[5] Yichuan Chuan Shen. tikzcd-editor, 2022. https://tikzcd.yichuanshen.de/ (Last Accessed: 7/26/2022).

[6] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2016. https://www.pearson.com/us/higher-education/program/Sommerville-Software-Engineering-10th-Edition/PGM35255.html (Last Accessed: 7/26/2022).

[7] William A. Stein. SageMath: System for algebra and geometry experimentation, 2022. https://www.sagemath.org/, (Last Accessed: 7/26/2022).

[8] Unity Technologies. Unity Real-Time Development Platform — 3D, 2D VR & AR Engine, 2022. https://unity.com/, (Last Accessed: 7/26/2022).

[9] Wolfram. Wolfram Mathematica: Modern technical computing, 2022. https://www.wolfram.com/mathematica/, (Last Accessed: 7/26/2022).