USER MANUAL - A WEB APPLICATION THAT CALCULATES THE ALEXANDER POLYNOMIAL VIA PYSCRIPT

This manual provides a comprehensive guide for users to understand and effectively utilize the web application, available on GitHub [1], to draw knot diagrams and calculate their corresponding Alexander polynomials.

1. Key Features and Functionalities

1.1. Interactive Diagram Creation.

- Users can draw knot diagrams directly on the canvas by clicking to establish vertices and connecting them with line segments.
- The interface dynamically labels each vertex as it is created, enhancing visual clarity and reference during the drawing process.
- To close the diagram, the user must move the mouse over the first vertex. A dedicated modal dialog, the "Close Knot" dialog (Figure 1), prompts users to confirm the completion of their knot diagram. Upon confirmation, the script seamlessly connects the final vertex to the initial vertex, ensuring a closed loop structure that is fundamental to knot theory.

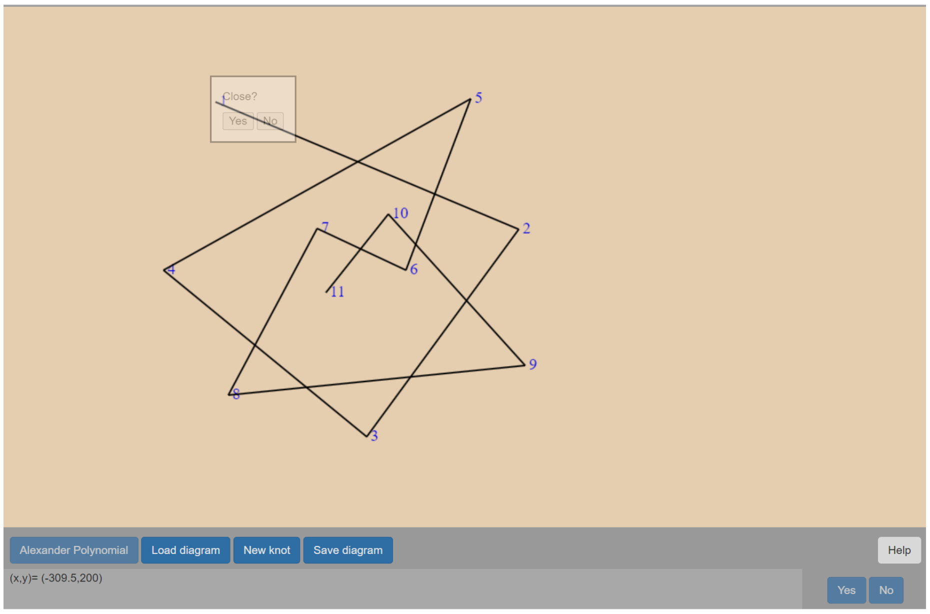
1.2. Intersection Detection and Resolution.

- The script identifies and visually highlights points where line segments intersect within the knot diagram.
- Through an interactive process, users are asked to choose which segment should pass over the other at each crossing point (Figure 2). Visual aids, such as a blinking dot and highlighted segments, assist users in making these critical decisions.

1.3. Alexander Polynomial Calculation.

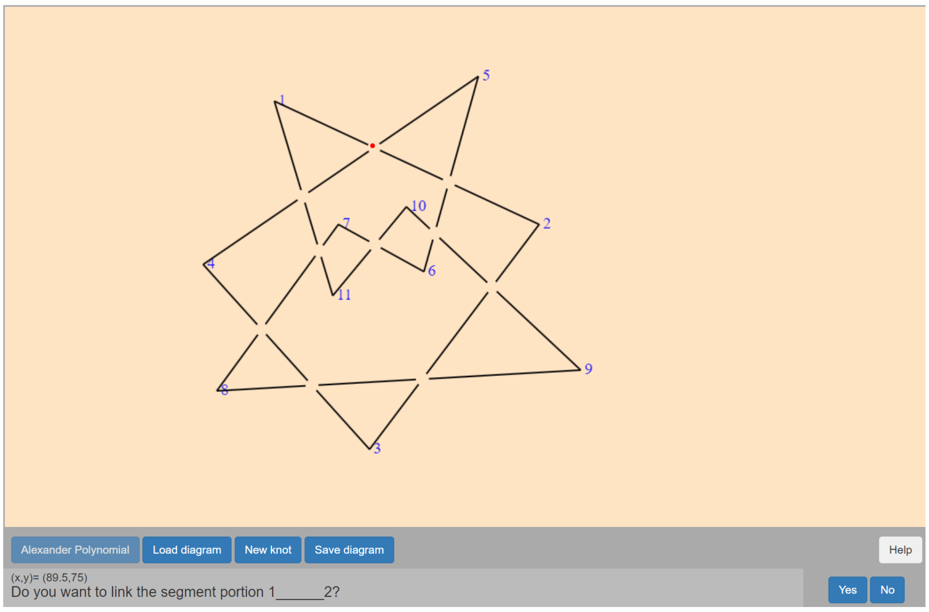
- Once the knot diagram is finalized and all crossing relationships are established, the "Alexander Polynomial" button becomes active.
- Clicking this button initiates the execution of a Python script ('main.py') embedded within the HTML using Pyodide and PyScript. After the calculation, the application displays the

[1] https://github.com/Tacioli/Alexander-Polynomial



(x,y)= (-309.5,200)

FIGURE 1. Knot diagram.



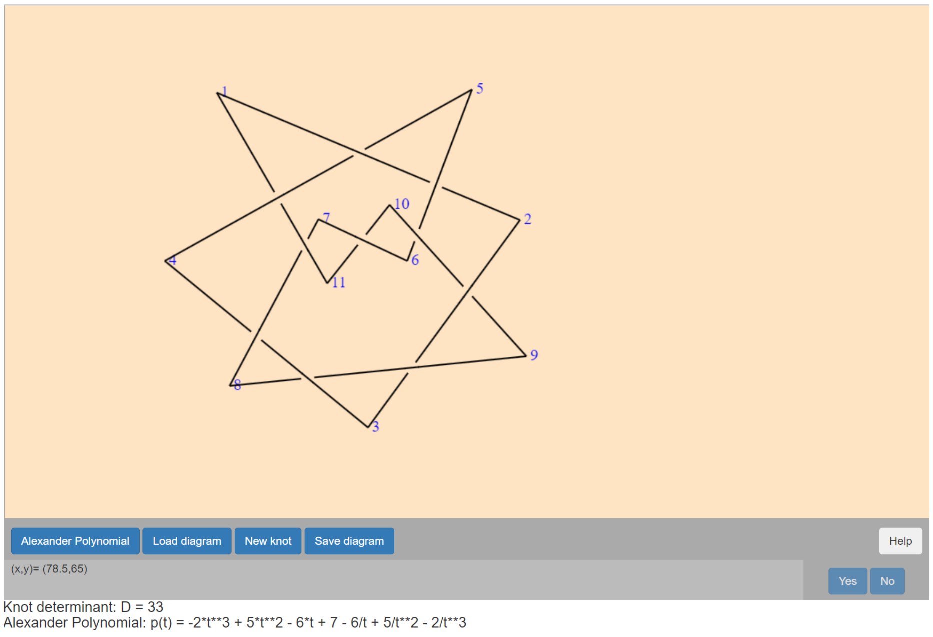
(x,y)= (89.5,75)
Do you want to link the segment portion 1______2?

FIGURE 2. Intersection detection.

determinant, $\mathbf{D} = |p(-1)|$, and the Alexander polynomial, $p(t)$, see Figure 3.

- If the user wants to change any intersection, just click on the intersection with the mouse to swap the choice.

1.4. **Diagram Persistence.**



Knot determinant: D = 33
Alexander Polynomial: p(t) = -2*t**3 + 5*t**2 - 6*t + 7 - 6/t + 5/t**2 - 2/t**3

Figure 3. Intersection detection.

- The "Save Diagram" button allows users to preserve their work by saving the current knot diagram to a `.txt` file. This file stores both the vertex coordinates and the crossing choices made by the user.
- The "Load Diagram" button enables users to retrieve previously saved diagrams, fostering continued analysis, modification, or comparison.

1.5. **Intalling.**

- Click in the link https://tacioli.github.io/Alexander-Polynomial/ to use the application.
- Go to the page https://github.com/Tacioli/Alexander-Polynomial to get the entire folder.

2. User-Friendly Interface

- The interface is built using Bootstrap, ensuring a clean, organized, and responsive layout.
- Interactive buttons trigger actions, while text element display the calculated Alexander polynomial.
- A "Help" button reveals a set of instructions, guiding users on how to effectively utilize the application's features.
- The "New knot" button clears the canvas.

3. Additional Technical Details

- Buttons are dynamically enabled or disabled based on the current state of the knot diagram (e.g., the "Alexander Polynomial" button remains disabled until the diagram is closed and all crossings are resolved).
- The script continuously updates a display element to show the current mouse coordinates as the user interacts with the canvas.
- Open only files in `.txt` format.
- When separately creating a `.txt` file of a diagram of a knot, make sure not to create blank lines.