Filippo Riva, Asier Olan, Bernardo Dias

MT2200

Final Project – Report

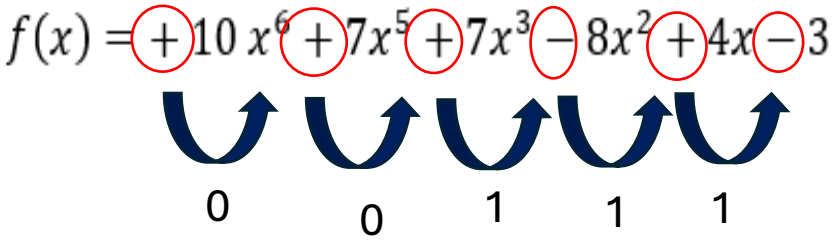
Mathematics and Example:

Descartes' Rule of Signs is a mathematical theorem that helps determine the possible number of positive and negative real roots of a polynomial equation, where a root is any value of x that satisfies f(x) = 0. According to this rule, the number of positive real zeros of the function f(x)f(x)f(x) is equal to the number of sign changes in f(x), or the number of sign changes minus an even integer greater than 0. Similarly, the number of negative real zeros is determined by analyzing f(−x); it is equal to the number of sign changes in f(-x), or the number of sign changes minus an even integer greater than zero

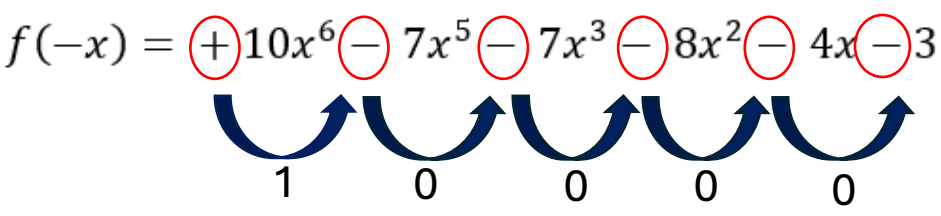
Initial function:

$$f(x) = +10\,x^6 + 7x^5 + 7x^3 - 8x^2 + 4x - 3$$

Positive roots: [3,1]



Negative roots: [1]



Report of the project:

The project was successful overall, largely because the team began working on it early and divided the tasks effectively. Filippo and Bernardo focused on developing the code, while Asier worked on simplifying the explanation of the theorem for the class and researching real-world applications. One of the main challenges was designing a function to count the number of sign changes in the polynomial, both in its original form and after negation, without having to write separate functions for each case. Initially, we used a copy of the polynomial to count the sign changes for the positive polynomial. After obtaining the count, we applied a for loop to subtract 2 (the smallest even number) from the count until all positive roots were determined. To count the sign changes in the negative version of the polynomial, we worked directly with the original polynomial, negating it within the function, and created a new counter to determine how many times the sign changed. This approach gave us accurate counts for both the positive and negative cases. From there, we applied the for loops to calculate both positive and negative roots without needing to rewrite the function, simplifying the process and improving efficiency. Finally, we decided to utilize our Python skills along with libraries and functions already available in Sage to create an accurate graphical representation of the polynomial, highlighting both the positive and negative roots. To find the roots, we used the "*roots*" function from the NumPy library, which is pre-installed in CoCalc. This approach allowed us to efficiently compute and visually display the roots, providing a clear and precise understanding of the polynomial's behavior. In conclusion, applying our computational knowledge to develop a program based on mathematical rules was a good experience. Attending the presentations was equally impactful, as it provided insights into how mathematical concepts can be applied practically. Conducting research and exploring how our algorithm could potentially aid in the growth of businesses was particularly fascinating. We would love to see more groups focus on identifying real-world applications of their theorems, showcasing how mathematics can drive innovation and contribute to business development.