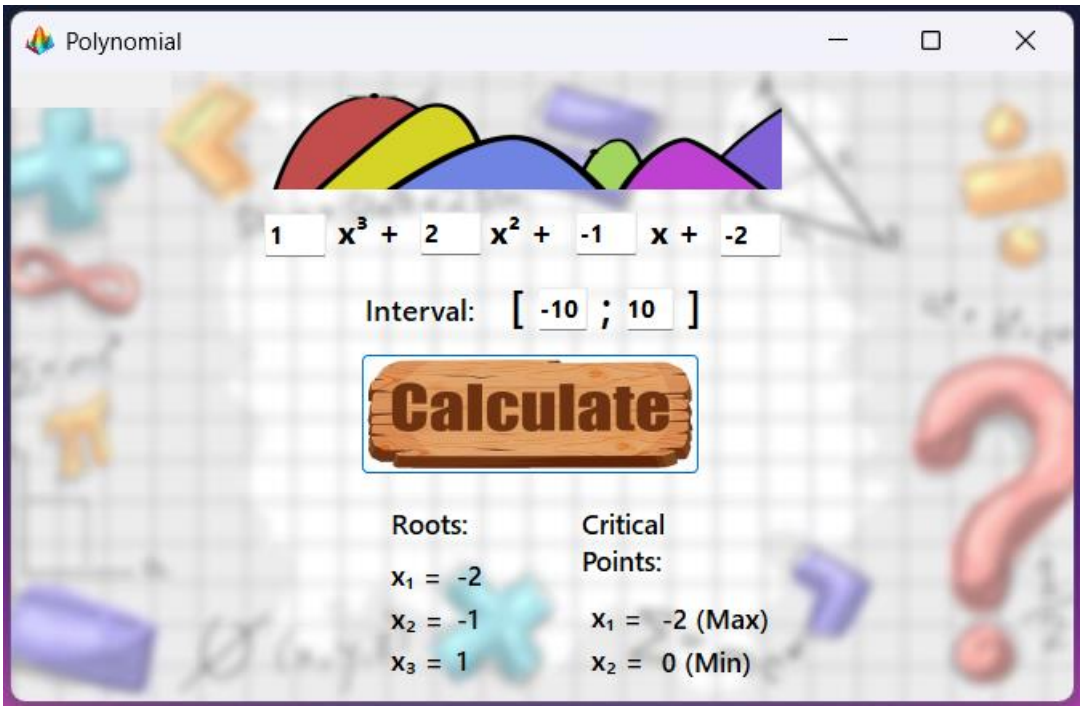
Proyecto Computacional

"Your future is whatever you make it,

so make it a good one"



Implementación

El proyecto computacional tiene como objetivo analizar polinomios de grado menor o igual a 3 de la forma $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ y extraer sus raíces reales, así como sus puntos críticos (máximos, mínimos y constantes). Para ello se desarrollaron un total de 11 métodos dónde 5 de ellos son los principales protagonistas:

- `public double Evaluate(double[] poly, double num)`
- `public bool Bolzano(double[] pol, double inf, double sup)`
- `public void Bisection(double[] poly, double inf, double sup, List<double> zero)`
- `public double[] GetFirstOrder(double[] poly)`
- `public string MaxOrMin(double[] first_order, double critical_point)`

Evaluate, toma dos parámetros: un arreglo que representa los coeficientes de un polinomio (`poly`) y un número (`num`). La función calcula el valor del polinomio (`poly`) evaluado en el número dado (`num`). Recorre cada coeficiente del polinomio, lo multiplica por el valor del número (`num`) elevado a la potencia correspondiente. Suma el resultado de cada multiplicación y lo almacena en una variable (`result`), una vez finalizado el recorrido retorna la variable `result`.

Bolzano, recibe tres parámetros en su definición: un arreglo de los coeficientes del polinomio (`poly`), un límite inferior (`inf`) y un límite superior (`sup`). Este método verifica si un polinomio tiene una raíz en un intervalo dado utilizando el Teorema de Bolzano, retorna `false` si no posee raíces y `true` en caso contrario, recordando dicho teorema:

> Sea $f(x)$ una función continua en el intervalo cerrado $[\,a; b\,]$. Si f cumple que $f(a_1) \cdot f(b_1) < 0 \;\; \forall a_1, b_i \epsilon\, [\,a; b\,]$ entonces existe $c\, \epsilon\, [\,a; b\,]$ tal que $f(c) = 0$

Bisection es el método principal del proyecto, admite cuatro parámetros en su implementación: un arreglo de los coeficientes del polinomio (`poly`), un límite inferior (`inf`) y un límite superior (`sup`) y una listada en la cual almacenará las raíces halladas (`zero`). Este método de manera recursiva calcula el punto medio del intervalo dado obteniendo así I_1 y I_2 verifica la existencia de raíces en estos nuevos intervalos y si existe la agrega a un listado de raíces, si no se encuentra una raíz, ajusta los límites superior e inferior en pequeños incrementos y repite el proceso hasta llegar a un límite definido $b_n - a_n < 0{,}0001$, (este valor sería nuestra *épsilon* ε).

GetFirstOrder, acopla un arreglo como parámetro el cual representa los coeficientes del polinomio (`poly`). El objetivo de este es calcula y retornar la derivada de primer orden del polinomio especificado. Para ello recorre cada coeficiente del polinomio original, excepto el primero ($\frac{d}{dx}(a_0) = 0$), multiplica el coeficiente por su exponente correspondiente y almacena el resultado en un nuevo arreglo (`first_order`) el cual es retornado una vez finalizado el proceso.

MaxOrMin, reicbe dos parámetros: un arreglo que representa los coeficientes de la primera derivada (`first_order`) del polinomio original y el valor (`critical_point`) a analizar. Este método determina si el valor especificado (`critical_point`) es un máximo, mínimo o constante para la derivada de primer orden de un polinomio (`first_order`). Evalúa la derivada en los puntos (`critical_point - 0.5`) y (0.5 + `critical_point`). Si el valor a la izquierda es positivo y a la derecha es negativo, devuelve “(Max)”. Si es al revés, devuelve “(Min)”. Si ambos tienen el mismo signo, devuelve “(Const)”.

> **Nota:** El programa cuenta con un algoritmo y una interfaz para verificar que los valores de entrada sean correctos

```
// Jery Rodríguez Fernández
```