Efficient Way of Find The Root of a High Degree Polynomial

By -

Reva Bhatt

HYPOTHESIS

Based on my research, there is a more efficient way for a computer to calculate the first root of a high degree polynomial.

QUESTION

Can a java program be created to efficiently solve to find the root of a high degree polynomial?

EXPECTED OUTCOMES

Overall, I am expecting a successful outcome. I really believe that a java program will efficiently find and have an accurate response to find the root. I know that there are math methods for java programming that allow us to find the solution using a program, so I will research to see if there is one/multiple for high degree polynomials to find the roots.

FORMULA USED

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

The formula shown above is used in the Newton-Raphson method. This formula is an iterative numerical method. It is known for finding the roots of polynomials, which was the task for this experiment. How this method works is that it automatically assumes that x0 is a root. It raises the value of the root till it becomes the actual root testing with larger and larger numbers.

PROCEDURE

Research on how to find the root of a high degree polynomial through different methods

↓

Begin the programming process in java

↓

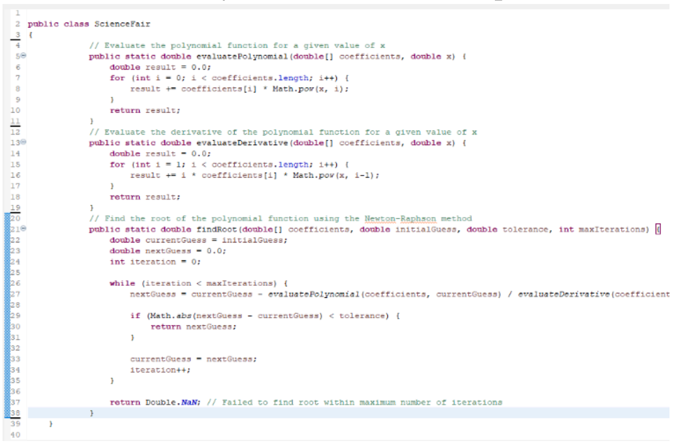
Test the program with different polynomials equations with different degrees to make sure the code is accurate and consistent

↓

Record data and compare

DATA ANALYSIS

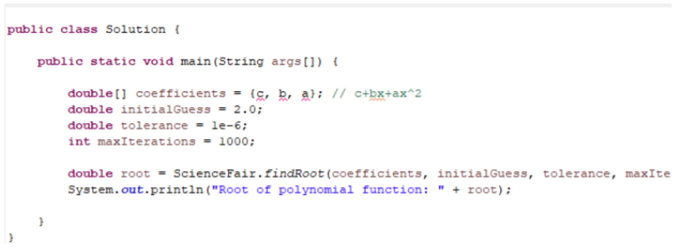
Based on the program's output, the accuracy of finding the root will be checked to see if it's accurate and analyzed with different equations.



```java

public class ScienceFair
{
            // Evaluate the polynomial function for a given value of x
            public static double evaluatePolynomial(double[] coefficients, double x) {
                double result = 0.0;
                for (int i = 0; i < coefficients.length; i++) {
                    result += coefficients[i] * Math.pow(x, i);
                }
                return result;
            }
            // Evaluate the derivative of the polynomial function for a given value of x
            public static double evaluateDerivative(double[] coefficients, double x) {
                double result = 0.0;
                for (int i = 1; i < coefficients.length; i++) {
                    result += i * coefficients[i] * Math.pow(x, i-1);
                }
                return result;
            }
            // Find the root of the polynomial function using the Newton-Raphson method
            public static double findRoot(double[] coefficients, double initialGuess, double tolerance, int maxIterations) {
                double currentGuess = initialGuess;
                double nextGuess = 0.0;
                int iteration = 0;

                while (iteration < maxIterations) {
                    nextGuess = currentGuess - evaluatePolynomial(coefficients, currentGuess) / evaluateDerivative(coefficient

                    if (Math.abs(nextGuess - currentGuess) < tolerance) {
                        return nextGuess;
                    }

                    currentGuess = nextGuess;
                    iteration++;
                }

                return Double.NaN; // Failed to find root within maximum number of iterations
            }
    }

```

Above is the program I have created to find the values through the polynomial.

↓



```java
public class Solution {

    public static void main(String args[]) {

        double[] coefficients = {c, b, a}; // c+bx+ax^2
        double initialGuess = 2.0;
        double tolerance = 1e-6;
        int maxIterations = 1000;

        double root = ScienceFair.findRoot(coefficients, initialGuess, tolerance, maxIte
        System.out.println("Root of polynomial function: " + root);

    }
}
```

This program above is a class within the original program where you have to input the specific value of each digit in the given polynomial. In the third line of the program, you can see I have shown the spot where to enter the values. This program works with more than three values, but I have just given three as an example.

ABSTRACT

The point of this science fair project was to develop a Java program that can efficiently and accurately find the first root of a high degree polynomial. The study of polynomials is a fundamental aspect of algebra and is widely used in many fields. Specifically, these polynomials are frequently found in many applications such as physics, engineering, finance, and computer sciences. The program I have created uses the Newton-Raphson method to find the value.

The requirements for the program to function is to enter the numerical values of the polynomial in reverse order, proving that it is very user friendly and very simple to use. For example, using this simple polynomial- x^2+5x-6, the values entered into the program would be -6, 5, 1. This program was tested with many different polynomials consisting of different degrees to evaluate its performance.

The results of the experiments showed that the program was able to accurately find the roots of high-degree polynomials. This program proved that it worked efficiently and precisely. Solving the root of a polynomial is very time consuming and tedious. Standard methods of finding roots including factoring and manually graphing are extremely difficult when polynomials become more complex even for a computer. I want to find an efficient way for a computer to solve this. The project's value comes from its ability to showcase the possible uses of numerical methods across a variety of fields and to show how crucial they are in solving complicated mathematical issues.

The method used in the program is an iterative technique. It uses the first derivative of the given polynomial to calculate the root. It approximates a curve by a line at each point, finding where the curve hits the x-axis. The process is then repeated until the desired level of accuracy is attained, using this point as the new root approximation.

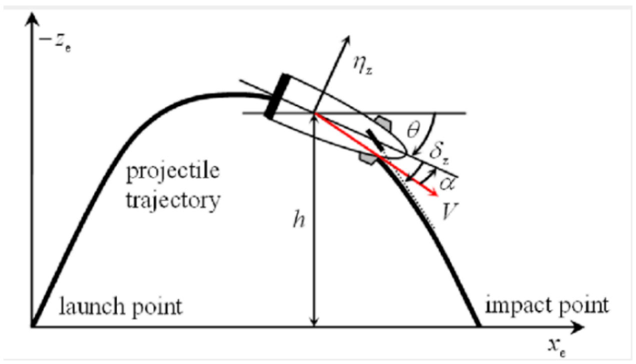
In conclusion, this study shows how well numerical methods work for locating the roots of high-degree polynomials. The created Java program is a practical tool that may be applied to numerous scientific fields that call for the solution of high degree polynomial equations.

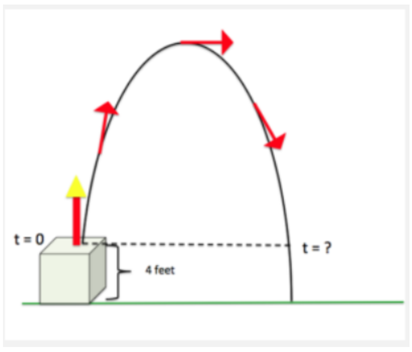
IMPACT

The roots of high degree polynomials are used in many different aspects of life such as engineering, physics, finance, computer science, and more. Finding these roots in an effective way could make a major impact on society possibly through:

- Advancements in science/technology: The ability to solve challenging mathematical problems, such as locating the roots of high degree polynomials, is essential for many scientific and engineering applications. New techniques for resolving these kinds of equations could result in advances in many fields.

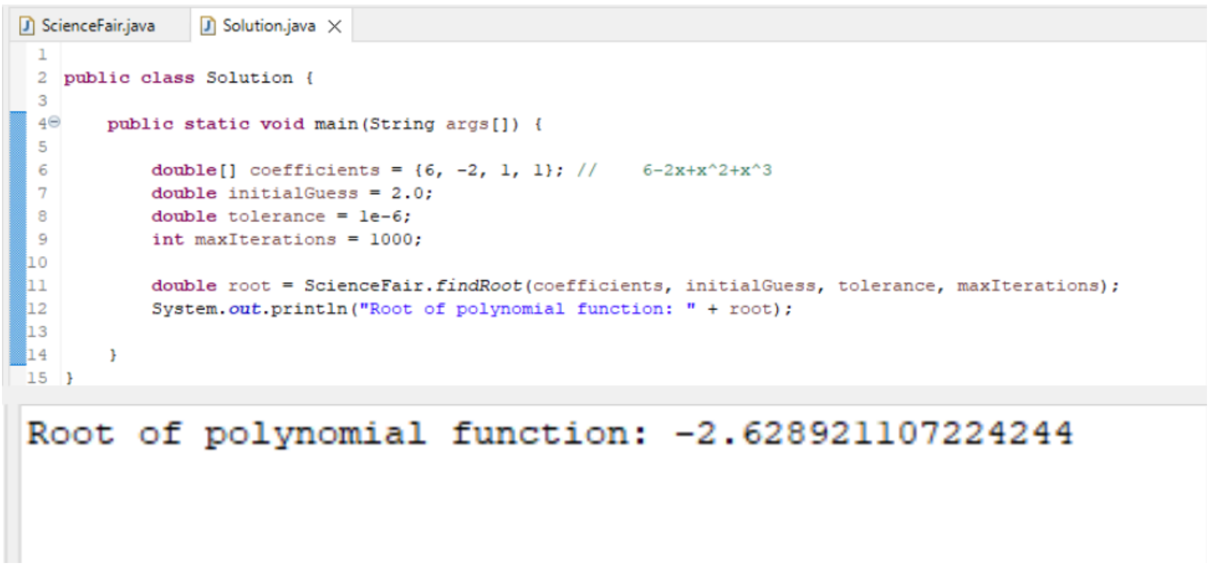
- Economic benefits: The development of new businesses and jobs in several industries may result from the discovery of new approaches for solving high degree polynomials, like this program. The economy might benefit from this in the long run.





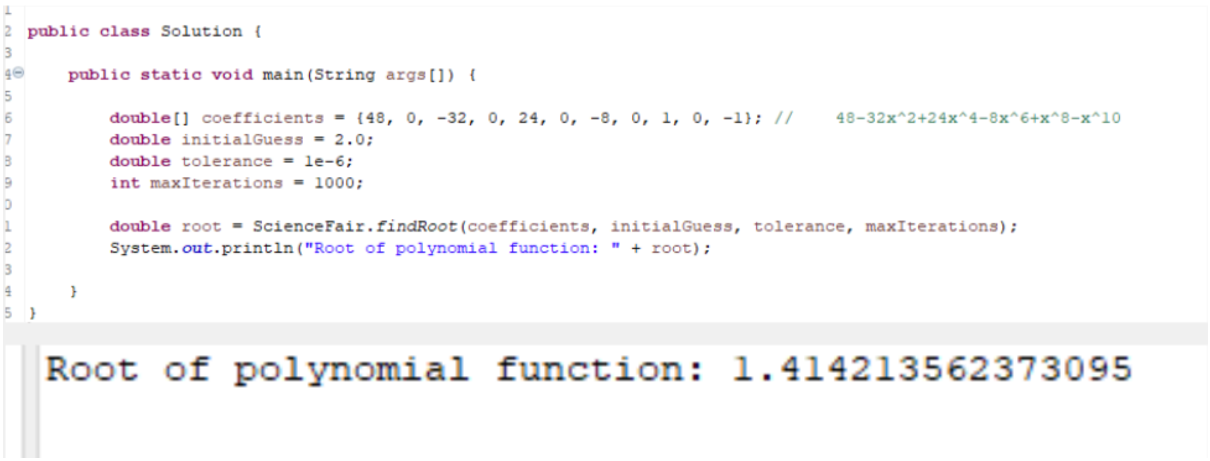
- Better decision making: It is challenging to solve many real-world problems that involve complex mathematical models without identifying the roots of high degree polynomials. We could improve our decision-making and results if we could solve these equations more effectively with a program.

RESULT



```

public class Solution {

    public static void main(String args[]) {

        double[] coefficients = {6, -2, 1, 1}; //     6-2x+x^2+x^3
        double initialGuess = 2.0;
        double tolerance = 1e-6;
        int maxIterations = 1000;

        double root = ScienceFair.findRoot(coefficients, initialGuess, tolerance, maxIterations);
        System.out.println("Root of polynomial function: " + root);

    }
}
```

```
Root of polynomial function: -2.628921107224244
```



```
public class Solution {

    public static void main(String args[]) {

        double[] coefficients = {48, 0, -32, 0, 24, 0, -8, 0, 1, 0, -1}; //    48-32x^2+24x^4-8x^6+x^8-x^10
        double initialGuess = 2.0;
        double tolerance = 1e-6;
        int maxIterations = 1000;

        double root = ScienceFair.findRoot(coefficients, initialGuess, tolerance, maxIterations);
        System.out.println("Root of polynomial function: " + root);

    }
}
```

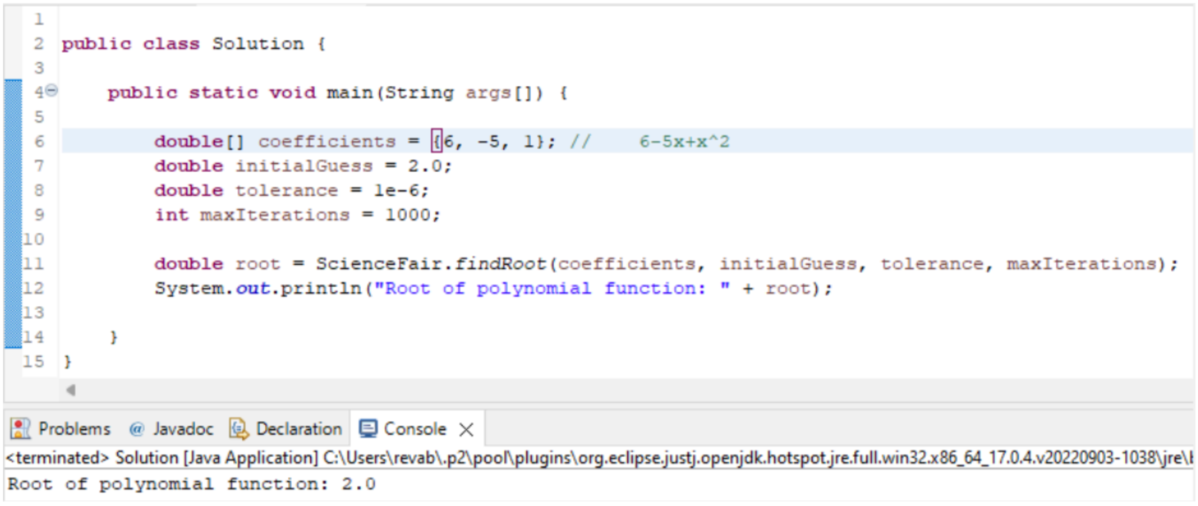
```
Root of polynomial function: 1.414213562373095
```

The program successfully worked in giving the value of the root of the given polynomial. This program is very user friendly and not hard to use at all.

CONCLUSION

The hypothesis that there is a more efficient way for a computer to calculate the first root of a high degree polynomial, is accepted. The data provided shows that the program is efficiently and accurately finding the first root of the given polynomial.

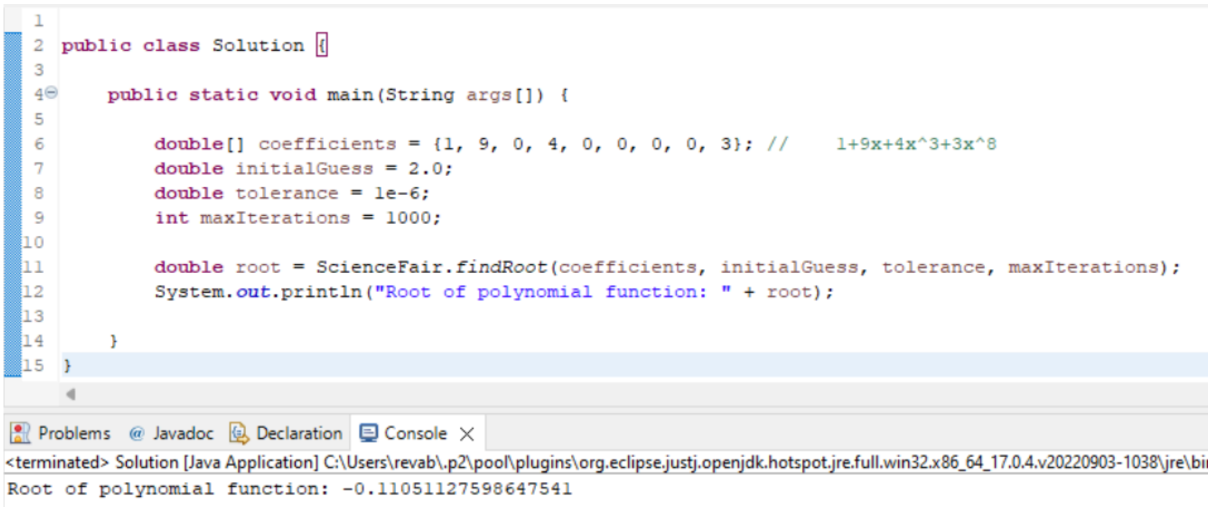
Lower Degree



```
public class Solution {

    public static void main(String args[]) {

        double[] coefficients = {6, -5, 1}; //    6-5x+x^2
        double initialGuess = 2.0;
        double tolerance = 1e-6;
        int maxIterations = 1000;

        double root = ScienceFair.findRoot(coefficients, initialGuess, tolerance, maxIterations);
        System.out.println("Root of polynomial function: " + root);

    }
}
```

```
<terminated> Solution [Java Application] C:\Users\revab\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\
Root of polynomial function: 2.0
```

Higher Degree



```
public class Solution {

    public static void main(String args[]) {

        double[] coefficients = {1, 9, 0, 4, 0, 0, 0, 0, 3}; //    1+9x+4x^3+3x^8
        double initialGuess = 2.0;
        double tolerance = 1e-6;
        int maxIterations = 1000;

        double root = ScienceFair.findRoot(coefficients, initialGuess, tolerance, maxIterations);
        System.out.println("Root of polynomial function: " + root);

    }
}
```

```
<terminated> Solution [Java Application] C:\Users\revab\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bi
Root of polynomial function: -0.11051127598647541
```

FUTURE DIRECTION

My experiment results are very encouraging. I believe that such programs can help speed up processes of jobs. A future direction of this project would be finding every root of the polynomial. In my research, I have encountered additional methods to find the most dominant root of a polynomial using linear algebra- power and QR method. My goal is to explore this area and develop programs for them too.

BIBLIOGRAPHY

www.khanacademy.org

https://brilliant.org/wiki/newton-raphson-method/#:~:text=The%20Newton%2DRaphson%20method%20(also,straight%20line%20tangent%20to%20it.

https://web.mit.edu/10.001/Web/Course_Notes/NLAE/node6.html