Método de la ingeniería.

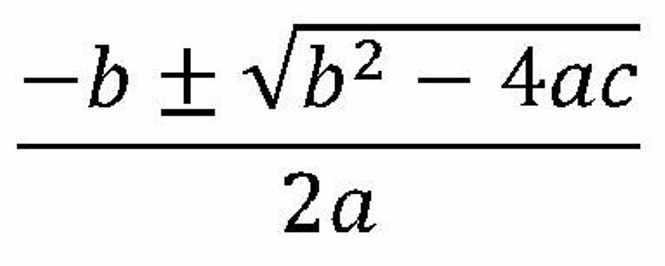
1.Identificar el problema.

2. un polinomio del latín polynomium, y este del griego, πολυς polys ‘muchos’, ​ es una expresión algebraica constituida por una suma finita de productos entre variables (valores no determinados o desconocidos) y constantes (números fijos llamados coeficientes), o bien una sola variable. Las variables pueden tener exponentes de valores definidos naturales incluido el cero y cuyo valor máximo se conocerá como grado del polinomio.

Las raíces de un polinomio son números tales que hacen que un polinomio valga cero.

Existen algunos casos de polinomios donde se presentan raíces que son evidentes, es decir, que podemos averiguar simplemente mediante la observación y evitando hacer cálculos demasiado trabajosos.

**La fórmula de Bhaskara**, que te permitirá averiguar fácilmente cuales son las raíces.

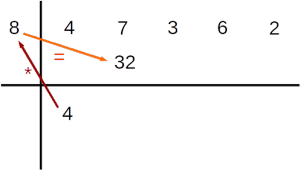


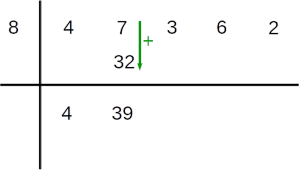
**El método de Ruffini** para polinomios nos permite dividir un polinomio de grado 2 o mayor con el fin de obtener uno de un grado menor, para seguir en la búsqueda de las raíces que aún no se conozcan. Sirve mucho en el caso de polinomios de tercer grado, ya que si logramos descifrar alguna de sus raíces podremos obtener uno de grado 2 y finalmente mediante la fórmula de Bhaskara, averiguar las dos raíces restantes.

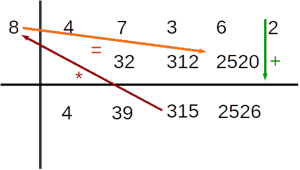
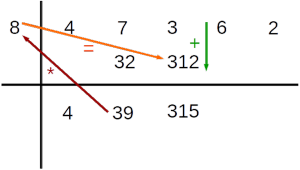
Los siguientes métodos se ven apoyados en gran medida por **método de Horner** que presentaba una nueva manera más rápida de calcular el valor de un polinomio y de manera análoga su convergencia a 0:

Colocamos los coeficientes del polinomio en una tabla junto con el valor de x que quiere evaluarse

Bajamos el primer coeficiente y lo multiplicamos por el valor de x colocando el resultado debajo del siguiente coeficiente en la tabla

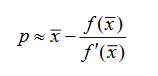


Sumamos los dos valores obteniendo un nuevo resultado parcial

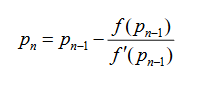
Repetimos la operación para cada coeficiente

Al llegar al último coeficiente obtenemos el resultado final.

Podemos ver en este ejemplo como se ha reducido la cantidad de multiplicaciones necesarias ahorrando al sistema 6 operaciones extras.

**El método de Newton** para hallar las raíces de la ecuación f(x) = 0, es el más conocido, y a menudo, el más efectivo.

El método de Newton consiste en tomar una aproximación inicial, x, y a continuación obtener una aproximación más refinada mediante la fórmula de arriba. Es decir, se trata de acercarnos a la raíz p por medio de la fórmula recursiva:



**El Método Muller** es un algoritmo de búsqueda de raíces para encontrar la raíz de una ecuación de la forma, f (x) = 0. Fue descubierto por David E. Muller en 1956.

Comienza con tres supuestos iniciales de la raíz, y luego construye una parábola a través de estos tres puntos, y toma la intersección del eje x con la parábola como la siguiente aproximación. Este proceso continúa hasta que se encuentra una raíz con el nivel de precisión deseado.

Aunque es más lento que el método de Newton - Raphson, que tiene una tasa de convergencia de 2, pero supera uno de los mayores inconvenientes del método de Newton-Raphson, es decir, el cálculo de la derivada en cada paso.

**El enfoque de Bairstow** es usar el método de Newton para ajustar los coeficientes u y v en la ecuación cuadrática x^2 + ux + v hasta que sus raíces también sean las raíces del polinomio resuelto. Las raíces de la ecuación cuadrática pueden entonces determinarse, y el polinomio puede dividirse por la ecuación cuadrática para eliminar esas raíces. Este proceso se itera entonces hasta que el polinomio se vuelve cuadrático o lineal, y se han determinado todas las raíces.

Cómo la definición de polinomio, la definición de una raíz, métodos comunes para encontrarlas, algoritmos conocidos para realizar esta actividad.

Podemos buscar un poco de la historia de los métodos que se encontraron para resolver este problema.

Primera aproximación Newton-Horner solo crea reales.

Mejor Bairstow's Method or Mullers metod.

Explicacion del porque escoger esos 2:

<https://youtu.be/zEvfkSuPqWk>

Código para Baristows:

<https://youtu.be/dmsosSGVxgs>

Explicación.

<https://github.com/ergenekonyigit/Numerical-Analysis-Examples/blob/master/Java/Bairstow's%20Numerical%20Analysis%20method.java>

Aquí esta con algunos comentarios muy buenos.

Código para Muller:

<https://youtu.be/XIIEjwtkONc>

Explicación.

<https://www.geeksforgeeks.org/program-muller-method/>

Un blog con muy buenas explicaciones que deberían ir aquí.

<https://www.programcreek.com/java-api-examples/?class=org.apache.commons.math.util.MathUtils&method=sign>

Una pagina llena de códigos interesantes.

3. Solo enunciar las ideas que se nos ocurren para resolver el problema incluyendo los algoritmos que habíamos encontrado.

-Basada en la investigación anterior podemos determinar que tratar de crear un método para hallar raíces resulta más complejo de lo que se quisiera admitir, aunque tratar de hacer una búsqueda completa de las raíces sería un ejercicio interesante, no resultaría particularmente útil, dado que la complejidad temporal podría (y sería lo más probable) ser exponencial. Incluso creando restricciones a partir de las opciones que nos plantea Ruffini con la división del coeficiente de mayor grado sobre el de menor grado, para determinar las posibles raíces el proceso no sería remotamente tan eficiente como el de las siguientes opciones a considerar.

-Codificar los métodos de Newton, Muller o Bairstows, este ejercicio resulta más conveniente dado que estos procedimientos están dados en sí mismos como algoritmos iterativos en los que se busca a partir de un proceso matemático tratar de aproximar los valores de las raíces.

4. Poner los algoritmos que creamos pueden resolver el problema, sin importar que podamos o no analizar su complejidad temporal, poner la complejidad temporal de las que podamos crear (Deben ser al menos 3).

5. Para encontrar las respuestas que estamos buscando debemos tener en cuenta factores como la complejidad temporal, y el hecho de que podamos calcularla.

Bibliografía:

<https://prezi.com/kdjcizosymv7/metodo-de-muller/>

<https://www.geeksforgeeks.org/program-muller-method/>

<https://github.com/ergenekonyigit/Numerical-Analysis-Examples>

<https://www.youtube.com/watch?v=dmsosSGVxgs&feature=youtu.be>