Reto 1 - Análisis Numérico

Santiago Romero, Sandra Chávez, Ricardo Bernal

12 de Septiembre del 2020

1. Introducción

Existen diversos métodos para hallar las raíces de una función que ofrecen garantizar la precisión decimal que sea deseada. A continuación se presentará un taller enfocado en la implementación y comparación de eficiencia de los métodos de Newton-Horner, Laguerre, Brent, Remez y Taylor.

2. Evaluación de raíces de un polinomio

1. Análisis Geométrico y Explicación del Algoritmo

   En el método de Newton-Raphson, busca un cero de la función $f(x)$ por aproximaciones sucesivas a partir de una modificación inicial X_0. El valor sucesivo $X_n + 1$ es la abscisa del punto en la que la tangente a la gráfica de $f(x)$ en X_n corta al eje x.

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

   La anterior ecuación es equivalente a:

   $$g(x) = x - \frac{f(x)}{f'(x)} \tag{2}$$

   Para garantizar la convergencia del método se requiere algún conocimiento acerca de la primera y segunda derivada. De forma particular, podemos afirmar que si $f(x)$ y $f'(x)$ no se anulan, conservan el signo $[a, b]$ y si se cumple $'f(x_0) * f''(x_0) > 0$, con x_0 y la raíz pertenecientes a $[a, b]$, el método converge. Si no se cumplen dichas condiciones, el método diverge

   En esta modificación denominada Método de Newton-Horner, se evaluara el valor y la derivada del polinomio usando el método de Horner dentro del método de Newton.

2. Casos de prueba

- Función 1:

$$x^4 - 9x^2 - 5x^3 + 155x - 250 = 0 \tag{3}$$

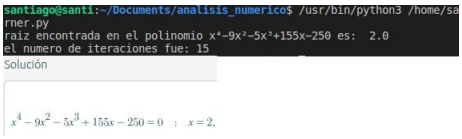

Figura 1: Resultados del caso de prueba pedido en taller: Wolfram vs Algoritmo de Newton-Horner

- Función 2:

$$10x^5 + 3x^2 - 2 = 0 \tag{4}$$

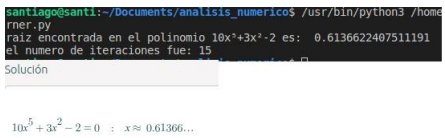

Figura 2: Resultados del caso de prueba 2: Wolfram vs Algoritmo de Newton-Horner

- Función 3:

$$x^3 - 2x^2 + 4x/3 - 8/27 = 0 \tag{5}$$

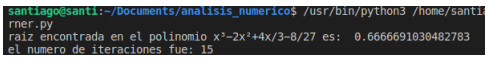

Figura 3: Resultados del caso de prueba 2: Wolfram vs Algoritmo de Newton-Horner

3. Eficiencia del Algoritmo

- Función 1: Gráfica de Iteraciones vs Tolerancia

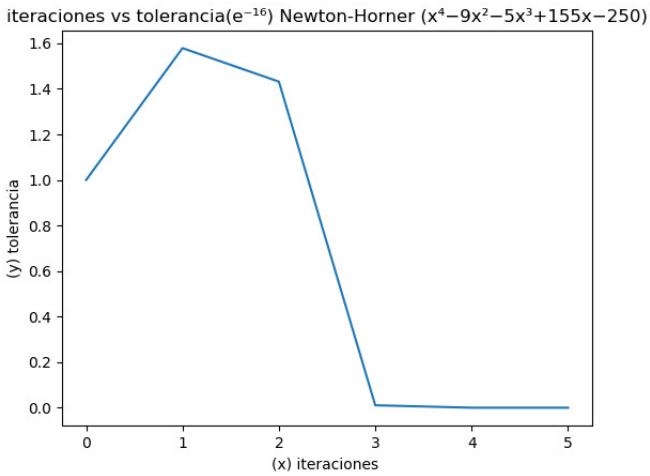

Figura 4: Gráfica de caso de prueba 1. tolerancia vs. iteraciones Algoritmo de Newton-Horner

Gráficas Error vs Error+1

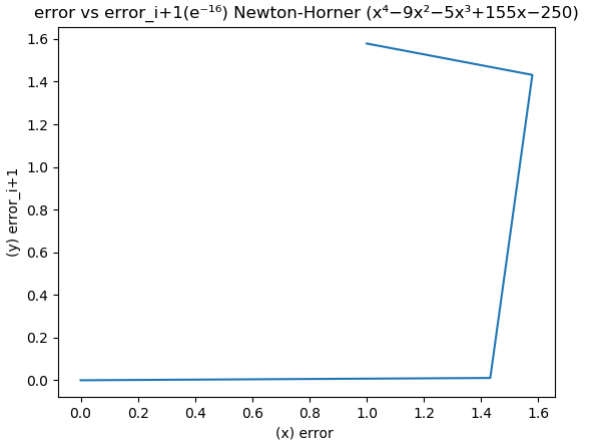

Figura 5: Gráfica de caso de prueba 1. error vs. error i+1 Algoritmo de Newton-Horner

- Función 2: Gráfica de Iteraciones vs Tolerancia

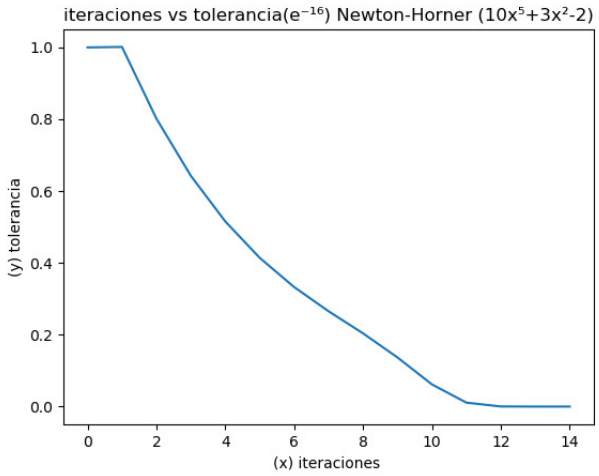

Figura 6: Gráfica de caso de prueba 2. tolerancia vs. iteracionesAlgoritmo de Newton-Horner

Gráficas Error vs Error+1

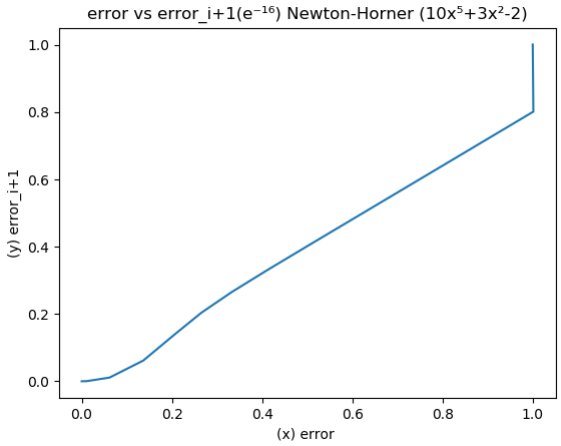

Figura 7: Gráfica de caso de prueba 2. error vs. error i+1 Algoritmo de Newton-Horner

- Función 3: Gráfica de Iteraciones vs Tolerancia

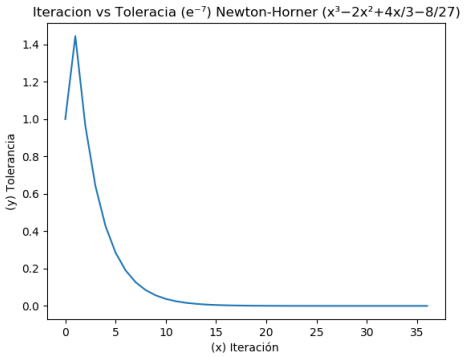

Figura 8: Gráfica de caso de prueba 2. tolerancia vs. iteraciones Algoritmo de Newton-Horner

Gráficas Error vs Error+1

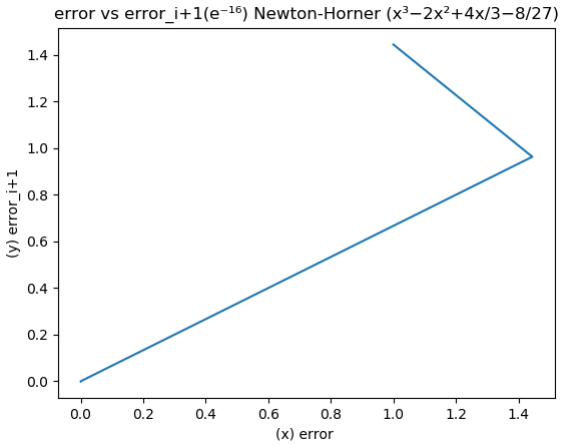

Figura 9: Gráfica de caso de prueba 2. error vs. error i+1 Algoritmo de Newton-Horner

3. Algoritmo de Brent

El algoritmo de Brent es un método híbrido que utiliza en cada punto lo mas conveniente de las estrategias de bisección y de secante (o Muller); este método también conocido como *zeroin* o $fzero$, ha sido el método mas popular para encontrar ceros de funciones desde que se desarrollo en 1972. Este método suele converger muy rápidamente a cero; para las funciones dificiles ocasionales que se encuentran en la practica.

1. Análisis Geométrico y Explicación del Algoritmo

   El método se aplica una función f continua en un intervalo delimitado por a y b donde $f(a) * f(b) < 0$. En cada iteración , hay tres puntos o valores involucrados:

   - i: Numero de la iteracion
   - b_i: Es la iteración actual, la aproximación actual de la raíz del polinomio
   - a: Es el 'contrapunto', es decir donde la función tiene signo opuestos.
   - b_{i-1}: Es la iteración anterior, inicialmente definida como a_0
   - δ

   $$s = \begin{cases} b_k - \frac{b_k - b_{k-1}}{f(b_k) - f(b_{k-1})} f(b_k), & \text{if } f(b_k) \neq f(b_{k-1}) \\ m & \text{otherwise} \end{cases} \tag{6}$$

   Como el método de Brent se trata de una versión modificada del método de Dekker, en cada iteración se evalúan una serie de condiciones para determinar que método utilizar:

   - Si la iteración anterior utilizo el método de bisección, se debe cumplir

     $$|\delta| < |b_i - b_{i-1}| \tag{7}$$

     para realizar interpolación. De lo contrario, se realiza el método de bisección y su resultado se utiliza en la siguiente iteración.
   - Si en la iteración anterior se realizo una interpolación, se debe cumplir

     $$|\delta| < |b_{i-1} - b_{i-2}| \tag{8}$$

     para realizar interpolación. De lo contrario, se realiza método de bisección.
   - Además, si en la iteración anterior utilizo método de bisección, debe cumplirse

     $$|s - b_i| < \frac{1}{2}|b_i - b_{i-1}| \tag{9}$$

     En caso contrario, se usa bisección y su resultado es utilizado en la siguiente iteración. Si la iteración anterior utilizo interpolación, debe cumplirse

     $$|s - b_i| < \frac{1}{2}|b_{i-1} - b_{i-2}| \tag{10}$$

2. Casos de prueba

   - Función 1:

$$x^3 - 2x^2 + 4x/3 - 8/27 = 0$$

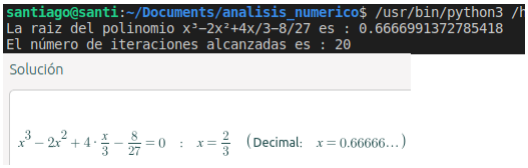

Figura 10: Resultados del caso de prueba pedido en taller: Wolfram vs Algoritmo de Brent

   - Función 2:

$$10x^5 + 3x^2 - 2 = 0 \tag{11}$$

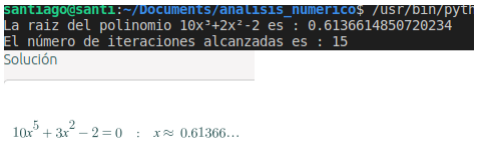

Figura 11: Resultados del caso de prueba 2: Wolfram vs Algoritmo de Brent

3. Eficiencia del Algoritmo

   - Función 1:

$$x^3 - 2x^2 + 4x/3 - 8/27 = 0$$

     Gráfica de Iteraciones vs Tolerancia

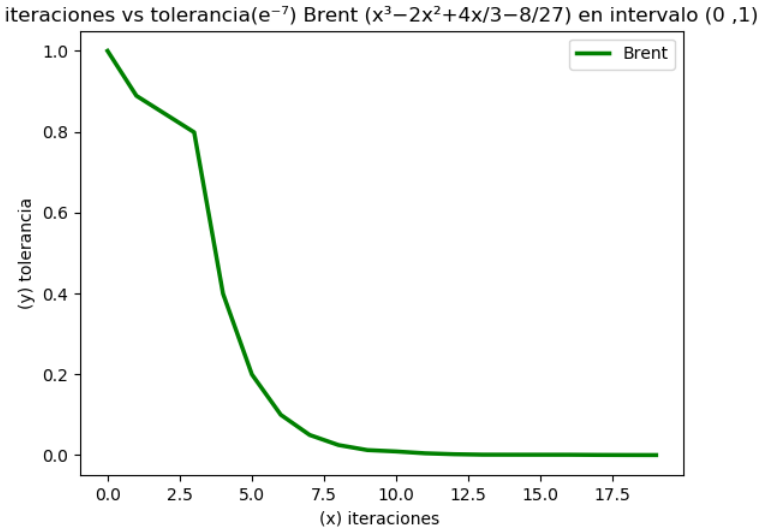

Figura 12: Gráfica de caso de prueba 1. tolerancia vs. iteraciones Algoritmo de Brent

Gráficas Error vs Error+1

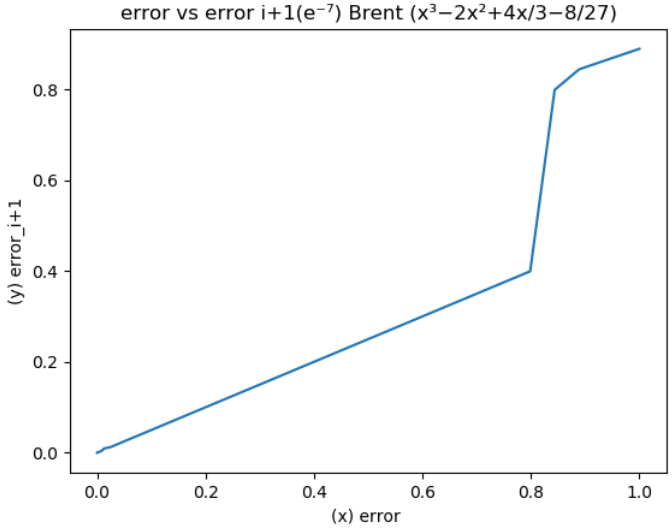

Figura 13: Gráfica de caso de prueba 1. error vs. error i+1 Algoritmo de Brent

- Función 2:

$$10x^5 + 3x^2 - 2 = 0 \tag{12}$$

Gráfica de Iteraciones vs Tolerancia

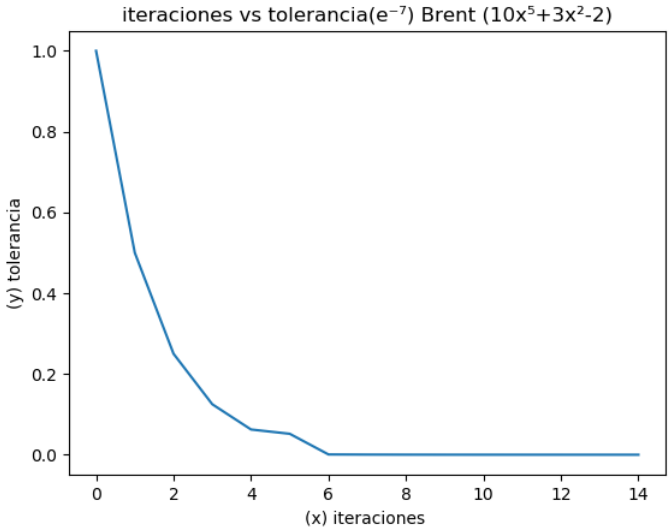

Figura 14: Gráfica de caso de prueba 2. tolerancia vs. iteraciones Algoritmo de Brent

Gráficas Error vs Error+1

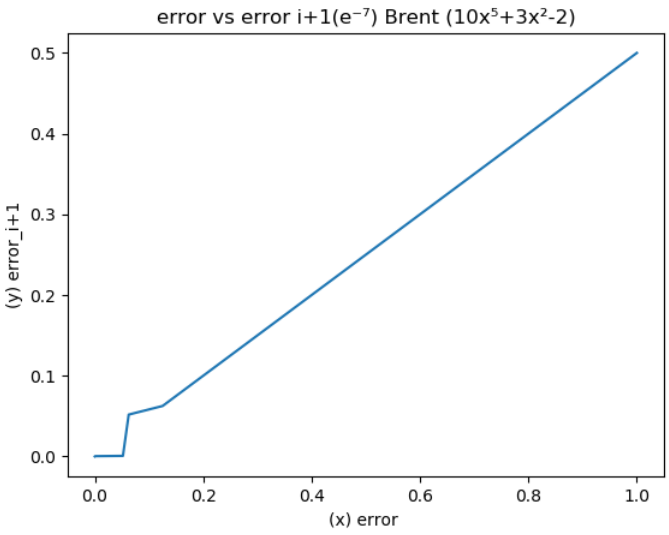

Figura 15: Gráfica de caso de prueba 2. error vs. error i+1 Algoritmo de Brent

4. Comparaciones

- Newton-Horner vs Brent.

  Como se puede observar en la figura 3 el numero de iteraciones con el algoritmo de Newton-Horner es menor a la de la figura 10 (evaluadas en la misma función) que se evalúa con el algoritmo de Bren por consiguiente el algoritmo de Newton es más efectivo con una cantidad menor de iteraciones a diferencia del de Brent que necesita muchas más iteraciones para disminuir la tolerancia.

- Comparación con Bisección

  Evaluación del Polinomio caso de estudio del método de Brent con método de bisección.

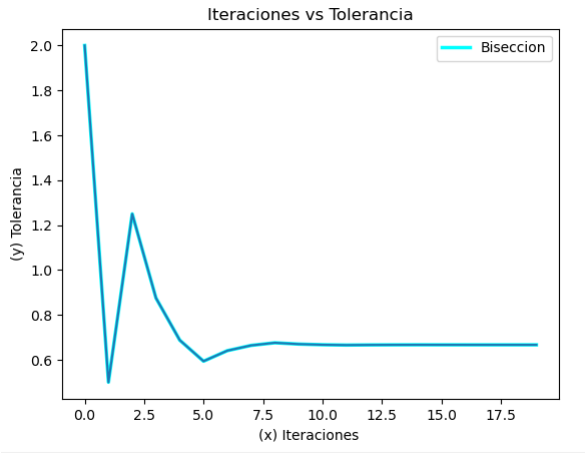

Figura 16: Gráfica Tolerancia vs Error para $x^3-2x^2+4x/3-8/27$ con el método de bisección

```
Numero Iteraciones:  22
Raiz: 0.6666660309
```

Figura 17: Resultado para $x^3 - 2x^2 + 4x/3 - 8/27$ con el método de bisección

Evaluación del Polinomio caso de estudio del método de Newton-Horner con método de bisección.

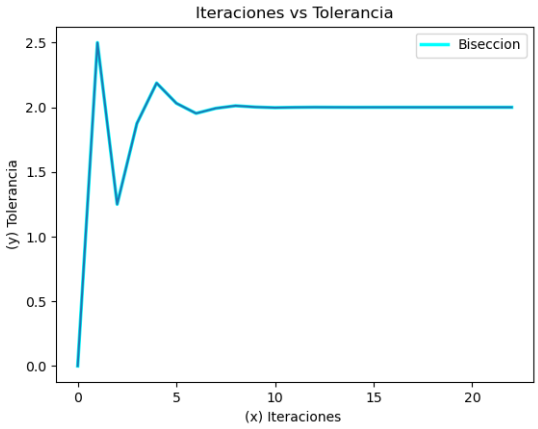

Figura 18: Gráfica Tolerancia vs Error para $x^4 - 9x^2 + 5x^3 + 155x - 250$ con el método de bisección

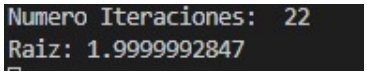

Figura 19: Resultado para $x^4 - 9x^2 + 5x^3 + 155x - 250$ con el método de bisección

Como se puede evidenciar en la figura 16, el algoritmo de bisección requiere de 22 iteraciones para encontrar la raiz del polinomio a diferencia de su par (método Brent) que tan solo requiere de 22 por lo que este converge mas rápido. Por otro lado, en el caso del polinomio de prueba para Newton-Horner, el método converge de forma mas lenta ya que requirió un numero mayor de iteraciones comparado con el método modificado de Newton-Horner

5. Conclusiones

- El método de Newton-Horner converge durante menos iteraciones que el método de Brent.

- Los resultados obtenidos por parte de los algoritmos implementados al compararse con los resultados de herramientas matemáticas, en nuestro caso Wolfram, se aproximan bastante en valores a las raíces reales disminuyendo su latencia.Por consecuencia, la implementación realizada sigue los lineamientos de los algoritmos matemáticos que se quisieron estudiar.
- El algoritmo de Newton-Horner es más efectivo con una mayor cantidad de iteraciones dado que la tolerancia va disminuyendo tal se muestra en la figura 4.
- Tal y como se muestra en los resultados anteriores el algoritmo de Brent para una ecuación determinada no suele variar mucho en cuanto a la raíz pero si en cuanto al número de iteraciones dependiendo de la tolerancia que se le añada, además este mismo algoritmo al contener en él otras validaciones importantes como lo son el de (Figura 16 a Figura 19) y la secante, permite que puedan hallarse las raíces de las funciones en pocas iteraciones, comparadas con las que se tardarían al buscar las raíces solo por una de esta validaciones.
- Profe... como puede ver, no hicimos el punto 3 pero nos esforzamos mucho :)

Referencias

- Heinold Brian. (2013). An Intuitive Guide to Numerical Methods. Mount St. Mary's University. Recuperado de: `https://www.brianheinold.net/notes/An_Intuitive_Guide_to_Numerical_Methods_Heinold.pdf`
- Larrosa C. Ignacio. Metodo de Newton-Raphson. Recuperado de: `https://www.geogebra.org/m/XCrwWHzy`
- Segura Javier. (2011) Resolucion numerica de ecuaciones no lineales. Universiad de Cantabria. Recuperado de: `https://ocw.unican.es/pluginfile.php/1070/course/section/1253/cn1tema2.pdf`
- Ryan Nick. (2017). Root-Finding Algorithms Tutorial in Python: Line Search, Bisection, Secant, Newton-Raphson, Inverse Quadratic Interpolation, Brent Metod. Recuperado de: `https://nickcdryan.com/2017/09/13/root-finding-algorithms-in-python-line-search-bisection-secant-newton-raphson-boyde`
- Wikipedia contributors. (2020). Brent's method. In Wikipedia, The Free Encyclopedia. Recuperado de: `https://es.qwe.wiki/wiki/Brents_method`
- Burden Burden y Faires [2011], Dennis y Schnabel [1996], Hager [1988], Heath [2002], Nocedal y Wright [2006], Quarteroni, Sacco y Saleri [2000] y Sauer [2012]; Lauter y Dinechin [2013]

- Burton A. (1992). Newton's Method and Fractals. Whitman College. Recuperado de: https://www.whitman.edu/Documents/Academics/Mathematics/burton.pdf