Reto 1

John Jairo Gonzalez Martinez
Daniel Esteban Tibaquira Galindo
Karen Sofia Coral Godoy
Stiven Gonzalez Olaya

13 de Septiembre 2020

1 Evaluación de las raíces de un Polinomio

1.1 Problema 1

Sea $P(x) = a_0x^n + a_1x^{n-1} + ... + a_n$ un polinomio entonces, implementar una modificación del método de Horner que evalue de manera eficiente $f'(x_0)$ y $f''(x_0)$ la primera y segunda derivada en cualquier punto

1.1.1 Metodo de Horner

Un polinomio de grado n, $p_n(x) = a_1x^n + a_2x^{n-1} + ... + a_nx + a_{n+1}$, donde los coeficientes a_i pueden ser números reales o complejos. Si tiene raíces complejas, se darán en pares de la forma $(x_r + ix_i, x_r - ix_i)$, donde x_r y x_i son las partes reales e imaginarías, respectivamente.

Para evaluar un polinomio $p_n(x)$ lo mejor es proceder secuencialmente asi:

$$\begin{gathered} p_0(x) = a_1 \\ p_1(x) = a_2 + xp_0(x) \\ p_2(x) = a_3 + xp_1(x) \\ p_3(x) = a_4 + xp_2(x) \\ ... \\ p_n(x) = a_{n+1} + xp_{n-1}(x) \end{gathered}$$

Es decir, para un polinomio de grado n

$$p_0(x) = a_1 \; p_i(x) = a_{n+i} + xp_{i-1}(x), \; i = 1, 2, ..., n$$

Las derivadas primera y segunda saldrían de estas expresiones

$$\begin{gathered} p_0'(x) = 0 \; p_i'(x) = p_{i-1}(x) + xp_{i-1}'(x), \; i = 1, 2, ..., n \\ p_0''(x) = 0 \; p_i''(x) = 2p_{i-1}'(x) + xp_{i-1}''(x), \; i = 1, 2, ..., n \end{gathered}$$

Data: f,X_0
Result: Retorna el polinomio evaluado en la función, primera derivada y segunda derivada
initialization;
n = longitud(f)-1
p = f[1]
dp = 0
ddp = 0
for *n veces* **do**
  ddp = ddp*x + 2*dp
  dp = dp*x +p
  p = p*x + a[i+1]
end
return *p,dp,ddp*

Algorithm 1: Algoritmo de Horner

1.1.2 Pruebas y Resultados

Se tomaron 3 polinomios para las pruebas y fueron evaluados en los puntos 1 y 2. El tercer polinomio se utilizo para evaluar las derivadas de un numero complejo y estas son:

a) $x^4 - 9x^2 - 5x^3 + 155x - 250$
Valor polinomio : $f(1) = 108$, $f(2) = 0$
Valor primera derivada: $f(1) = 126$, $f(2) = 91$
Valor segunda derivada: $f(1) = -36$, $f(2) = -30$

b) $x^2 - 4$
Valor polinomio : $f(1) = -3$, $f(2) = 0$
Valor primera derivada: $f(1) = 2$, $f(2) = 4$
Valor segunda derivada: $f(1) = 2$, $f(2) = 2$

c) $2x^4 + -3x^2 + 3x - 4$
Valor polinomio : $f(2 - 3i) = -221 + 267i$, $f(2) = 30$
Valor primera derivada: $f(2 - 3i) = -377 - 54i$, $f(2) = 18$
Valor segunda derivada: $f(2 - 3i) = -126 - 288i$, $f(2) = 12$

1.2 Problema 2

Utilizar los resultados anteriores y el algoritmo de Laguerre para obtener un método de, Newton-Horner, de convergencia cuadrática en el que el algoritmo de Newton reemplaza al de Laguerre.

1.2.1 Aplicar para el polinomio $x^4 - 9x^2 - 5x^3 + 155x - 250$ el algoritmo creado

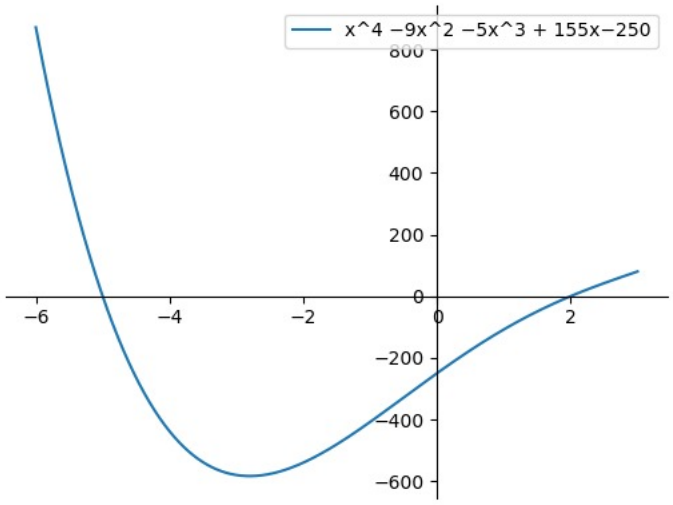

Figura 1: Grafica de la función $x^4 - 9x^2 - 5x^3 + 155x - 250$

Con un X_0 de -1, un maximo de iteraciones de 100, y una tolerancia de $10e^{-32}$ da como resultado:

Raiz $1 = 2 + 0i$ Numero de iteraciones $= 4$
Raiz $2 = 4 + 3i$ Numero de iteraciones $= 5$
Raiz $3 = -5 + 0i$ Numero de iteraciones $= 1$
Raiz $4 = 4 - 3i$ Numero de iteraciones $= 2$

1.2.2 Comparación con el algoritmo de Laguerre

Metodo de Laguerre

Debe su nombre al trabajo de Edmond Laguerre. El método es utilizado para considerar un caso especial de polinomio de grado n en el que raíz es $x = r$ y las $n - 1$ restantes es una raíz múltiple $x = q$, dicho polinomio se podría expresar así:

$$p_n(x) = (x - r)(x - q)^{n-1}$$

Si calculamos su derivada con respecto a x se tiene que

$$p'_n(x) = (x - q)^{n-1} + (n - 1)(x - r)(x - q)^{n-2} = p_n(x)(\tfrac{1}{x-r} + \tfrac{n-1}{x-q})$$

Es decir,

$$\frac{p_n'(x)}{p_n(x)} = \frac{1}{x-r} + \frac{n-1}{x-q} \tag{1}$$

Si derivamos de nuevo,

$$\frac{p_n''(x)}{p_n(x)} - \left[\frac{p_n'(x)}{p_n(x)}\right]^2 = -\frac{1}{(x-r)^2} - \frac{n-1}{(x-q)^2} \tag{2}$$

Si introducimos la notación que sigue

$$g(x) = \tfrac{p_n'(x)}{p_n(x)} \text{ y } h(x) = g^2(x) - \tfrac{p_n''(x)}{p_n(x)}$$

Las ecuaciones (1) y (2) quedan

$$g(x) = \tfrac{1}{x-r} + \tfrac{n-1}{x-q}$$

$$h(x) = \tfrac{1}{(x-r)^2} + \tfrac{n-1}{(x-q)^2}$$

Si despejamos de la primera de estas dos ecuaciones $x-q$ y lo sustituimos en la segunda, obtenemos una función cuadrática de $x-r$, la solución de la cual es la denominada fórmula de Laguerre. Tiene por expresión:

$$x - r = \frac{n}{g(x)\sqrt{(n-1)[nh(x)-g^2(x)]}}$$

El algoritmo de Laguerre para calcular las raíces de un polinomio sigue los pasos:

1. Comenzar con un número cualquiera como raíz x de $p_n(x) = 0$
2. Evaluar $p_n(x)$, $p_n'(x)$ y $p_n''(x)$ mediante la función evalpoly()
3. Calcular el valor de $g(x)$ y $h(x)$
4. Determinar una raíz mejor r con la fórmula de Laguerre (adecuando el signo correspondiente de la raíz)
5. Hacer x como r y repetir los pasos 2 a 5 hasta que $|p_n(x)| < tol$ o $|x-r| < tol$, siendo tol una precisión dada.

La convergencia del algoritmo de Laguerre es cúbica.(1)

Pruebas y Resultados

El metodo de Newton-Horner nos permite evaluar las raices de un polinomio pero en este método se presentan las raices reales. Ya el metodo de Laguerre nos permite encontrar las raices con parte imaginaria. Para este polinomio a calcular por el metodo de Newton-Horner mejorado con Laguerre nos da como resultado:

El metodo de Laguerre se establecion un X_0 de -1 y una tolerancia de $10e^{-32}$ da los siguientes resultados:

Raiz $1 = 2 + 0i$ Numero de iteraciones = 5
Raiz $2 = 4 + 3i$ Numero de iteraciones = 6

Raiz $3 = -5 + 0i$ Numero de iteraciones $= 2$
Raiz $4 = 4 - 3i$ Numero de iteraciones $= 3$

Ya el metodo de Newton-Horner se necesito 2 X_0 con el valor de -10 y -1 para calcular las 2 raices que presentaba este polinomio y una tolerancia $10e^{-32}$ da :

Raiz $1 = 2 + 0i$ Numero de iteraciones $= 6$
Raiz $2 = -5 + 0i$ Numero de iteraciones $= 10$

Disminuyendo la tolerancia a $10e^{-16}$ para los metodos anteriores genera los mismos resultados y el mismo numero de iteraciones Ya cuando se disminuye la tolerancia a $10e^{-8}$, el unico metodo que presenta cambios es el de Laguerre que genera los siguientes resultados:

Raiz $1 = 2.000000000061482 + 0i$ Numero de iteraciones $= 4$
Raiz $2 = 3.9999999999736993 + 2.9999999999992827i$ Numero de iteraciones $= 4$
Raiz $3 = -5.000000000008881 + 0i$ Numero de iteraciones $= 2$
Raiz $4 = 3.9999999999736993 - 2.9999999999992827i$ Numero de iteraciones $= 2$

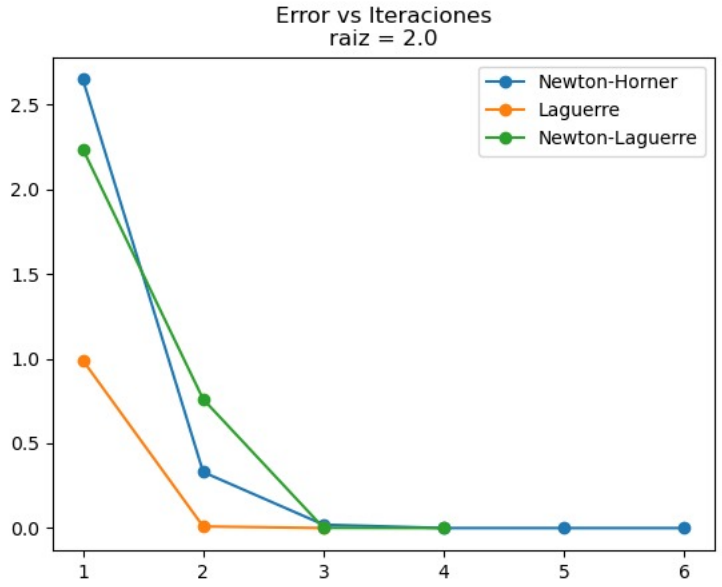

Figura 2: Comparación numero de iteraciones con diferentes metodos para hallar la raiz 2

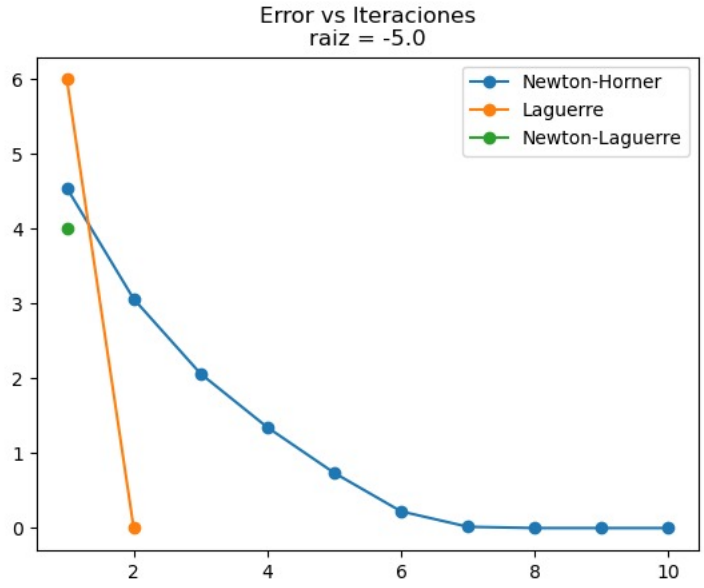

Figura 3: Comparación numero de iteraciones con diferentes metodos para hallar la raiz 2

Se probó estos metodos en la función $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$, tambien con un x_0 y una toleracia de $10e^{-16}$ genero los siguientes resultados con el metodo de Newton-Horner mejorado con Laguerre:

Raiz 1 $= 0.6666644651612263 - 3.813105375110996e^{-06}i$ Numero de iteraciones $= 35$
Raiz 2 $= 0.6666710696817929 + 2.451181113007232e^{-12}i$ Numero de iteraciones $= 1$
Raiz 3 $= 0.6666644651569807 + 3.813102923929883e^{-06}i$ Numero de iteraciones $= 2$
Ya el metodo de Laguerre genero los siguientes resultados:
Raiz 1 $= 0.6666666666666634 + 7.804971946461846e^{-08}i$ Numero de iteraciones $= 2$
Raiz 2 $= 0.6666665983157656 - 3.902486256658249e^{-08}i$ Numero de iteraciones $= 3$
Raiz 3 $= 0.666666735017571 - 3.9024856898035965e^{-08}i$ Numero de iteraciones $= 2$

2 Algoritmo Brent

2.1 Problema

Aplicar el algoritmo de Brent para encontrar las raices del polinomio: $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$.

2.2 Metodo de Brent

Este método utiliza en cada punto lo más conveniente de las estrategias del de la bisección y del de la secante. Fue formulado en 1973 por Richard Peirce Brent.

Se aplica a un intervalo $[a, b]$ en cuyos extremos la función adopta signos distintos. Sigue la pista a un punto x_i, que es el mejor en el sentido del error hacia atrás, y un intervalo $[a_i, b_i]$ para la raíz. Aplica una interpolación cuadrática inversa (no $y = p(x)$ sino $x = p(y)$) a los tres puntos $(f(x_i), x_i)$, $(f(a_i), a_i)$ y $(f(b_i), b_i)$ con el fin de reemplazar uno de ellos con aquel- único- donde $x = p(y = 0)$. Si el error hacia atrás mejora y el intervalo de confinamiento de la solución se reduce al menos la mitad, el punto obtenido reemplaza a uno de los tres vigentes. Si no se cumplen esos requisitos se intenta el método de la secante con el mismo objetivo. Si también falla, se realiza un paso del método de la bisección.

2.3 Pruebas y Resultados

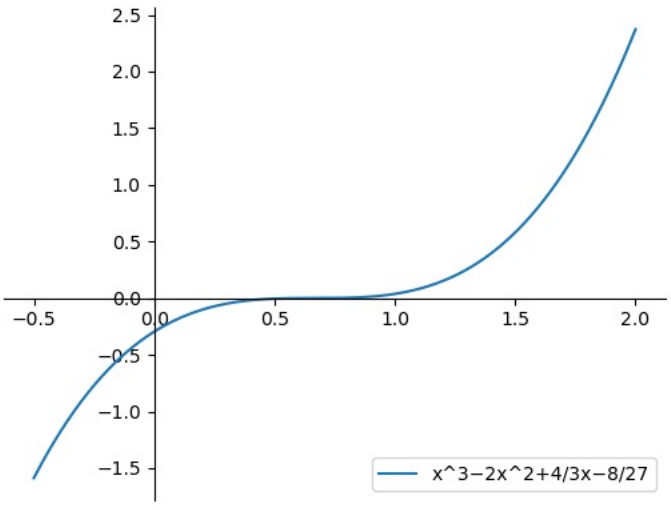

Figura 4: Grafica de la función $x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$

Se utilizo el metodo en un rango de $[0, 1]$, con una tolerancia de $10e^{-32}$ y dio el resultado de 0.6666685708874426 con 44 iteraciones. Se generan los mismos resultados al bajar la tolerancia a $10e^{-16}$ y $10e^{-8}$

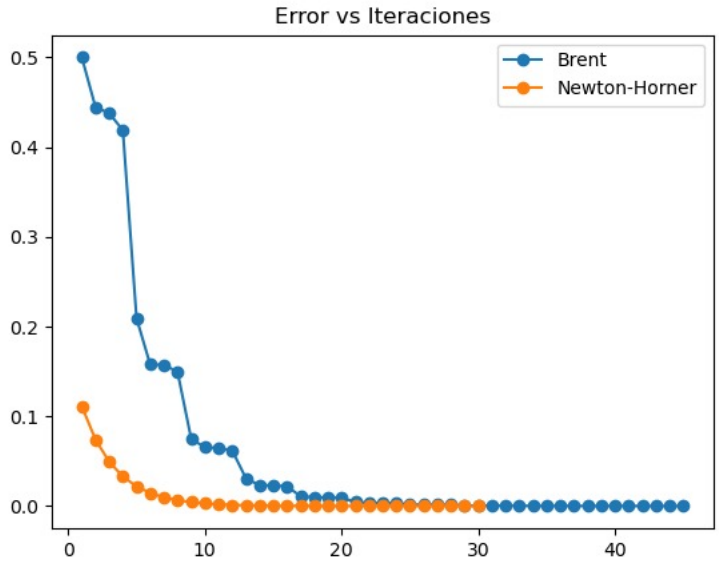

Figura 5: Grafica de la función $x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$

Se probo este metodo con la función $x^4 - 9x^2 - 5x^3 + 155x - 250$, con un rango de $[-10, 0]$, con una tolerancia de $10e^{-32}$ y dio el resultado de -5.0 con 25 iteraciones. Se generan diferentes resultados al bajar la tolerancia a $10e^{-16}$ y $10e^{-8}$. Con la primera da un valor de -5.000000000000001 con 25 iteraciones y ya con la segunda tolerancia es -4.9999999994892965 con 18 iteraciones.

3 Optima Aproximación Polinomica

3.1 Problema

Aplicar esta técnica de aproximación polinómica, para $f(x) = e^{sinx - cosx^2}$ en el intervalo $[-2^{-8}; 2^{-8}]$ con una precisión deseada doble-doble y un error menor de 2^{-90} y comparar con la aproximación de Taylor.
Para comprender de forma adecuada la implementación presentada primero revisaremos ambos métodos, el de Remez y la aproximación por Taylor para así formar un critero que nos permita determinar si es adecuado utilizar uno o el otro, también en qué casos conviene más utilizar Taylor o Remez.

3.2 Remez

Este algoritmo es muchas veces preferido ya que teóricamente es el más precisio cuando se tiene un dominio determinado. Pro la naturaleza de este algoritmo, se posee una convergencia cuadrática. El algoritmo de Remez es un procedimiento iterativo basado en el teorema de la alternancia (la cual puede encontrar la mejor aproximación min max de una función continua). En otras palabras este método calcula el polinomo iterativo $p^* \in P_n$, para $f \in C[a,b]$. La implementación puede ser encontrada como anexo con el nombre Remez.R

3.2.1 Entradas

Para poder hacer correcto uso de este algoritmo se deben definir valores del entrada que permitan al algoritmo generar la salida esperada, entre estos encontramos:
$f \in C[a,b]$, referencia inicial para el Sistema de Ecuaciones $x_0, ..., x_n$ donde $a \leq x_0 < ... < x_n \leq b$ siendo a y b los límites en los que se desea evaluar la función. También es necesario contar con un constante δ como criterio de parada.

3.2.2 Paso 1

Resolver el sistema de ecuaciones lineales:
$b_0 + b_1 x_i + ... + b_n(x_i^n)$, donde $i = 1, 2, ..., n+2$.

3.2.3 Paso 2

Teniendo ahora los coeficientes obtenidos en el paso inmediatamente anterior se genera el polinomio según como fue especificado en (2). Para nuestro caso, como es de nuestro interés buscar un error menor a 2^-90 entonces la base monomial será x^0, x^4, x^8, x^{12}. De tal forma obtendríamos el nuevo polinomio $P(x) = b_0x^0 + b_1x^4 + b_2x^8 + b_3x^{12}$.

3.2.4 Criterio de parada

Para tener la certeza de que se deben concluir las iteraciones es necesario encontrar $y \in [a,b]$ (recordemos que a y b es el intervalo en que se debe evaluar la función) que cumpla $|f(y) - P_i(y)| = max_(x \in [a,b])|f(x) - P_i(x)|$ y las iteraciones son interrumpidas cuando $|f(y) - P_i(y)| - |f(x_0) - P_i(x_0)| < \delta$.

3.2.5 Paso 3

Se definen $z_0 = a$ y $z_n + 1 = b$ y se debe buscar una raiz z_i en $x_(i-1), x_i$ para todo $i = 1, ..., n$

3.2.6 Paso 4

Para todo $i = 1, ..., n$ se tiene que buscar $y_i \in [z_i, z_i + 1]$ donde $|f(x_i) - P_i(x_i)| * |f(y) - P_i(y)|$ es el máximo posible, con esta información luego reemplazamos en los x definidos como entrada.

3.3 Resultados

Definida la función y en el intervalo donde se evaluará, la aproximación se comportó de la siguiente manera:

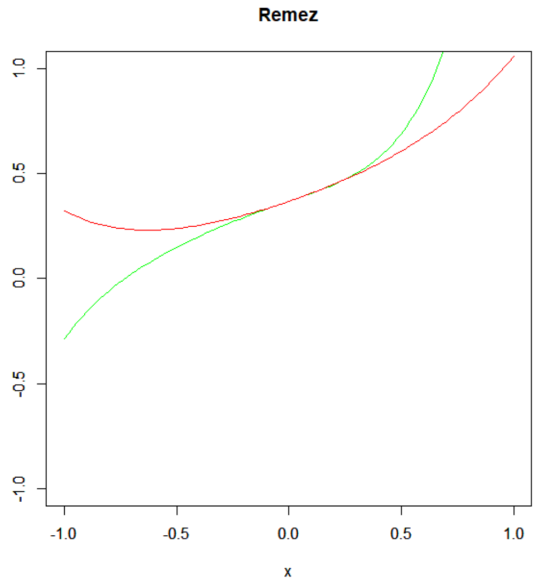

Figura 6: Aproximación utilizando Remez

Donde la gráfica verde representa la función y la curva roja la aproximación. Podemos ver que como fue retratado en los documentos de referencia este método es preciso en un intervalo definido, como se evalúan puntos fuera de este la estimación es pobre, como por ejemplo en el en intervalo menor a -0.5 la aproximación es catastrófica. Realizamos el cálculo de los erros y fue posible recurar que: el error relativo fue $1.116721e-20$ y el error absoluto $4.454435e-17$, siendo de cualquier manera magnitudes relevantes. A través de nuestra investigación encontramos que el comportamiento de su convergencia puede ser alterado[3], utilizando los conceptos propuestos por La Vallé Poussin es posible utilizar su teorema para generar restricciones que nos permitan modificar el algoritmo de Remez para obtener una convergencia lineal(3).

3.4 Aproximación usando Taylor

Recordemos que la series series de Taylor vienen definidas de la siguiente manera $f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f'(a)}{2!}(x-a)^2 \frac{f(a)}{3!}(xa)^3 + ...$
Tales calculos extensivos y los cocientes pueden inducir errores de magnitudes relevantes. Algunos datos importantes a tener en cuenta cuando se hace una comparación entre Remez y Taylor es que como vimos, en Remez no hace falta evaluar todos los términos, para el caso del ejercicio fue suficiente evaluar solo $x^0, x^4, x^8, x^(12)$ y en Taylor es necesario evaluar según el grado del polinomio. También es importante saber que la serie podría o no converger (existen casos donde diverge).

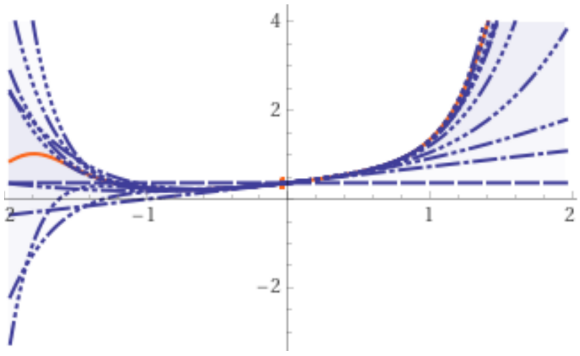

Figura 7: Aproximación utilizando Taylor

Aunque el resultado es preciso, es necesiario reconocer que la cantidad de cálculos necesarios lo vuelven un método exahustivo para la máquina.

4 Conclusiones

Despues de un largo trabajo se logró percibir la forma en que un problema puede ser resuelto de distintas formas, en este caso el problema de hallar las raices de una función o polinomio y como este puede ser resuelto por distintos algoritmos. También se logró evidenciar como cada algoritmo presenta su propia aproximación a la solución real y como cada una de estas soluciones puede ser de utilidad en distintos contextos, ya sea por la precisión de la solución o por la eficiencia en la que se llega a la misma.

De forma concreta, se evidenció como el algoritmo propuesto de Newton-Horner combinado con el algoritmo de Laguerre se impuso sobre los demás dadas sus caracteristicas, ya que este obtuvo los resultados de forma mas eficiente, requiriendo de solo una pequeña cantidad de iteraciones para poder entregar el resultado final.

Por ultimo, quedó en evidencia la dificultad de un computador para poder representar con precisión numeros demasiado pequeños, lo que afecta de forma directa el desarrollo de los algoritmos aquí presentados, alterando la precisión

de sus resultados, como el caso del polinomio evaluado durante del desarrollo del algoritmo de Brent.

References

[1] J. L. de la Fuente O'Connor, *Ingeniería de losAlgoritmos y Métodos Numéricos*. Editorial Círculo Rojo, 2017.

[2] F. D. Dinechin and C. Q. Lauter, "Optimizing polynomials for floating-point implementation," 2008.

[3] E. de Groot, "Finding best minimax approximations with the remez algorithm," 2017.