Pontificia Universidad Javeriana
Departamento de Matemática
Análisis Numérico

Reto 1

Ejercicios realizados
por

Laura Mariana Jiménez Jiménez
la.jimenez@javeriana.edu.co

Paula Valentina Sanchez Peña
pa-sanchez@javeriana.edu.co

Sebastián Gutiérrez Zambrano
sebastian_gutierrez@javeriana.edu.co
Profesora: Eddy Herrera Daza

13 de septiembre de 2020

1. Evaluación de las raíces de un Polinomio

1.1. Explicación de métodos

1.1.1. Método de Horner

Definición

El método de Horner es un algoritmo iterativo que permite evaluar polinomios de forma monomial. El algoritmo evalúa el polinomio con más eficiencia. Si se evalúa normalmente el polinomio, se requerirían n sumas y $(n2+n)/2$ multiplicaciones como minimo. Mientras que utilizando el algoritmo de Horner se requerirían simplemente n sumas y n multiplicaciones.

Procedimiento

En este reto se utilizó el método de Horner para evaluar polinomios en ciertos puntos. Los polinomios que se evaluavan eran los que se obtenían de derivar un función. A continuación un ejemplo de funcionamiento del algoritmo.

- Se empieza con el siguiente polinomio: $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n$
  Donde se sabe que $a_0, a_1 \cdots$ son constantes.
- Se quiere evaluar el polinomio en un x_0
- Se reescribe el polinomio de la siguente forma: $p(x) = a_0 + x(a_1 + x(a_2 + \cdots x(a_{n-1} + a_nx)\cdots))$
- Se reconocen las siguientes constantes, donde
  $b_n := a_n$
  $b_{n-1} := a_{n-1} + b_nx_0$
  .
  .
  .
  $b_0 := a_0 + b_1x_0$
- Finalmente se sabe que b_0 es el valor del polinomio evaluado en x_0
- Se empieza a sustituir b_i iterativamente hasta llegar a b_0

1.1.2. Método de Newton

Definición

Método de Newton-Raphson es una eficiente técnica para encontrar las raíces de una función. Como se usa en los cálculos diferenciales, este se basa en una aproximación lineal la cual describe que en una función $f(x)$ con r una raíz de la ecuación $f(x) = 0$ se puede estimar un valor aproximado a r empezando desde un valor inicial x_0 y que con este se puede estimar x_1 tal que este es un valor aún más aproximado al valor de r y, a partir de este x_1 se puede estimar un x_2 todavía más cercano a r y se continua de esta manera hasta que se llegue a un valor estimado lo suficientemente cercano a r. Lo anterior es un método iterativo que usa el método de Newton-Raphson.

El método dice que se parte de un x_0 el cual es un valor estimado de r en donde $r = x_0 + h$. Dado que el valor verdadero es r y $h = r - x_0$, el número h mide qué tan lejano se encuentra el valor estimado x_0 del valor verdadero.

Dado que h es un valor muy "pequeño"se puede usar la aproximación lineal por tanto se concluye que

$$0 = f(r) = f(x_0 + h) \approx f(x_0) + hf'(x_0) \tag{1}$$

y a menos de que $f'(x_0)$ sea muy cercano a 0

$$h = -\frac{f(x_0)}{f'(x_0)} \tag{2}$$

Con esto se puede reemplazar h en la expresión que se mencionó anteriormente:

$$r = x_0 + h \approx x_0 - \frac{f(x_0)}{f'(x_0)} \tag{3}$$

De esta manera si hay un nuevo valor estimado x_1 que se acerca aún más a r, este es expresado de esta manera:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{4}$$

Y de esta manera se puede decir que el valor estimado x_{n+1} es calculado a partir del valor estimado actual x_n lo cual finalmente el método de Newton-Raphson se expresa de la siguiente manera:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{5}$$

1.1.3. Método de Laguerre

Definición

Es un método algebraico que sirve para evaluar ecuaciones polinomiales. Es un método iterativo, que permite calcular las raíces reales y complejas de cualquier polinomio de grado n. Este método es iterativo. Su orden de convergencia es cúbico con raices de multiplicidad unitaria (por ejemplo: $p(x) = x^3 + 2x^2 - 7x + 4$ este polinomio para su raiz de -4 sería unitaria porque se puede expresar de la siguente manera $p(x) = (x+4)(x-1)^2$). Cuando la multiplicaida es otra, el orden de convergencia puede disminuir.

Procedimiento

- Se empieza con el siguiente polinomio
  $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_nx^n$
  Al cual se le quiere hallar sus raíces o por lo menos acotarlas.

- Se tiene la fórmula Laguerre:

$$x - r = \frac{n}{g(x) \pm \sqrt{(n-1)\left[nh(x) - g^2(x)\right]}} \tag{6}$$

Donde se sabe que:

$$g(x) = \frac{1}{x-r} + \frac{n-1}{x-q} \tag{7}$$

$$h(x) = \frac{1}{(x-r)^2} + \frac{n-1}{(x-q)^2} \tag{8}$$

- Se considera que este polinomio de grado n se puede expresar de la siguiente manera $P_n(x) = (x-r)(x-q)^(n-1)$.
- Se escoge un x inicial.
- Se evalúa $P_n(x)$, $P'_n(x)$, $P''_n(x)$.
- Se calcula el valor de $g(x)$ y $h(x)$ de acurdo a las fórmulas antes dadas.
- Se determina una raíz mejor haciendo uso de la ecuacion 1.
- Se igualar x a r y se repiten los pasos hasta que $P'_n(x) < tolerancia$ o $x - r < tolerancia$

Para el manejo del algoritmo se utilizó la función Laguerre que de acuerdo al valor inicial que se le ponga, retorna una de las varias raices que pueda tener el polinomio.

1.2. Resultados

Polinomio $x^4 - 9 * x^2 - 5 * x^3 + 155 * x - 250$

A continuación la gráfica del polinomio. Realizada en R. Se puede ver las diferentes raíces que este tiene. Raíces que se espera encontrar.

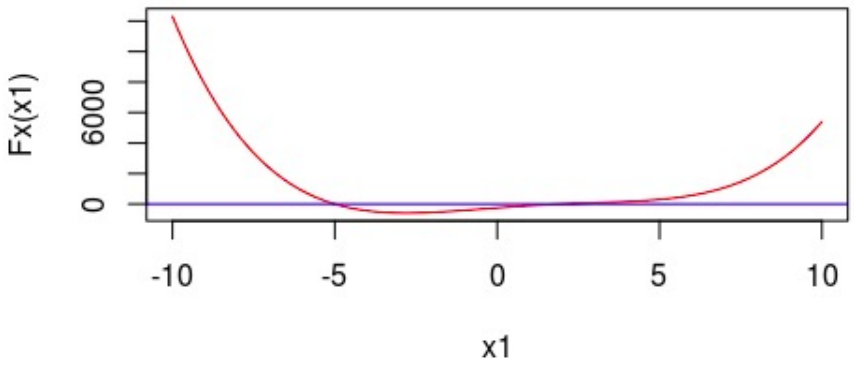


Fig 1. gráfica polinomio R

A continuación la gráfica realizada con Wolfram

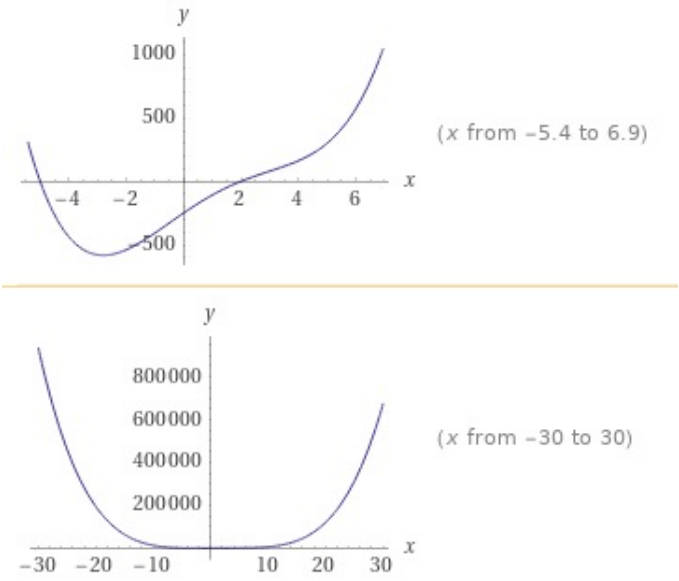


Fig 2. gráfica polinomio wolfram

Tolerancia $1.e^{-}8$

Primera raíz real

Se tomó como x inicial 3.

```
I= 1    X= 3    E= 1.081081
I= 2    X= 1.918919     E= 0.07999346
I= 3    X= 1.998912     E= 0.001087422
I= 4    X= 2    E= 1.95e-07
I= 5    X= 2    E= 5.995204e-15
Iteraciones: 5 Raiz[1] 2
El resultado con Laguerre es:  2
```

Fig 3. Resultados

Se puede observar que el algoritmo que emplea Newton realiza 5 iteraciones y llega al mismo resultado del algortimo de Laguerre. A continuación se moestrará la gráfica del error versus el número de iteraciones para este algoritmo de Newtor-Horner.

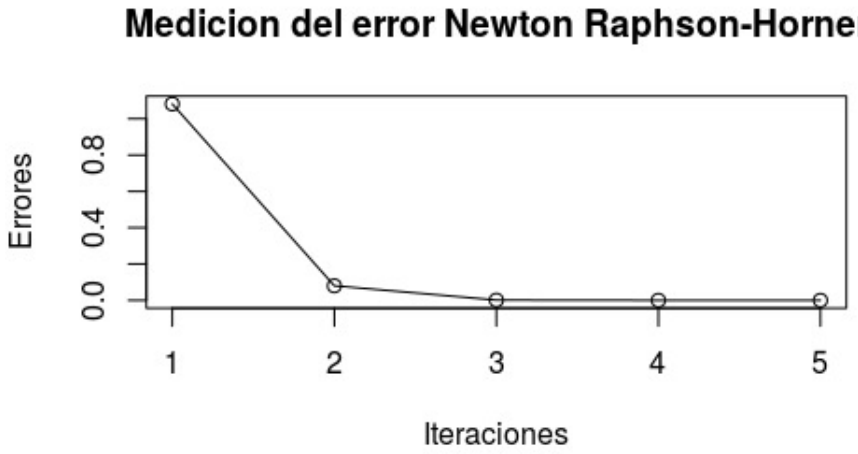


Fig 4. Gráfica error vs num iteraciones

Segunda raíz real
Se tomó como x inicial -10. Y se evidencia que el valor dado por Laguerre y Newton es el mismo.

```
I= 1    X= -10  E= 2.381413
I= 2    X= -7.618587    E= 1.545047
I= 3    X= -6.07354     E= 0.8079338
I= 4    X= -5.265606    E= 0.2440225
I= 5    X= -5.021583    E= 0.02142519
I= 6    X= -5.000158    E= 0.0001581578
I= 7    X= -5   E= 8.576509e-09
  El resultado con Laguerre es:  -5
```

Fig 5. Resultados

Gráfica del error para este caso. El error se comporta igual que en la anteriror tendiendo a 0 cada vez que realiza una iteración.

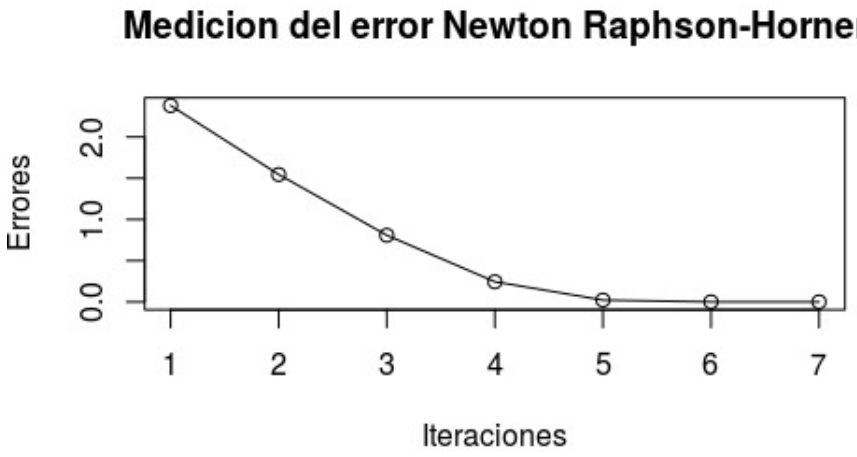


Fig 6. Gráfica error vs num iteraciones

Primera raíz imaginaria
Se tomó como x inicial 7-5i. Y se evidencia que el valor dado por Laguerre y Newton es el mismo.

```
I= 1    X= 7    E= 1.688861
I= 2    X= 5.736821     E= 1.133121
I= 3    X= 4.832835     E= 0.6486868
I= 4    X= 4.24101      E= 0.2405959
I= 5    X= 4.003788     E= 0.03023231
I= 6    X= 3.999675     E= 0.000453621
I= 7    X= 4    E= 1.028891e-07
I= 8    X= 4    E= 5.329071e-15
Iteraciones: 8 Raiz[1] 4-3i
El resultado con Laguerre es:  4-3i
```

Fig 7. Resultados

Gráfica del error para este caso. El error se comporta igual que en la anteriror tendiendo a 0 cada vez que realiza una iteración.

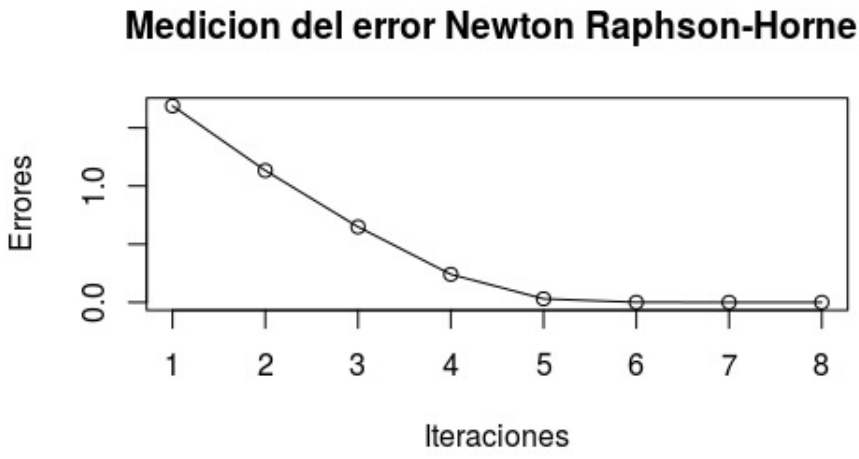


Fig 8. Gráfica error vs num iteraciones

Segunda raíz imaginaria

Se tomó como x inicial 7+5i. Y se evidencia que el valor dado por Laguerre y Newton es el mismo.

```
I= 1    X= 7    E= 1.688861
I= 2    X= 5.736821     E= 1.133121
I= 3    X= 4.832835     E= 0.6486868
I= 4    X= 4.24101      E= 0.2405959
I= 5    X= 4.003788     E= 0.03023231
I= 6    X= 3.999675     E= 0.000453621
I= 7    X= 4    E= 1.028891e-07
I= 8    X= 4    E= 5.329071e-15
Iteraciones: 8 Raiz[1] 4+3i
El resultado con Laguerre es:  4+3i
```

Fig 9. Resultados

Gráfica del error para este caso. El error se comporta igual que en la anteriror tendiendo a 0 cada vez que realiza una iteración.

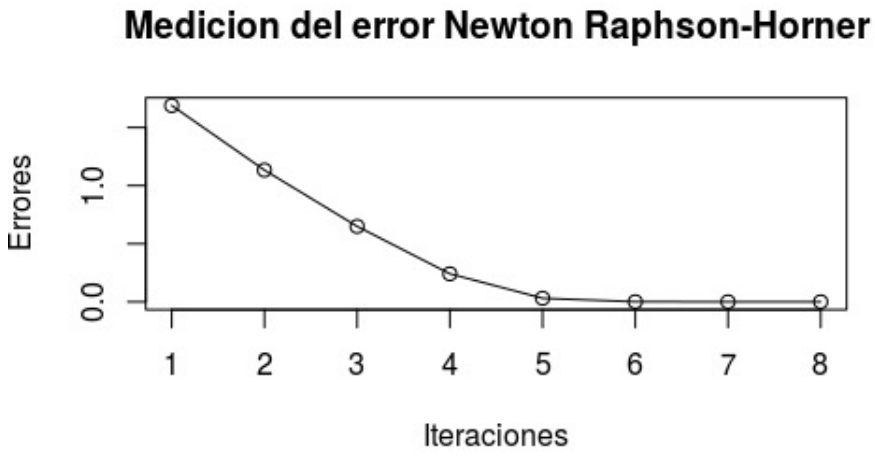


Fig 10. Gráfica error vs num iteraciones

Tolerancia $1.e^{-}16$
Primera raíz real

Los valores iniciales igualmente se mantuvieron cone sta neuva tolerancia, sin embargo se puede ver que con respecto a la tolerancia anterior, el numero de iteraciones se mantuvo, esto debia a que el error se iba acercado mucho a la tolerancia y ahí paraba de iterar. Sin embargo se mantuvo correcto el resultado

```
I= 1    X= 3    E= 1.081081
I= 2    X= 1.918919     E= 0.07999346
I= 3    X= 1.998912     E= 0.001087422
I= 4    X= 2    E= 1.95e-07
I= 5    X= 2    E= 5.995204e-15
I= 6    X= 2    E= 0
Iteraciones: 6 Raiz[1] 2
El resultado con Laguerre es:  2
```

Fig 11. Resultados

Para este caso se observa que el número de iteraciones solo aumentó en uno y el resultado sigue siendo el mismo, así como también se mantiene la gráfica de eroor versus iteraciones.

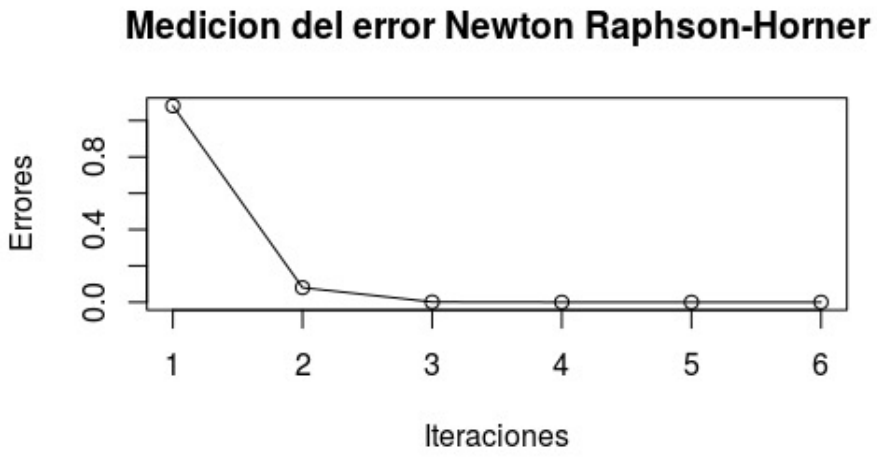


Fig 12. Gráfica error vs num iteraciones

Segunda raíz real
En este caso el número de iteraciones también aumenta solo en uno.

```
I= 1    X= -10  E= 2.381413
I= 2    X= -7.618587    E= 1.545047
I= 3    X= -6.07354     E= 0.8079338
I= 4    X= -5.265606    E= 0.2440225
I= 5    X= -5.021583    E= 0.02142519
I= 6    X= -5.000158    E= 0.0001581578
I= 7    X= -5   E= 8.576509e-09
I= 8    X= -5   E= 0
Iteraciones: 8 Raiz[1] -5
El resultado con Laguerre es:  -5
```

Fig 13. Resultados

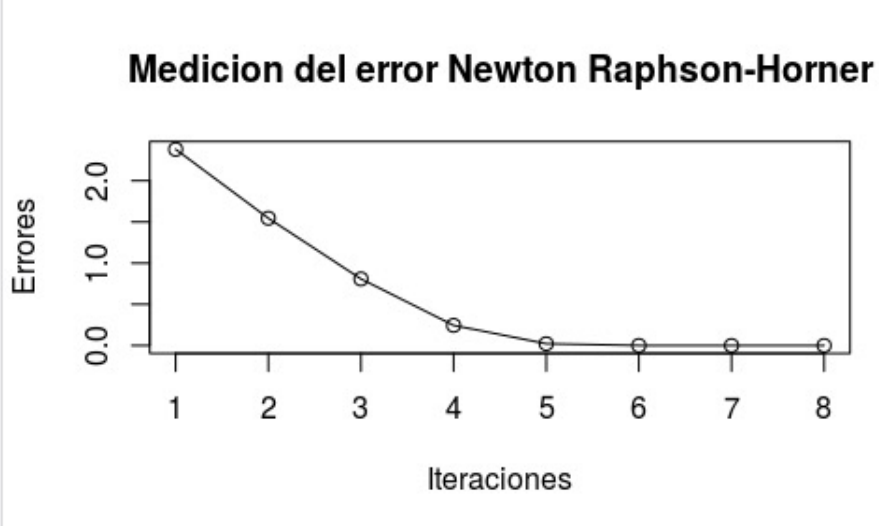


Fig 14. Gráfica error vs num iteraciones

Primera raíz imaginaria

El resultado es el esperado, aumenta en uno el número de iteraciones, y el s igual al resultado dado por el algoritmo de Laguerre.

```
El resultado con Laguerre es:  -5
I= 1    X= 7    E= 1.688861
I= 2    X= 5.736821     E= 1.133121
I= 3    X= 4.832835     E= 0.6486868
I= 4    X= 4.24101      E= 0.2405959
I= 5    X= 4.003788     E= 0.03023231
I= 6    X= 3.999675     E= 0.000453621
I= 7    X= 4    E= 1.028891e-07
I= 8    X= 4    E= 5.329071e-15
I= 9    X= 4    E= 0
Iteraciones: 9 Raiz[1] 4-3i
El resultado con Laguerre es:  4-3i
```

Fig 15. Resultados

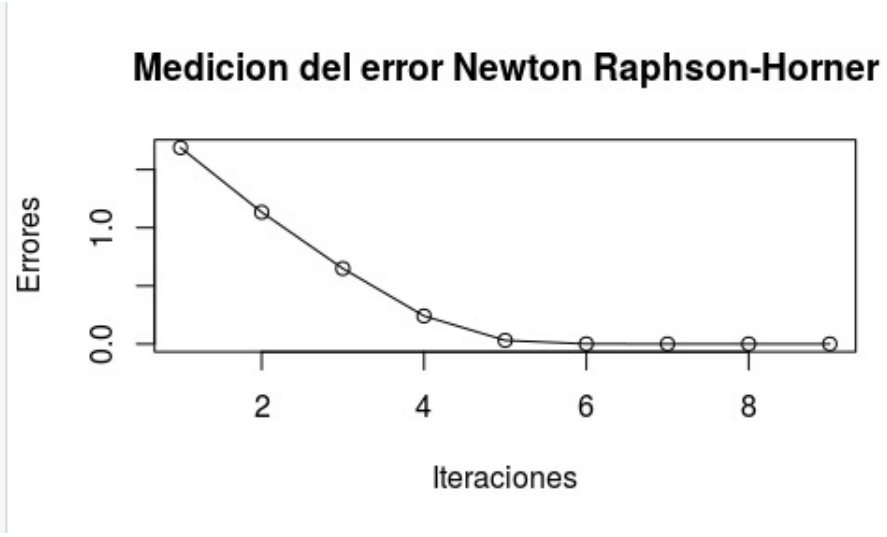


Fig 16. Gráfica error vs num iteraciones

Segunda raíz imaginaria
Para este el numero de iteraciones también aumenta en uno.

```
I= 1    X= 7    E= 1.688861
I= 2    X= 5.736821     E= 1.133121
I= 3    X= 4.832835     E= 0.6486868
I= 4    X= 4.24101      E= 0.2405959
I= 5    X= 4.003788     E= 0.03023231
I= 6    X= 3.999675     E= 0.000453621
I= 7    X= 4    E= 1.028891e-07
I= 8    X= 4    E= 5.329071e-15
I= 9    X= 4    E= 0
Iteraciones: 9 Raiz[1] 4+3i
El resultado con Laguerre es:  4+3i
```

Fig 17. Resultados

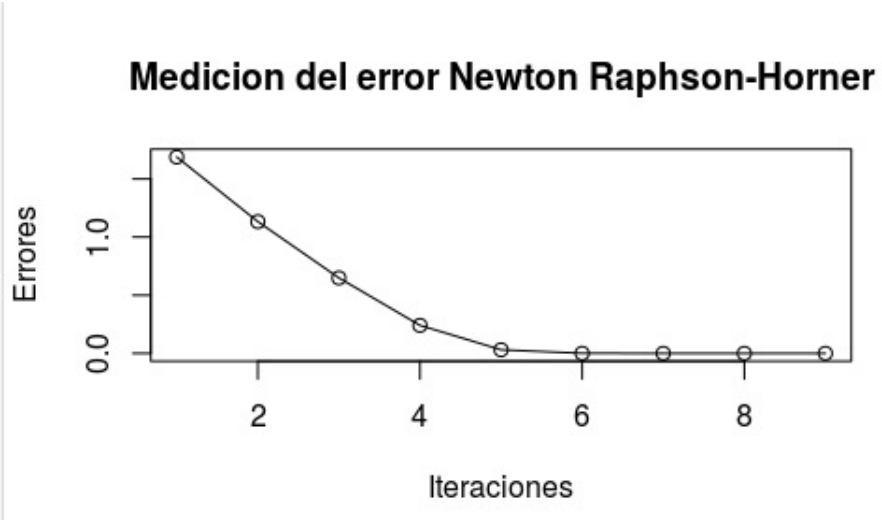


Fig 18. Gráfica error vs num iteraciones

Tolerancia $1.e^{-}32$
Primera raíz real

```
I= 1    X= 3    E= 1.081081
I= 2    X= 1.918919     E= 0.07999346
I= 3    X= 1.998912     E= 0.001087422
I= 4    X= 2    E= 1.95e-07
I= 5    X= 2    E= 5.995204e-15
I= 6    X= 2    E= 0
Iteraciones: 6 Raiz[1] 2
```

Fig 19. Resultados

Para este caso se observa que el número de iteraciones solo aumentó en uno y el resultado sigue siendo el mismo, así como también se mantiene la gráfica de eroor versus iteraciones.

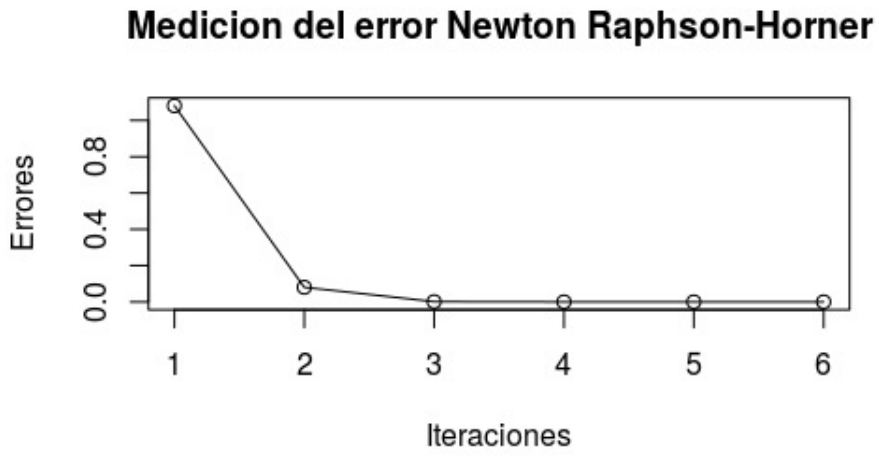


Fig 20. Gráfica error vs num iteraciones

Segunda raíz real
Para este el numero de iteraciones se mantiene igual que con la tolerancia de $1.e^{-}16$

```
I= 1    X= -10  E= 2.381413
I= 2    X= -7.618587    E= 1.545047
I= 3    X= -6.07354     E= 0.8079338
I= 4    X= -5.265606    E= 0.2440225
I= 5    X= -5.021583    E= 0.02142519
I= 6    X= -5.000158    E= 0.0001581578
I= 7    X= -5   E= 8.576509e-09
I= 8    X= -5   E= 0
Iteraciones: 8 Raiz[1] -5
```

Fig 21. Resultados

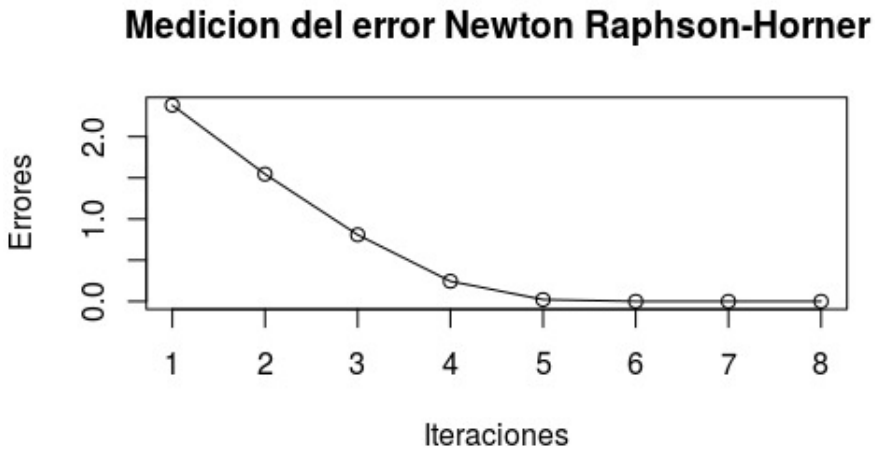


Fig 22. Gráfica error vs num iteraciones

Primera raíz imaginaria

COmo los ejercicios anteriores el resultado es el esperado sin embargo el numero de iteraciones se mantiene igual a la tolerancia anterior.

```
El resultado con Laguerre es:  -5
I= 1    X= 7    E= 1.688861
I= 2    X= 5.736821     E= 1.133121
I= 3    X= 4.832835     E= 0.6486868
I= 4    X= 4.24101      E= 0.2405959
I= 5    X= 4.003788     E= 0.03023231
I= 6    X= 3.999675     E= 0.000453621
I= 7    X= 4    E= 1.028891e-07
I= 8    X= 4    E= 5.329071e-15
I= 9    X= 4    E= 0
Iteraciones: 9 Raiz[1] 4-3i
El resultado con Laguerre es:  4-3i
```

Fig 23. Resultados

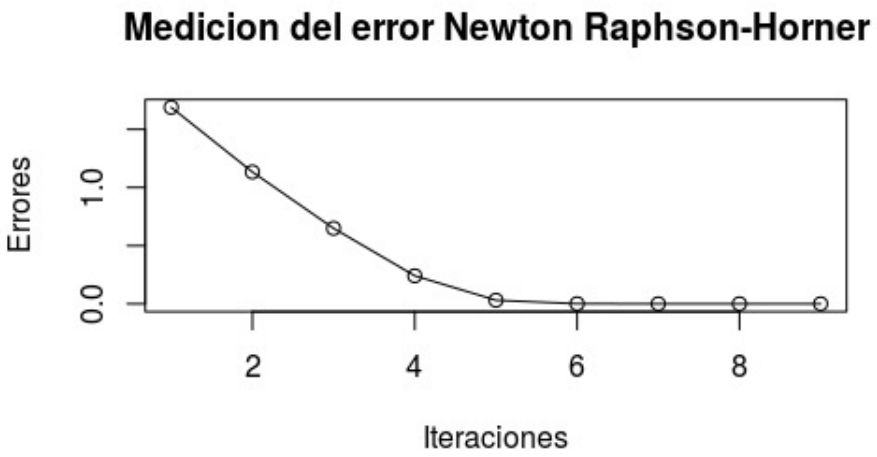


Fig 24. Gráfica error vs num iteraciones

Segunda raíz imaginaria

```
I= 1    X= 7    E= 1.688861
I= 2    X= 5.736821     E= 1.133121
I= 3    X= 4.832835     E= 0.6486868
I= 4    X= 4.24101      E= 0.2405959
I= 5    X= 4.003788     E= 0.03023231
I= 6    X= 3.999675     E= 0.000453621
I= 7    X= 4    E= 1.028891e-07
I= 8    X= 4    E= 5.329071e-15
I= 9    X= 4    E= 0
Iteraciones: 9 Raiz[1] 4+3i
El resultado con Laguerre es:  4+3i
```

Fig 25. Resultados

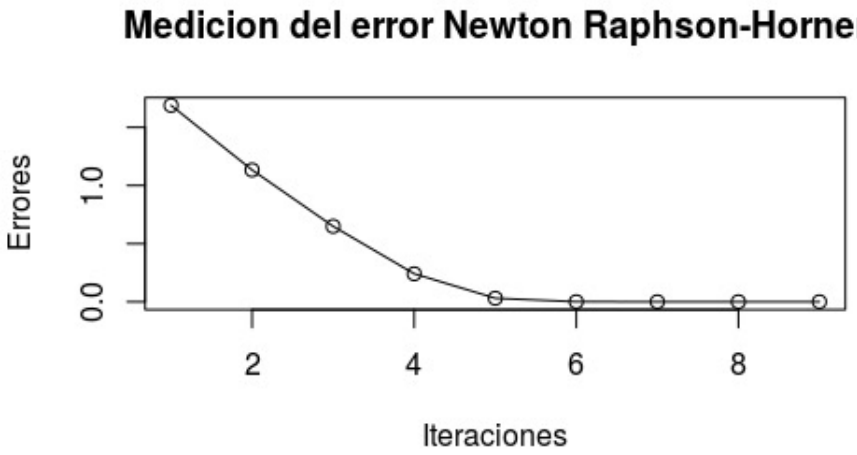


Fig 26. Gráfica error vs num iteraciones

A continuación se comparará estos resultados las raices que calcula wolfram. Se observa que los resultados son correctos.

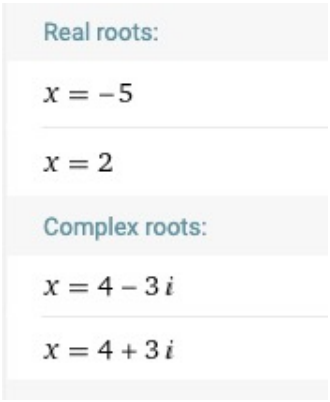


Fig 26. raices del polinomio

Conclusiones

Se pudo notar que si es más eficiente usar el algoritmo de Newton en vez del de Laguerre. Sin embargo habría que evaluar que tánto lo es dependiendo al caso que se tenga pues finalmente solo se reduce una iteración dependiendo el caso. No es un cambio muy grande u importante.

2. Algoritmo Brent

2.1. Explicación del algoritmo

El método de Brent es un método mejorado del algoritmo de Dekker. Este algoritmo combina el método de bisección , el método de la secante y la interpolación cuadrática inversa para hallar las

raices de un polinomio. Debido a esto, para entender el algoritmo de Brent es necesario entender el de Dekker(algoritmo iterativo).

Algortimo de Dekker

El algoritmo de Dekker a su vez es derivado del método de bisección, por lo tanto, el polinomio a evaluar debe cumplir con las mismas caracteristicas.

- Se va a hallar la raiz para un intervalo [a,b]
- f(a) y f(b) debe tener signos opuestos
- f debe ser continua en el intervalo [a,b]
- Gracias al teorema del valor medio, se debe garantizar la existencia de una raíz en este intervalo.

A la hora de realizar la iteración se deben tener claros los métodos de la secante y del punto medio. Primero se realiza la iteración con el método de la secante de usando la ecuacion 9.

$$s = \begin{cases} b_k - \frac{b_k - b_{k-1}}{f(b_k) - f(b_{k-1})} f(b_k), & \text{si} f(b_k) \neq f(b_{k-1}) \\ m & \text{de lo contrario} \end{cases} \tag{9}$$

Entonces, se pasa a la siguiente iteración ($b_{k+1} = s$) si el resultado de esta ecuación esta entre b_k y m. Si no es el caso, entonces se itera con el algoritmo del punto medio ($b_{k+1} = m$) usando la ecuación 10.

$$m = \frac{a_k + b_k}{2} \tag{10}$$

Luego de haber realizado estos calculos, es hora de cambiar el intervalo pero con mucho cuidado porque f(a_{k+1}) debe tener signo opuesto con f(b_{k+1}). Para asegurarse de esto se verifica si con el a_k se cumple esto, si si se cumple $a_{k+1} = a_k$. Pero en caso de que no, $a_{k+1} = b_k$. Otra cosa puede pasar y es que, si son signos opuestos pero $f(a_{k+1}) < f(b_{k+1})$. En este caso es más simple, ya que simplemente se intercambian los valores de a_{k+1} y b_{k+1}. Y esta sería una iteración del método de Dekker.

El problema con el método de Dekker es que puede pasar que en todas sus iteraciones emplee el algoritmo de la secante, pero a diferencia de este algoritmo, las iteraciones de Dekker irían aumentando muy poco por lo que se demoraría más iteraciones que el algoritmo de la secante. Brent quiso resolver este problema. Así Brent asegura que siempre en la k-esima iteración se realice bisección porque pone condiciones para que esto suceda. Este algoritmo está demostrado que requiere como máximo n^2 iteraciones.

2.2. Resultados

En esta sección se exponen los reusltados de las pruebas realizadas con el algoritmo de Brent para una función con diferentes tolerancias y presiciones. Fue necesario cambiar la presicion para que se pueda medir con tolerancias muy pequeñas.

$x^3 - 2 * x^2 + (4/3) * x - (8/27)$

Primero es importante graficar el polinomio, reconocer que tenga raices y compararlo con la gráfica que da Wolfram para asegurarse que se está haciendo un bueno uso de R.

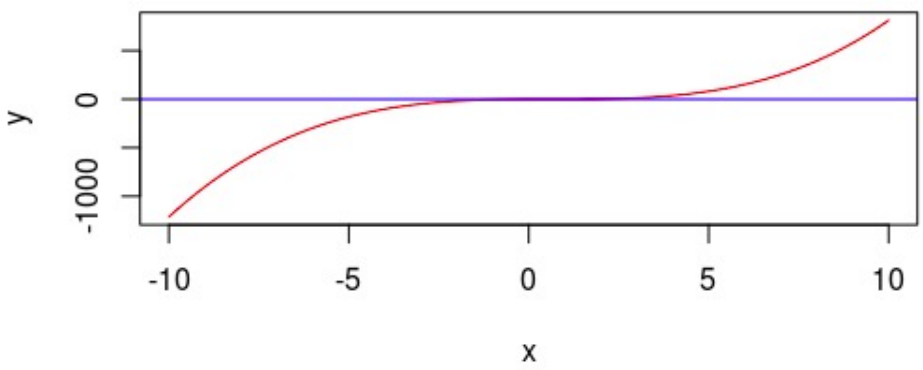


Fig 27. Función polinomio$x^3 - 2 * x^2 + (4/3) * x - (8/27)$

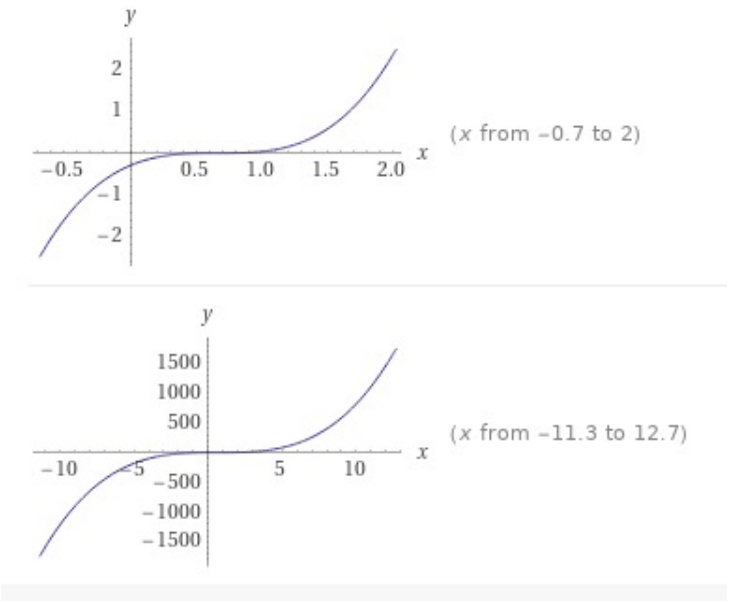


Fig 28. Función polinomio$x^3 - 2 * x^2 + (4/3) * x - (8/27)$ en Wolfram

Tolerancia de $1.e-16$

Con la tolerancia de $1.e-16$ fue necesario utlizar una presicion doble. La raíz utlizando dicha presicion fue de 0.6666660308837891 y se realizaron 20 iteraciones.

```
"Raiz:"
0.6666660308837891
"Número de iteraciones"
20
```

Fig 29. resultados

Estos resultados ayudaron a realizar la gráfica siguiente de error versus numero de iteraciones, donde ser ve a medida que van aumentando las iteraciones, el error va disminuyendo.

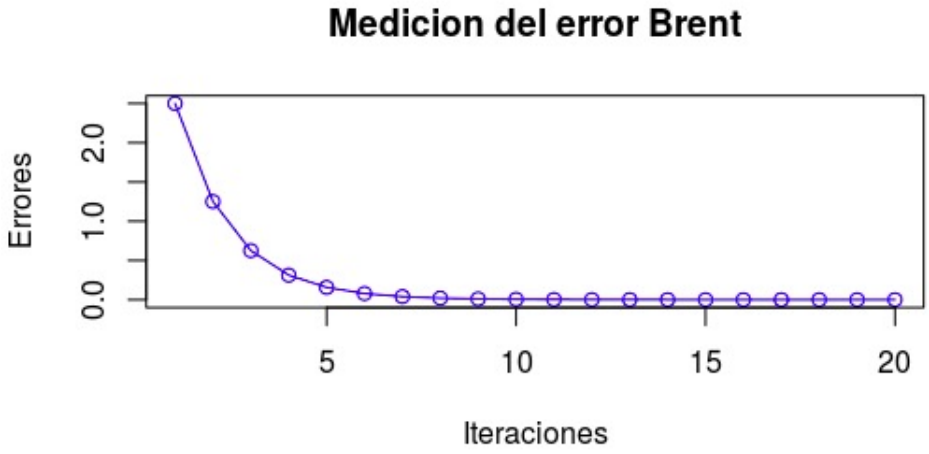


Fig 30. grafica error vs num iteraciones Brent

Al ser el método de Dekker un método derivado del de bisección, parece pertinente hacer la comparación, y evaluar si realmente es mejor este mñetodo. Una ventaja del método que se tiene de bisección serí que logra calcular más de una raíz, sin embargo como se ve en los resultados, ocupa muchas más iteraciones que el método de Dekker, mas de el doble.

```
 las raíces son:
  0.6666641 y  1.638489
con error <= 5.551115e-17
 El numero de iteraciones es:  55
```

Fig 31. resultados bisección

Tolerancia de $1.e-32$

Con esta tolerancia, la presicion tuvo que cambiar a 128 bits. Y con dicha presicion el resultado, como se esperaba, tine más decimales.
0.6666698708192892000817759411253026758172
También cabe destacar que el algoritmo necesitó de 55 iteraciones.

```
[1] "Raiz:"
1 'mpfr' number of precision  128   bits
[1] 0.66666987081928920008177594112530267581 72
[1] "Número de iteraciones"
1 'mpfr' number of precision  128   bits
[1] 55
```

Fig 32. resultados

Con estos resultados se realizó una gráfica de error versus numero de iteraciones donde se evidencia el mismo fenómeno del ejercicio anterior, mientras más iteraciones se hacen el error más se va acercando a 0.

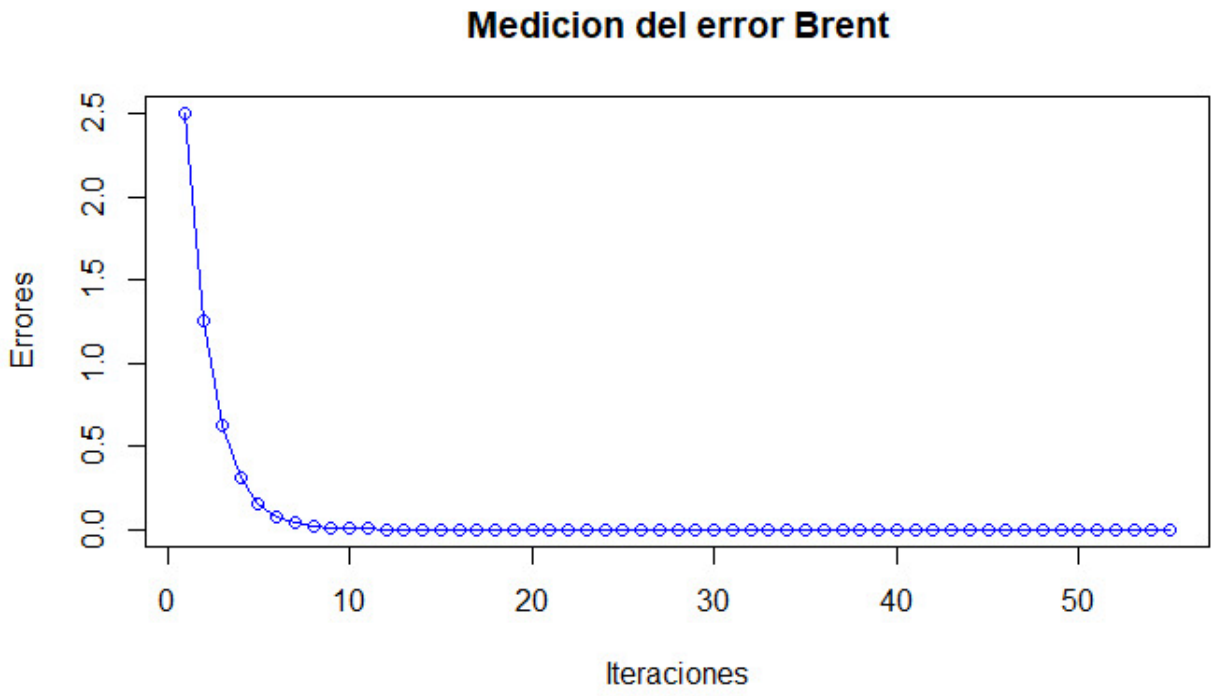


Fig 33. grafica error vs num iteraciones Brent

Ahora, si utilizamos el mismo algoritmo de bisección, se sigue el mismo patrón del ejercicio anterior donde el número de iteraciones es mucho mayor(108). Una desventaja de este algoritmo en este caso, es que no tiene configurada la presicion para que brinde más números decimales como si se hizo en el algoritmo de Brent.

```
 las raíces son:
  0.6666641 y  1.638489
con error <= 6.162976e-33
 El numero de iteraciones es:  108
```

Fig 34. resultados bisección

Tolerancia de $1.e-64$

Para este ultimo ejercicio se utilizó una tolerancia más pequeña ($1.e-64$) y una presicion aún mayor de 256 bits. Y cómo se tiene más bits el resultado de la raiz da con más números decimales. 0.6666698708192892000817759410841537389976379452294160207040896064465686698134835

```
[1] "Raiz:"
1 'mpfr' number of precision  256   bits
[1] 0.6666698708192892000817759410841537389976379452294160207040896064465686698134835
[1] "Número de iteraciones"
1 'mpfr' number of precision  128   bits
[1] 107
```

Fig 35. resultados

Con estos resultados se realizó la gráfica siguiente.

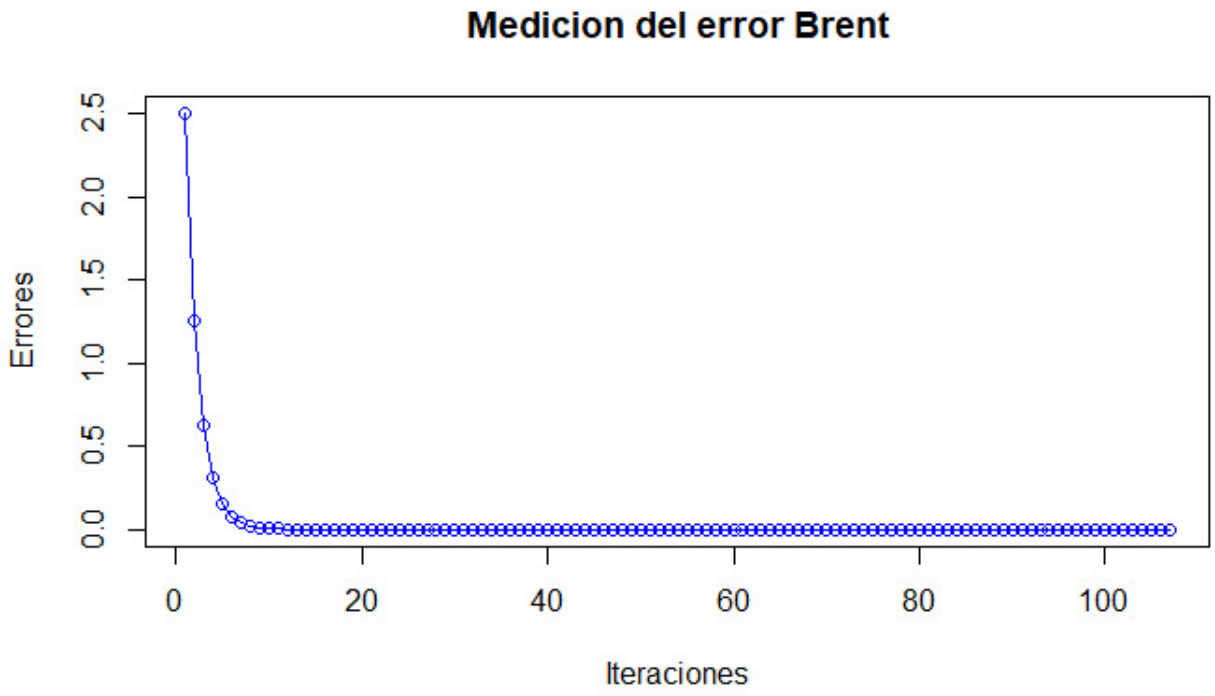


Fig 36. grafica error vs num iteraciones Brent

Con el algoritmo de biseccion el número de iteraciones, de nuevo, es casi el doble.

```
 las raíces son:
  0.6666641 y  1.638489
con error <= 7.596454e-65
 El numero de iteraciones es:  214
```

Fig 37. resultados bisección

Finalmente es necesario comparar las gráficas de relacion de error para Brent y para Bisección. Se observa que está más inclinada la gráfica de bisección, debido a los resultados que se vieron anteriormente.

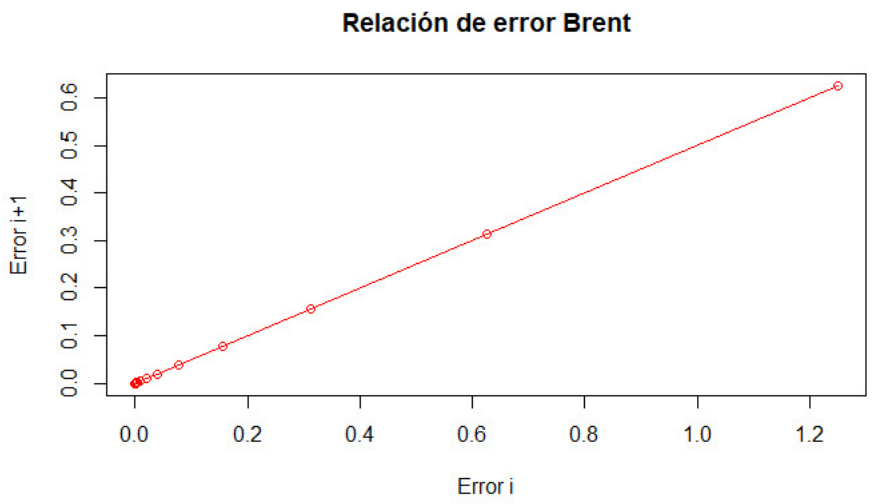


Fig 38. Error i+1 vs error i para Brent

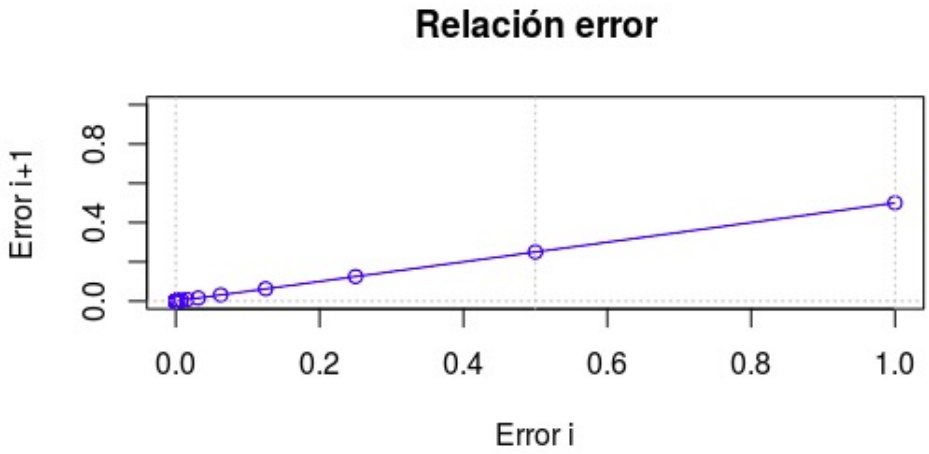


Fig 39. Error i+1 vs error i para Bisección

Conclusiones

Efectivamente el método de Brent comparado con el de biseccion del cual se deriva, es mejor ya que reduce significativamente el número de iteraciones a casi la mitad.

3. Optima Aproximación Polinómica

3.1. Explicación algoritmo de Remez

3.1.1. Definición

El algoritmo de Remez es un algoritmo iterativo utilizado para encontrar aproximaciones simples a las funciones, en concreto, las aproximaciones de funciones en un espacio se Chebyshev. Un espacio Chebyshev es el subespacio de polinomios de orden n en el espacio de las reales funciones continuas en un intervalo c[a,b]. El polinomio de mejor aproximación dentro de un subespacio dado se define como el que minimiza la máxima diferencia absoluta entre el polinomio y la función.

3.1.2. Procedimiento

El algoritmo a grandes pasos se puede definir como:

- Resolver el sistema de ecuaciones.
- Usar b_n como coeficiente formar el polinomio $P(n)$
- Encontrar el conjunto M de puntos con el error máximo local $P_n - f(x)$
- Si los errores en cada $m \in M$ son de igual magnitud y se alterna en el signo, entonces P_n es el polinomio de aproximación minmax. Si no, se reemplaza x con M y se itera nuevamente.

3.1.3. Solución

Para realizar la aproximación de la función $f(x) = e^{sin(x)cos(x)^2}$ en el intervalo $[-2^{-8}, 2^{-8}]$ mediante el método de Remeze fue necesario elegir el grado del polinomio de aproximación que queremos calcular. Para este caso especifico y debido al corto intervalo utilizado se eligió un polinomio de grado $n = 1$. Luego se procedió a hallar una cantidad de puntos que corresponden al valor de $n + 2$ mediante la creación de nodos conocidos como nodos de Chebyshev (llamado asi en honor al matemático ruso Pafnuti Chebyshov) muy conocidos en análisis numérico ya que intentan disminuir el fenómeno de Runge, problema que sucede cuando se usa interpolación polinómica con polinomios de alto grado utilizando nodos equidistantes. Estos nodos fueron creados haciendo uso de la libreria capn de donde se deriva la función chebnodegen que genera nodos chebyshev unidimensionales con los cuales se arma una matriz de tamaño $n+2$ con sus respectivos errores. El resultado de la matriz da los valores necesarios para reconstruir el polinomio resultante a partir de sus coeficientes.

3.2. Explicación teorema de Taylor

3.2.1. Definición

También conocido como método de multiplicación anidada, es un teorema usado en el análisis numérico creado por el matemático ingles William George Horner en el siglo $XVII$ que consiste en calcular el valor de un polinomio de grado n utilizando el menor número de multiplicaciones. Esta característica de realizar el menor número de operaciones posibles, es quiza la razón por la que es ampliamente usado pues esto quiere decir que tiene gran eficiencia.

El teorema enuncia que un polinomio de la forma $f(x) = a_0x^n + a_1x^{n-1} + ... + a_n$ puede evaluarse en un número n de multiplicaciones y n adiciones.

3.2.2. Procedimiento

La versión básica del problema se puede representar como:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^k + h_k(x)(x-a)^k,$$

Sustituyendo estas expresiones en el desarrollo en serie se obtiene el método de Taylor del orden de precisión que se desee.

3.2.3. Solución

Para realizar la aproximación de la función $f(x) = e^{sin(x)cos(x)^2}$ mediante el teorema de Taylor se hizo uso de la libreria pracma de la que se deriva una función Taylor la cual genera una aproximación polinomial local mediante series de Taylor.

3.3. Resultados

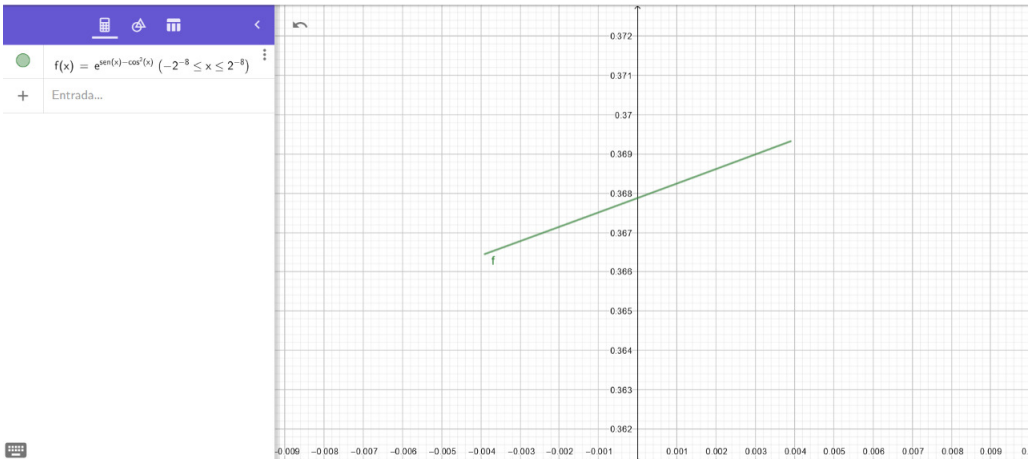
3.4. Comparación

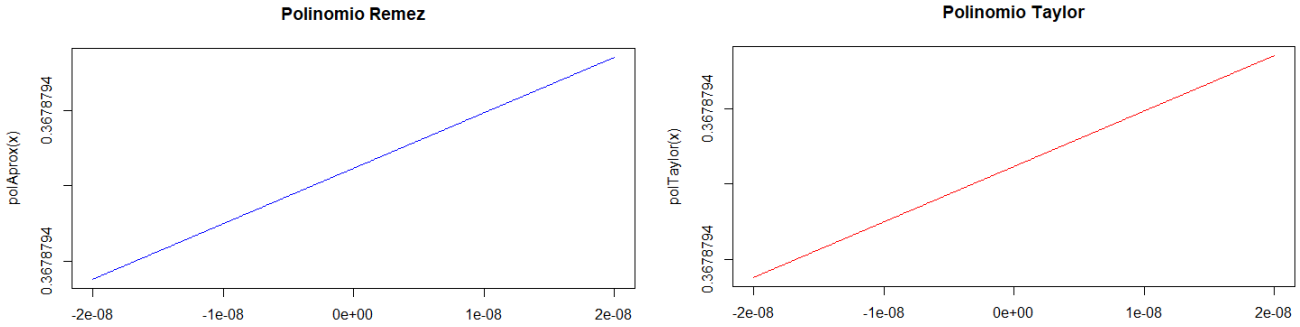
Una vez aplicados los métodos previamente descritos sobre la función $f(x) = e^{sin(x)cos(x)^2}$ en R, se observa que los resultados de ambos son idénticos, exceptuando el termino cuadrático, como se muestra en la siguiente imagen, que es obtenida de la consola de la herramienta una vez termina de ejecutar los algoritmos:

```
> polAprox = remezMethod(n,a,b,E)
> print(polAprox)
0.3678794 + 0.3678794*x - 5.551115e-09*x^2
>
> polTaylor = aproxTaylor(fx,0,n+1)
> print(polTaylor)
0.3678794 + 0.3678794*x + 0.5518192*x^2
```

Viendo estas salidas podemos concluir que ambos métodos son bastante buenos debido a que su aproximación es acertada, sin embargo, en el término cuadrático se puede apreciar mayor precisión en Remez debido a que su resultado es un poco más exacto.

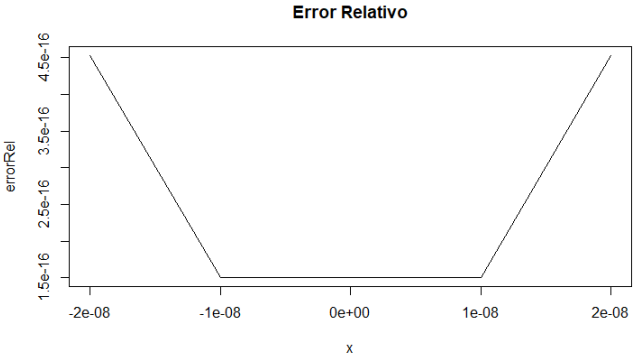
Una manera de ver de manera más clara el acierto de los resultado de los teoremas es con las gráficas resultantes de la comparación del polinomio de Remez con el polinomio de Taylor y con la gráfica de la función, como pueden ser apreciadas a continuación:

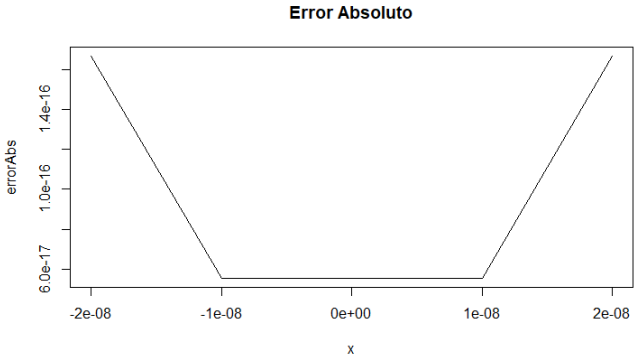




3.5. Errores

Las siguientes gráficas de error absoluto y error relativo fueron relizadas tomando como valor teórico el resultado del polinomio de Taylor y como valor experimental el resultado del polinomio de Remez. Un buen indicador es que estos errores se mantienen bastante cercanos a cero por lo que se puede decir que el método de Remez es muy efectivo tomando como referencia Taylor.





3.6. Convergencia y eficiencia de los algortimos

4. Referencias

[1]J. Fuente O'Connor, Ingeniería de los algoritmos y métodos numéricos. [El Ejido, Almería]: Círculo Rojo, 2017.

[2] HURTADO, N.A y DOMINGUEZ, S.F., Métodos Numéricos Aplicados a la Ingeniería, primera Edición Ebook, Patría, México, 2014.

[3] R. Brent, Algorithms for minimization without derivatives. Englewood Cliffs, N.J: Prentice-Hall, 1973.