Reto 1

Julian Builes
Santiago Bermudez
Daniel Reyes
Daniel Fierro

Septiembre 2020

A continuación se presentará la solución para el primer reto de la materia de Análisis numérico inicialmente modificamos el algoritmo de Horner que permita la evaluación de la derivada de la función y permita la evaluación de valores en el plano complejo con este acercamiento pretendemos facilitar la combinación de herramientas que nos permiten obtener el resultado mas cercano, de este modo podemos realizar una combinación de los métodos de Laguerre con Horner para obtener Newton-Horner; Seguido a esto analizamos el método de Brent el cual es un algoritmo complicado que combina el método de la bisección, el de la secante e interpolación cuadrática inversa, finalmente hacemos una óptima aproximación polinómica para encontrar la raíces de un polinomio partiendo de una función trigonométrica que sera expresada como un polinomio de Taylor para su evaluación en el algoritmo de Remez.

1 1. Evaluación de las raíces de un Polinomio

1.1 Evaluación derivadas

Sea $P(x) = a_0xn + a_1x_{n-1} + ... + a_n$ un polinomio entonces, implementar en R y/o Python una modificación del métodos de Horner que evalúe de manera eficiente f0(x0) y f00(x0) la primera y segunda derivada en cualquier punto.

El cálculo de las raíces de un polinomio puede presentar dificultades dependiendo de su grado ya que incrementa la complejidad por la cantidad de operaciones que debe realizar el equipo para realizar la aproximación de su raíz, debido a esto se explora la posibilidad de trabajar con su derivada evaluando así el desempeño del método en cada caso

Haciendo uso de la posición de la variable independiente lo cual corresponde al grado del polinomio se calcula el producto de su posición con el coeficiente valor en su posición que corresponde a su coeficiente, de este modo obtenemos un nuevo arreglo el cual refleja la derivada, se repite para obtener la segunda

$$f(x) = x^4 - 9x^2 - 5x^3 + 155x - 250$$
$$f'(x) = 4x^3 - 18x - 15x^2$$
$$f''(x) = 12x^2 - 18 - 30x$$

A partir de la evaluación teórica establecemos un modelo que no permite evaluar la derivada en la aproximación iterativa del algoritmo de Horner.

- Resultados derivada

```
Polinomio derivado 0
El numero de multiplicaciones con horner es  5
(-108+0j)
El numero de multiplicaciones con horner es  4
(126+0j)
El numero de multiplicaciones con horner es  3
(-36+0j)
```

Como observamos el numero de multiplicaciones requeridas para el algoritmo de horner se reduce en cada derivadada del polinomio esto es porque su grado de complejidad se reduce haciendo así que lo temo realizar menos operaciones.

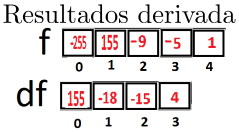
el polinomio esta represaentado por un vector de coeficientes entonces partimos del polinomio y su derivada.

$f(x) = x^4 - 5x^3 - 9x^2 + 155x - 250$

$f'(x) = 4x^3 - 15x^2 - 18x + 155$

Si conparamos las funciones observamos que el numero de coeficientes del original es 5 y el de su derivada 4 entonces se reduce en una unidad, representado en un arreglo.

Resultados derivada



Como apreciamos la derivada se puede representar de la manera tal que se desplaza una posición y el valor contenido se multiplica por su correspondiente valor numérico de la posición de esta manera evaluamos el polinomio en el algoritmo de horner sobre el vector aplicándolo únicamente sobre en el transformado df.

- ## 1.2 Evaluación del plano complejo

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

  La evaluación de los valores de la función en el plano complejo es fundamental ya que existen funciones que no cumplen condiciones de continuidad en el plano real además las raíces se pueden cancelar en puntos flotantes cercanos a cero.

  para evaluar los puntos en donde la función no tiene representación en el plano real hacemos uso de la función complex el cual es metodo que retorna un numero complejo a partir de una parte real e imaginaria.

1.3 Utilizar los resultados anteriores y el algoritmo de Laguerre para obtener un metodo de, NewtonHorner, de convergencia cuadratica en el que el algoritmo de Newton reemplaza al de Laguerre.

partimos del polinomio.

$$f(x) = x^4 - 5x^3 - 9x^2 + 155x - 250$$

Inicialmente obtenemos las raices a partir de un valor cercano de la raiz seguido a esto obtenemos una aproximación que nos permite representar el el algortimo de Newton-Horner, cabe mencionar que se aplica para un polinomio $pn(x)n3$ y debe tener al menos una raiz real entonces obtenemos el siguiente proceso.

$x0raizobtenidoenLaguerre$
$Pn(Xo)^Pn(Xo)- > EvaluacionenelalgoritmodeHorner$
$X1 = Xo - Pn(Xo)/Pn(Xo)- > EvaluacionenelalgoritmodeNewton$
.
.
.
.

iterativamente se determinan nuevas aproximaciones partiendo de la inicial obtenida por Laguerre.

2 Algoritmo Brent

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

  Como el algoritmo implementa el de bisección es bastante confiable porque toma valores existentes que nos permiten evaluar el polinomio de manera adecuada. El algoritmo intenta utilizar el método secante de convergencia lineal potencialmente o la interpolación cuadrática inversa si es posible, pero recurre al método de bisección más robusto si es necesario.

2.1 Implementación

Al probar el polinomio con el algoritmo de Brent se obtuvieron los siguientes resultados por iteración

Resultados método de Brent

Valor de X	Error	Iteracion
1,909985935	6,909985935	1
1,836015161	6,836015161	2
1,473086088	6,473086088	3
1,473086088	3,236543044	4
1,473086088	1,618271522	5
1,473086088	0,809135761	6
0,663950327	0,404567881	7
0,663950327	0,20228394	8
0,663950327	0,10114197	9
0,663950327	0,050570985	10
0,663950327	0,025285493	11
0,663950327	0,012642746	12
0,663950327	0,006321373	13
0,663950327	0,003160687	14
0,667111014	0,001580343	15
0,667015822	0,001485151	16
0,666928815	0,001398144	17
0,666928815	0,000699072	18
0,666928815	0,000349536	19
0,666579279	0,000174768	20
0,666666663	8,7384E-05	21
0,666666648	8,73695E-05	22
0,666666648	1,45349E-08	23
0,666666641	7,26747E-09	24

Resultados método de bisección

Iteración	Valor X
1	0.5
2	0,75
3	0.625
4	0.6875
5	0.65625
6	0.671875
7	0.6640625
8	0.66796875
9	0.666015625
10	0.6669921875
11	0.66650390625
12	0.666748046875
13	0.6666259765625
14	0.66668701171875
15	0.666656494140625
16	0.6666717529296875
17	0.6666641235351562
18	0.6666603088378906
19	0.6666622161865234
20	0.6666631698608398
21	0.666663646697998
22	0.666663408279419
23	0.6666632890701294
24	0.6666632294654846
25	0.6666631996631622
26	0.666663184762001
27	0.6666631773114204
28	0.6666631735861301
29	0.666663171723485
30	0.6666631707921624
31	0.6666631703265011
32	0.6666631700936705
33	0.6666631699772552
34	0.6666631699190475
35	0.6666631698899437
36	0.6666631698753918
37	0.6666631698681158
38	0.6666631698644778
39	0.6666631698626588
40	0.6666631698617493
41	0.6666631698612946
42	0.666663169861522
43	0.6666631698616357
44	0.6666631698616925
45	0.6666631698617209
46	0.6666631698617351
47	0.6666631698617422
48	0.6666631698617458
49	0.6666631698617476
50	0.6666631698617485
51	0.6666631698617489
52	0.6666631698617491
53	0.6666631698617492
54	0.6666631698617493

Y si comparamos el número de iteraciones con el método de la bisección ,

se evidencia que el método de Brent es más eficiente puesto que necesita menos iteraciones para llegar al valor de la raíz .
Brent se encarga de encontrar la raíz de una función en un intervalo dado , en este caso la función es :

$$f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$$

Y se evalúa en el intervalo

$$[0, 2]$$

y Brent tiene como condición si

$$f(a_0)f(b_0) < 0$$

determinamos a_0 como el valor de la función evaluado en la cota inferior del intervalo

$$a_0 = f(0) = 0^3 - 2 * 0^2 + \frac{4}{3} * 0 - \frac{8}{27}$$
$$a_0 = -\frac{8}{27}$$

y de ese mismo modo b_0 es calculado como el valor de la función en la cota superior del intervalo :

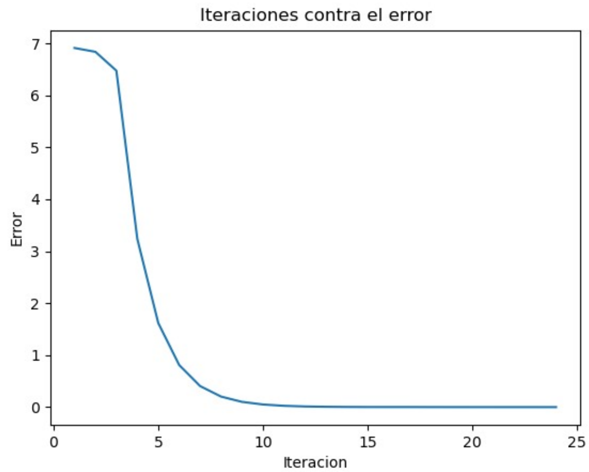
$$b_0 = f(2) = 2^3 - 2 * 2^2 + \frac{4}{3} * 2 - \frac{8}{27}$$
$$b_0 = \frac{64}{27}$$

para evaluar que :

$$f(a_0)f(b_0) < 0$$
$$-\frac{8}{27} * \frac{64}{27} < 0$$

Entonces el método si se puede aplicar para este polinomio en este intervalo .

El error por iteración



Podemos observar que el erro a medida que se iba iterando el algoritmo de Brent , iba disminuyendo, es decir cada vez se acerca más a la raíz de la función , se puede analizar que el error disminuye considerablemente en las primeras iteraciones , para después oscilar sobre aproximadamente el mismo error

3 Óptima Aproximación Polinómica

- Aplicar esta ttécnica 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

- ## 3.1 Definición del problema

La evaluación de las raíces de una función trigonométrica presenta grandes retos debido a su naturaleza, sabemos gracias al teorema fundamental del álgebra podemos encontrar al menos una raíz en un polinomio de grado mayor a cero; Hemos estudiado métodos que nos permiten hacer la aproximación de raíces de un polinomio, en este sentido vamos a expresar la función trigonométrica en forma polinómica, para ello se hace uso del método de Taylor para obtener su polinomio de Taylor, existen además maneras de optimizar la búsqueda de acuerdo a la paridad del grado polinomial, es importante mencionar que el polinomio se puede cancelar en puntos donde sus coeficientes no son cero, flotantes cercanos a cero, un acercamiento utilizado es asignar valores de 0 para que estos valores se cancelen.

Por otra parte, el algoritmo de Remez se utiliza para obtener un polinomio optimo que se aproxima a una funcion $f(x)$ en un intervalo dado. Este algoritmo es de caracter iterativo, que converge a un polinomio que tiene una funcion de error de $n+2$ niveles extremos. Dados esos $n+2$ puntos de referencia se resuelven las ecuaciones:
$P(x_1) - f(x_1) = +\epsilon$
$P(x_2) - f(x_2) = -\epsilon$
$P(x_3) - f(x_3) = +\epsilon$
.
.
.
$P(x_{n+2}) - f(x_{n+2}) = \pm\epsilon$

Despues de esto se alternan los lados de la derecha
$P_0 + P_{1x_1} + P_{2x_1^2} + P_{3x_1^3} + ... + P_{nx_1^n} - f(x_1) = +\epsilon$
$P_0 + P_{1x_2} + P_{2x_2^2} + P_{3x_2^3} + ... + P_{nx_2^n} - f(x_2) = -\epsilon$

Dado que todas las $f(x)$ son conocidas, a su vez que todos los datos x, se evidencia la existencia de $n+2$ ecuaciones lineales. Y ddado los puntos x se puede resolver el sistema y obtener el polinomio y su ϵ.

La aproximacion de Taylor, es el mismo concepto de las series de Taylor, la cual es una aproximcion de funciones mediante series de potencias entre polinomios, cuya suma se calcula a partir de derivadas de la funcion para un valor especifico, uno de los puntos mas importantes de este metodo es su capacidad de calcular de manera optima la aproximacion.

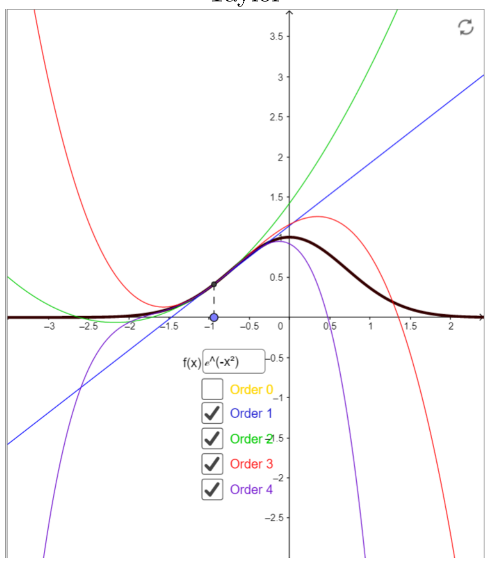
Esto se obtiene a partir de las siguientes potencias:

$f(a) + \frac{f^{'}(a)}{1!}(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \frac{f^{'''}(a)}{3!}(x-a)^3 + ...$
Que se puede ver asi:

$$\sum_{i=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Se procedio a enviar una funcion $f(x)$ a los algoritmos de Taylor/Remez , y lo simplifica en un polinomio $P(x)$ y se obtiene que, al remplazar el valor de x en ambas funciones el resultado es el mismo. Entonces, se compara algoritmo de Taylor/Remez con el valor original.

Taylor



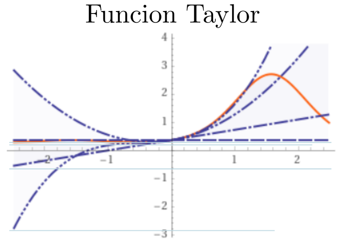
Se muestra, en la grafica anterior, el comportamiento de las aproximaciones de Taylor, con visiblimente almenos una raiz (corte en eje x).

Resultados método de Taylor

```
[1] "Taylor"
> print(polinomioCalculado)
4 'mpfr' numbers of precision  128   bits
[1] -0.99999784285432868902177006020792759954493
[3]  0.99999999995938337882250834809383377432882
```

Al utilizar el metodo de la aproximacion de Taylor para la funcion $f(x) = e^{sinx-cosx^2}$ se obtuvieron los resultados de los coeficientes de la funcion, mostrados en la imagen

Funcion Taylor



En la anterior grafica, validada en Wolfram, se nota el comportamiento que tienen las aproximaciones de Taylor, en $f(x) = e^{sinx-cosx^2}$,

Resultados método de Remez

```
> print("Remez")
[1] "Remez"
> print(r$p)
[1] -0.9999883 -0.4999968  1.0000000  1.0000000
>
```

Comparacion polinomio versus funcion original

```
Numero Evaluado en la funcion y el polinomio 1 'mpfr' number of precision  128   bits
[1] -0.00390625
el valor con taylor es  1 'mpfr' number of precision  128   bits
[1] 0.996086180210143608165632314364277286392
el valor esperado es 1 'mpfr' number of precision  128   bits
[1] 0.9960861802579226120154487238285236581492
[1] "-------------------------------------"
Numero Evaluado en la funcion y el polinomio 1 'mpfr' number of precision  128   bits
[1] -0.003716249999999999989539617439859853220696
el valor con taylor es  1 'mpfr' number of precision  128   bits
[1] 0.996276896066331864410776508083030547284
el valor esperado es 1 'mpfr' number of precision  128   bits
[1] 0.9962768961054836854774807902586591580203
[1] "-------------------------------------"
Numero Evaluado en la funcion y el polinomio 1 'mpfr' number of precision  128   bits
[1] -0.003526249999999999979079234879719706441392
el valor con taylor es  1 'mpfr' number of precision  128   bits
[1] 0.9964675766274581342836194345808615771739
el valor esperado es 1 'mpfr' number of precision  128   bits
[1] 0.9964675766592030036029419394082405669043
[1] "-------------------------------------"
Numero Evaluado en la funcion y el polinomio 1 'mpfr' number of precision  128   bits
[1] -0.003336250000000000022828960943854781362461
el valor con taylor es  1 'mpfr' number of precision  128   bits
[1] 0.9966582218523685065049448872590112356864
el valor esperado es 1 'mpfr' number of precision  128   bits
[1] 0.996658221877806516049766089562421741073
[1] "-------------------------------------"
```

Para obtener estos resultados, evaluando Taylor, se procedio a a implementar la funcionalidad de Horner con el fin de poder obtener el resultado del polinomio.Se observa que el resultado que arroja Taylor es muy cercano al valor que se espera obtener, con respecto al numero x evaluado, todo esto con respecto al polinomio $f(x) = e^{sinx-cosx^2}$.