UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

2DO CUATRIMESTRE DE 2022

Análisis Numérico

Búsqueda de Raíces

Curso:

Sassano

Integrantes:

Benito Agustin	abenito@fi.uba.ar	108100
Bucca Martín	mbucca@fi.uba.ar	109161
Dalton Ashling	adalton@fi.uba.ar	106407
Jorge Theo	tjorge@fi.uba.ar	107862

Lenguaje Elegido: Python

1.2 Objetivo del Trabajo

En este informe se trabaja sobre los de bisección, punto fijo, secante, Newton-Raphson y Newton-Raphson modificado, aplicados como resolución a diferentes problemas.

En el ejercicio 1 se muestran los diferentes valores que pueden tomar las fracciones al representarlas en la máquina en base al formato usado.

En el ejercicio 2 buscamos las raíces de un polinomio de segundo grado de 2 formas, tradicional y dinámica, desarrollando los algoritmos correspondientes.

En el ejercicio 3 se realiza la comparativa entre los diferentes métodos a la hora de buscar la raíz de 3 polinomios continuos diferentes.

1.3 Tabla de Contenidos

Índice

1.4 Introducción

Como fue mencionado en la lista de objetivos (1.2), en este informe buscamos desarrollar y resolver diferentes problemas relacionados con la búsqueda de las raíces de funciones con los métodos vistos y trabajados en clase (bisección, punto fijo, secante, Newton-Raphson y Newton-Raphson modificado). Esto lo hacemos a través de la experimentación de los métodos en la programación, que fue realizada en lenguaje Python con la plataforma Google Colab.

El objetivo de este trabajo no solo incluye la resolución de los problemas, sino que también poder comparar la eficiencia de cada método junto con la conveniencia de aplicación en cada caso.

Asimismo, en el ejercicio 1 también trabajamos con el concepto de propagación de

error junto con como la elección de la máquina con la que trabajamos puede afectar los resultados.

Para el desarrollo de este informe tendremos tanto la resolución del ejercicio 1 como las indicaciones que tiene que seguir el lector para poder visualizar el código y los resultados para los ejercicios 2 y 3. Mientras que la programación a la que nos referimos, los gráficos y las conclusiones correspondientes estarán incluidos en el Colab.

1.5 Cuerpo del informe

1.5.1 Ejercicio 1: No usar fracciones

Comenzamos planteando la fracción y pasándola a decimales para luego representarla en cada formato de máquina pedido.

$$Total = \sum \text{último número de padrón}(integrante_i) = 10$$

$$\frac{1}{Total} = \frac{1}{10} = 0.1$$

0.1 en binario:

0.1 = 0.0001100110011001100110011001100110011001100110011001100110011010...

Half (16 bits):

Half (1/10) = 0.1

0

Signo (1 bit)

-4 + 15 = 11

0	1	0	1	1

Exponente (5 bits)

1.	1	0	0	1	1	0	0	1	1	0

Mantisa (10 bits)

$$0.1 = 0.00011001100110 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} = 0.099975585$$

Siguiente:

$0.00011001100110 + 0.00000000000001 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-14}$
$= 0.1000036621$

Anterior:

0.099975585 - 2^{-14} = 0.099914549

Single (32 bits):

Single (1/10) = 0.1

0

Signo (1 bit)

-4 + 127 = 123

0	1	1	1	1	0	1	1

exponente (8 bits)

1.	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1

mantisa (23 bits)

$0.1 = 0.00011001100110011001101$
$= 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21}$
$+ 2^{-24} + 2^{-25} + 2^{-26} = 0.100000001$

Siguiente:

$0.00011001100110011001111$
$= 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21}$
$+ 2^{-24} + 2^{-25} + 2^{-26} + 2^{-27} = 0.1000000016$

Anterior:

0.100000001 - 2^{-26} = 0.099999986

Double (64 bits):

Double (1/10) = 0.1

0

signo(1 bit)

-4 + 1023 = 1019

0	1	1	1	1	1	1	1	0	1	1

exponente (8 bits)

1.	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

1	1	0	0	1	1	0	1	0

mantisa (52 bits)

$$0.1 = 0.0001100110011001100110011001100110011001100110011001100110011010$$

$$= 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} + 2^{-24} + 2^{-25} + 2^{-28} + 2^{-32} + 2^{-33} + 2^{-36} + 2^{-37} + 2^{-40} + 2^{-41} + 2^{-44} + 2^{-45} + 2^{-48} + 2^{-49} + 2^{-52} + 2^{-53} + 2^{-55} = 0.099999999$$

Siguiente:

$0.0001100110011001100110011001100110011001100110011001100110011010 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-20} + 2^{-21} + 2^{-24} + 2^{-25} + 2^{-28} + 2^{-32} + 2^{-33} + 2^{-36} + 2^{-37} + 2^{-40} + 2^{-41} + 2^{-44} + 2^{-45} + 2^{-48} + 2^{-49} + 2^{-52} + 2^{-53} + 2^{-55} + 2^{-56} = 0.0999999990000000138779$ (calculadora 64 bits)

Anterior:

0.0001100110011001100110011001100110011001100110011001100110011010- 2^{-56}= 0.0999999989999999861222 1 (calculadora 64 bits)

b) Si, debemos tener en cuenta que cada fracción es una variable más porque la máquina al realizar una fracción muchas veces no puede representar el número exacto entonces lo aproxima al más cercano, dependiendo de si es “Half”, “Single” o “Double” con mayor o peor precisión. Por lo tanto este error de redondeo/ aproximación se debe tener en cuenta en la propagación de errores, aunque sea muy pequeño.

c) Deberíamos tener en cuenta el tipo de precisión que tiene la herramienta. Si tiene una arquitectura “Half” va a poder asegurar 3 dígitos significativos, si tiene “Single” va a poder asegurar 7 y si tiene “Double” 15. La Herramienta (según el tipo que sea) te dará la respuesta más aproximada que pueda, pero siempre deberíamos tener en cuenta el error de redondeo que realiza esta para trabajar con los números con coma por ejemplo. Debido a esto, la conmutatividad y asociatividad se pierden en la suma y la multiplicación y es otro aspecto a tener en cuenta.

1.5.2 Ejercicio 2: Resolución de polinomio de segundo orden.

Para ver los resultados simplemente hay que correr la celda “Punto 2” y ver como varían los resultados de las raíces en base a los valores de a, b y c.

1.5.3 Ejercicio 3: Búsqueda de raíces.

Para ver los resultados del punto 3 es necesario seguir las siguientes indicaciones:

1. Correr la celda titulada “Funciones y derivadas”, donde se guardan las 3 funciones con sus respectivas derivadas que van a ser utilizadas en los distintos métodos y en distintas celdas del programa.
2. Correr la celda titulada “Gráfico de funciones en el intervalo [0, 2]” donde se muestran las distintas funciones en el intervalo pedido y sus raíces marcadas con un punto.
3. Correr cada celda (en orden) tituladas con los distintos métodos. En cada una se podrá observar debajo del código las distintas tablas, si es que son necesarias, donde se representan las iteraciones y en cada columna se representa f evaluada en el punto , el punto xn, y el error (en módulo) con respecto a la iteración anterior. En algunos métodos se muestra información extra como la función derivada evaluada en el punto o la derivada segunda evaluada en el punto.
4. Observar que debajo de cada celda hay una pequeña conclusión de lo que pasa para cada una de las funciones f1, f2, y f3.
5. Correr la celda titulada “Métodos”, donde se guarda la información de la búsqueda de raíces con sus iteraciones para cada función y para cada método, siendo que va a ser utilizada luego para comparar resultados y mostrar gráficos.
6. Correr las distintas celdas y observar los 3 gráficos para cada una.

Punto fijo

Hay que transformar el problema de búsqueda de raíces en uno de búsqueda de punto fijo. Para eso, partimos de las funciones $f_1(x), f_2(x)\ y\ f_3(x)\ y$ nos creamos $g_1(x), g_2(x)\ y\ g_3(x)$:

Para todos los casos vamos a buscar la $g(x)$ buscando el parámetro c para el cual la siguiente función cumpla los teoremas de Existencia y Unicidad:

$$g(x) = x - c.f(x) \qquad (1)$$

$$f_1(x) = x^2 - 2$$

Usando la fórmula: $g_1(x) = x - c.f_1(x) = x - c(x^2 - 2)$

Prueba de existencia: Como la : $g_1(x))$ es un polinomio de grado 2, elegimos c de modo que el vértice quede en un punto fácil de calcular. Usaremos $c=0.5$ de modo que queda:

$g_1(x) = -0.5x^2 + x + 1$

Esta función tiene un máximo en el punto $(1\ ,\ 1.5)$ y en los extremos del intervalo $[0\ ,\ 2]$ toma los siguientes valores: $g_1(0) = 1\ \ ;\ g_1(2) = 1$

Por lo que queda demostrado que cumple el teorema de existencia ya que en el intervalo $[0\ ,\ 2]\ ,\ g_1(x) \in [1\ ,\ 1.5]$

Prueba de unicidad: $g_1{'}(x) = 1 - x$

Esto es una recta constantemente decreciente por lo que me fijo en sus extremos:

$g_1{'}(0) = 1\ \ g_1{'}(2) = -1$

Esto es un problema ya que no podemos encontrar una constante k:$|\ g_1{'}(x)| \leq k < 1$. Pero sí encontramos una constante si usamos el intervalo $[0.1,1.9]$ ya que:

$g_1{'}(0.1) = 0.9;\ \ g_1{'}(1.9) = -0.9$

De modo que con $k=0.9$ cumple el teorema de unicidad

$$f_2(x) = x^5 - 6.6\,x^4 + 5.12x^3 + 21.312x^2 - 38.016x + 17.28$$

Aplicando la formula (1):

$$g_2(x) = x - c.f_2(x) = x - c(x^5 - 6.6\,x^4 + 5.12x^3 + 21.312x^2 - 38.016x + 17.28)$$

Con esta $g_2(x)$ podemos encontrar algún intervalo en donde se cumpla la existencia del punto fijo. Sin embargo, hay problemas con la prueba de unicidad. Nos dimos cuenta que 1.2 es raíz de multiplicidad 3. Por lo que al derivar $f_2(x)$ esta también se cancela en el 1.2

haciendo que la derivada de $g_2(x)$ siempre de 1 en el 1.2 de modo que nunca va a existir una constante menor a 1 que acote la derivada.

Terminamos usando una c cualquiera, que la calculamos para que minimice la cantidad de iteraciones necesarias para llegar al error de 10^{-5}. De todos modos converge muy lento, llegando a estar 10 minutos corriendo y no llegar al error de 10^{-13}.

$$f_3(x) = (x - 1.5)e^{-4(x-1.5)^2}$$

Aplicando la formula (1): $g_3(x) = x - c.f_3(x) = x - c(x - 1.5)e^{-4(x-1.5)^2}$

Con c=1 se verifican las hipótesis del teorema de existencia para el intervalo $[0\ ,\ 2]$. Sin embargo, al tratar de acotar la derivada:

$$g'_3(x) = 1 - e^{-4(x-1.5)^2} + 8(x - 1.5)^2 e^{-4(x-1.5)^2}$$

con una constante menor a 1 vemos que pasa solo para el intervalo $(1.14644\ldots\ ,\ 1.853553\ldots)$ por lo que tenemos que elegir un intervalo incluido en el anterior como el $[1.15\ ,\ 1.8]$

Para este intervalo encontramos una constante:

$g'_3(1.15) \cong 0.9877$, $g'_3(1.8) \cong 0.8046$ y tiene un mínimo en el punto $(1.5\ ,\ 0)$ por lo que k ≅ 0.9877<1

Orden De Convergencia P Vs Iteraciones

Analizando las tablas y los gráficos obtenidos y expuestos en el colab, vemos que el método de bisección tiene convergencia lineal en todos los casos y que tarda una gran cantidad de iteraciones para hacerlo (siendo f3(x) la excepción ya que al haber tardado solo dos iteraciones en converger, no es lo suficiente para graficar). Por otra parte, en los casos de f1(x) y f2(x) los métodos de Newton Raphson, Newton Raphson modificado y secante lograron convergencias con menores iteraciones y hasta los mismos valores en un caso. f3(x) es la excepción al caso ya que la semilla fue elegida de forma incorrecta y por lo tanto, nunca converge.

Tomando el método del punto fijo, vemos que para la función f2(x) no logra converger mientras que para f1(x) y f3(x) si lo hace y rápidamente. Confirmando nuevamente que el método más conveniente para usar varía acorde a la función.

Cabe mencionar el hecho de que por momentos, el gráfico no toma los valores esperados sino que, por ejemplo, llega a valores mayores a 2, esto sucede por distintos tipos de errores y más aún cuando se acumulan. Uno de estos errores incluye que estamos aproximando un límite de n tendiendo a infinito con pocas iteraciones, lo cual ya de por sí es muy lejano a la teoría. Asimismo, la sensibilidad del logaritmo es considerable y por momentos

la convergencia cuadrática lleva a tener la siguiente iteración con pocas cifras significativas de diferencia y en consecuencia no son suficientes para describir el comportamiento que estamos observando. Aún así, en todos los casos pudimos ver cómo a medida que aumentaban las iteraciones, el gráfico se volvía menos errático.

log10(|Δx|) vs iteraciones

Caso $f1(x)$:

Podemos ver que los métodos "Newton", "Newton modificado" y "secante" necesitan mucho menos iteraciones para converger con un criterio de paro de 1.10^{-13} que "Bisección" y "Punto fijo". En este gráfico lo que queremos observar es que tan rápido disminuye el error entre dos iteraciones seguidas para cada método (Δx). Por eso vemos que en los que convergen más rápido los gráficos caen abruptamente, eso quiere decir que el error se hace menor cada vez más rápido. En cambio en "Biseccion" y "Punto fijo", los gráficos caen también, pero más lento.

Caso $f2(x)$:

Vemos que los métodos "Secante", "Newton" y "Newton modificado" también caen abruptamente, mostrando que el error entre iteraciones se va a 0 cada vez más rápido. En esta función, "Bisección" converge mucho más rápido y se refleja en el gráfico ya que vemos que hay menos iteraciones y que por cada una el error es mucho menor, lo que hace que la función log caiga sea con una pendiente más pronunciada. Como punto fijo no converge, vemos que ya en la iteración 175, la curva se "aplana", lo que quiere decir que entre iteraciones el error es casi el mismo, no se hace menor y por lo tanto va a tardar mucho en llegar a con un criterio de paro que tenga un error muy chico (como 1.10^{-13})

Caso $f3(x)$:

Los métodos "Newton" y "Newton modificado" tampoco convergen con esta función por lo que el gráfico deja de caer en la iteración 10 y el error disminuye muy poco a medida que pasan las iteraciones. En los casos de "Secante" y "Bisección" el método converge en la primera iteración por lo que no hay nada que mostrar en el gráfico, ya que no podemos ver como cae el error en función de las iteraciones. Punto fijo, en cambio converge muy rápido y por eso tiene una caída abrupta hasta su última iteración que termina de converger.

log10(|xCandidata − xReal|) vs iteraciones

En este caso, los gráficos se asemejan mucho a los expuestos en el caso anterior ya que lo que buscan mostrar es que tan cerca está el punto de la raíz original a medida que pasan las iteraciones. Si el gráfico cae abruptamente, significa que se acerca muy rápido a

la raíz original. En cambio, si no cae y se “aplana” la curva, significa que el método nunca se va a acercar a la raíz real, porque a medida que pasan las iteraciones la distancia entre el punto en cuestión y la raíz real se mantiene constante, indicando que se acerca prácticamente nada.

1.6 Conclusiones:

Pudimos ver que el método de Newton-Raphson es muy eficiente, presentando un orden de convergencia cuadrático, asimismo, vemos que se puede encontrar una disminución del número de iteraciones utilizando el método de Newton-Raphson modificado. Aún así, hay que tener en cuenta que la elección de una semilla apropiada es crucial para la eficiencia de este método, ya que, que sea un método que converge rápidamente, vimos que también puede hacer lo contrario con la misma velocidad.

En el caso de punto fijo, notamos que la convergencia es lineal y puede ser rápida, aunque hay que considerar que este método puede variar según la función g(x) elegida a la hora de aplicarlo. Si se lleva a cabo un análisis de la función y se elige g(x) de forma precavida y consciente, se pueden evitar estos inconvenientes.

El método de la secante también puede ser considerado eficiente, viendo como logra convergencia en una cantidad apropiada de iteraciones. Aún así, hay que considerar que no siempre converge y que se necesitan más valores de Xn para aplicarlo, siendo necesarios además de Xn, Xn-1 y Xn-2

En el caso del método de bisección, podemos decir que si bien el mismo garantiza convergencia en todas sus aplicaciones, tiene un orden de convergencia lineal y suele necesitar una cantidad muy alta de iteraciones por lo que que no suele ser conveniente para resolver. Sin embargo, este método puede ser práctico a la hora de definir la semilla, si es que es necesario para luego usar el método más apropiado.

Referencias

[1] Cheney, W.; Kincaid, D. Numerical Mathematics and Computing. 6ta ed. EE.UU.: Thomson Brooks/Cole, 2008.

[2] Burden, R. L,; Faires, J.D. Análisis Numéirco. 2da ed. México: Iberoamérica, 1996.