Programación y análisis de algoritmos numéricos Matlab.

Bastián Loyola Jara (5 horas)
Departamento de Ingeniería Informática
Universidad de Santiago de Chile, Santiago, Chile
bastian.loyola@usach.cl

***Resumen*—Los algoritmos numéricos nos permiten encontrar la solución de un problema numérico a través de unos pasos bien definidos, existen varios con sus propios requisitos y sus propios resultados los cuales no siempre serán el resultado el final, ya que se trabaja con la aproximación, en este documento se contrastar mediante varias formas el método Newton-Raphson, Secante, Bisección y Regula-Falsi, para así encontrar el más eficiente, además se apreciará el uso de errores de los distintos métodos, para analizarlos**

***Palabras claves*—Algoritmo numérico, MatLab, error.**

I. Introducción

Las matemáticas, un elemento tan antiguo como la historia misma, intrínseco en toda la humanidad, la vimos nacer y la hemos visto evolucionar, cada vez alcanzando mayores niveles de dificultad, es por ello mismo que la humanidad ha necesitado resolver dichas dificultades y problemas, elaborando distintos métodos para resolver el mismo problema, de esta forma en este documento se hablará acerca de dichas soluciones, los algoritmos numéricos son las soluciones a los problemas matemáticos, es la serie de pasos para poder obtener el resultado, de estos existen distintas clases para distintos problemas como por ejemplo aquellos destinado a encontrar las raíces de una ecuación. de esta forma en este documento se revisarán y compararán 4 algoritmos para resolución de ecuaciones no lineales implementadas en el IDE matlab y 1 para sistemas de ecuaciones.

II. Antecedentes

II-A. Marco Teórico

II-A1. Matlab: Es una plataforma de programación y calculo numérico, que incluye un entorno de desarrollo integrado (IDE)[1], debido a sus funciones para el cálculo y matemáticas, es ideal para diseñar e implementar algoritmos numéricos para resolución de problemas matemáticos

II-A2. Estimación a Priori: Son aquellas estimaciones que dependen de una solución aproximada exacta, sin calcular para un problema; permitiendo teóricamente evaluar antes de siquiera obtener la solución.[2]

II-A3. Estimación a Posteriori: A diferencia de la anterior, dependen de la solución calculada pero no la solución exacta y permiten ser evaluados en la práctica.[2]

II-A4. Algoritmo numérico: Un algoritmo es según la RAE: Çonjunto ordenado y finito de operaciones que permite hallar la solución de un problema"[3] y numérico como: "Perteneciente o relativo a los números"[4]; entonces un algoritmo numérico debería ser aquel conjunto de operaciones relacionadas con los números que nos permite hallar la solución de un problema matemático.

Tabla I: Ejemplo de tabla

Columna 1	Columna 2	Columna 3	Columna 4
Ejemplo 1	A	A	A
Ejemplo 2	B	B	B
Ejemplo 3	C	C	C
Ejemplo 4	D	D	D

II-B. Algoritmos

A continuación, se podrán apreciar los distintos algoritmos numéricos, empleados para encontrar las raíces de ciertas ecuaciones

II-B1. Newton Raphson: Es de los métodos más populares para encontrar las raíces de una ecuación, utiliza como base los polinomios de Taylor[5], se define un punto arbitrario para desde el, acercarse a la función utilizando el valor de la derivada de esta, como se puede apreciar en la imagen 1 y el algoritmo para dicho método es presente en algoritmo 1

Algoritmo 1 Newton Raphson

Input: x0 punto arbitrario, tol tolerancia, f función a resolver.
Output: xn solución de la ecuación
$xn \leftarrow x0$
$e \leftarrow abs(f(xn))$
while $(e > tol)$ **do**
  $xn \leftarrow xn - \frac{f(xx)}{df(xx)}$
  $e \leftarrow abs(f(x))$
end while
return (xn)

II-B2. Secante: A diferencia del método anterior, en el método de la secante, no se necesita el valor de la derivada para cada aproximación, sin embargo, se utiliza una expresión

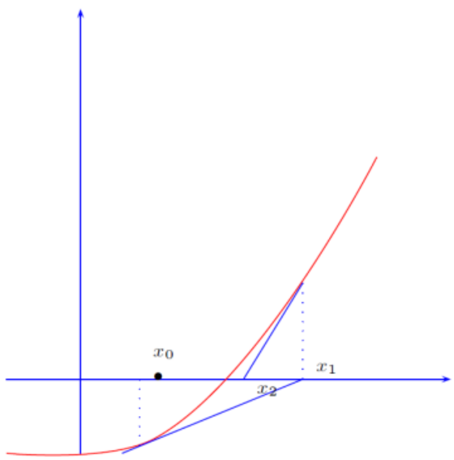

Figura 1: Método Newton Raphson

para cada valor de x aproximado, que deriva del método de Newton-Raphson está expresión de una manera simplificada está dada por el algoritmo 2, es importante aclarar que este método utiliza dos valores de x para trabajar, en la imagen 2 se puede apreciar la representación de este método. [5]

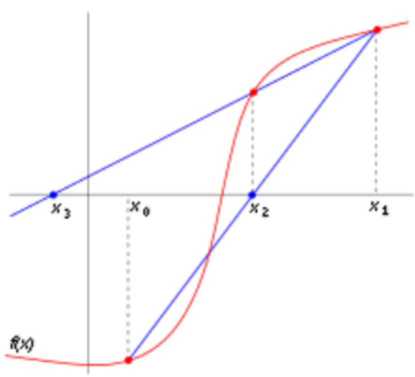

Figura 2: Método de la Secante

Algoritmo 2 Secante

Input: x0 primer punto arbitrario, x1 segundo punto arbitrario, tol tolerancia, f función a resolver.
Output: xn solución de la ecuación
$e \leftarrow abs(f(x))$
while $(e > tol)$ **do**
    $xn \leftarrow x1 - \frac{f(x)*(x1-x0)}{f(x1)-f(x0)}$
    $x1 \leftarrow x0$
    $x0 \leftarrow xn$
    $e \leftarrow abs(x1 - x0)$
end while
return (xn)

II-B3. Bisección: Se basa en el teorema del valor intermedio y en los valores de la función de dos puntos que van a ir cambiando con cada aproximación, mediante estos valores se podrá encontrar la raíz de la ecuación en cuestión, a continuación, se mostrará en la imagen 3 y el cómo implementarlo con el algoritmo 3. [5]

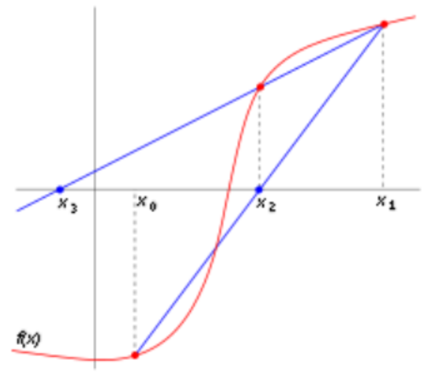

Figura 3: Método de la Bisección

Algoritmo 3 Bisección

Input: a comienzo de un intervalo definido, b final de un intervalo definido, tol tolerancia, f función a resolver
Output: c solución de la ecuación
while $(\frac{b-a}{2} > tol$ **do**
    $c \leftarrow a + \frac{b-a}{2}$
    if $(f(c) = 0)$ **then**
        $a \leftarrow c$
        $b \leftarrow c$
    end if
    if $(sign(f(a)) = sign(f(c)$ **then**
        $a \leftarrow c$
    end if
    if $(sign(f(b)) = sign(f(c)$ **then**
        $b \leftarrow c$
    end if
end while
return (c)

II-B4. Regula-Falsi: Una variante el método de la bisección, es el Regula-Falsi, el cual divide el intervalo definido [a,b] en dos subintervalos, donde se encuentra el punto c, el cual a diferencia del punto medio m, este consiste en la intersección de una recta secante que pasa por los valores (a,f(a)) y (b,f(b)) con el eje x; a continuación, se mostrará el algoritmo empleado en 4 y el cómo es gráficamente en la imagen 4. [5]

II-B5. Newton multivariable: A diferencia de los 4 métodos anteriores para encontrar las raíces de una ecuación, el método de newton multivariable sirve para resolver sistemas de ecuaciones, que recae la responsabilidad en calcular el inverso del Jacobiano generado por el sistema de ecuaciones y el uso de un punto y tener unos valores iniciales como solución aproximada.[5]

II-C. Calculo de los errores

A continuación, se mostrará el cálculo de los errores a priori de los algoritmos 1, 2 y 3; obtenidos desde el material

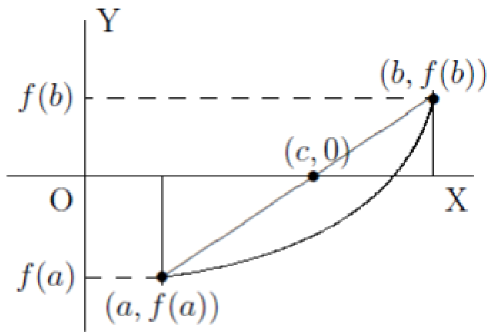

Figura 4: Método de la Regula-Falsi

Algoritmo 4 Regula-Falsi

Input: a primer punto arbitrario, b segundo punto arbitrario, tol tolerancia, f función a resolver.
Output: c solución de la ecuación
$c \leftarrow a$
while $(abs(f(c)) > tol)$ **do**
  $c \leftarrow b - f(b) * \frac{f(b-a)}{f(b)-f(a)}$
  if $(f(c) = 0)$ **then**
    $a \leftarrow c$
    $b \leftarrow c$
  end if
  if $(sign(f(a)) = sign(f(c)$ **then**
    $a \leftarrow c$
  end if
  if $(sign(f(b)) = sign(f(c)$ **then**
    $b \leftarrow c$
  end if
end while
return (c)

de estudio del curso .Älgoritmos Numéricosïmpartido en la Universidad de Santiago de Chile.[5]

II-D. Error a priori Newton-Raphson

Se calcula con la siguiente formula 1, es un proceso en donde se consideran valores previos en iteraciones anteriores, donde r es una solución simple de la ecuación y e_n es el error a priori en la iteración n.

$$e_{n+1} \approx \frac{1}{2} * \frac{f''(r)}{f'(r)} * e_n^2 \tag{1}$$

II-E. Error a priori Secante

Se calcula con la siguiente formula 2, un proceso donde también se consideran los valores previos de cada iteración para encontrar el valor final, donde r es una solución simple de la ecuación y e_n es el error a priori en la iteración n.

$$e_{n+1} \approx \frac{1}{2} * \frac{f''(r)}{f'(r)} * e_n * e_{n-1} \tag{2}$$

II-F. Error a priori Bisección

A diferencia de los errores anteriores, esté no sé calcula considerando los errores previos, si no que se considera los puntos del intervalo definido y el total de iteraciones a realizar

$$e_n = \frac{b-a}{2^n} \tag{3}$$

III. Materiales y métodos

III-A. Materiales

Los materiales utilizados para creación y desarrollo del presente documento: -Software LaTeX, para la creación del presente documento
-El ambiente donde fue realizado el documento y la simulación fue una computadora con sistema operativo Windows 10 de 64 bits.
-IDE Matlab, para la implementación y prueba de los algoritmos.

III-B. Métodos

Primero se desarrollará los 4 métodos mencionados para encontrar raíces de una ecuación, los cuales son Newton-Raphson, Secante, Bisección y Regula-Falsi, se implementaran mediante Matlab siguiendo los algoritmos 1, 2, 3 y 4, modificándolos un poco como por ejemplo agregando un límite de iteraciones para detenerlo en caso de ser necesario, agregar la derivada de la ecuación como una entrada para los algoritmos que la requerían, además añadiremos dos salidas a cada función que será, una lista llamada errores que contendrá los valores de los errores para cada iteración y otra llamada convergencia que almacenará los valores de la solución aproximada de cada iteración, esto para luego comparar dichos algoritmos para una misma función en temas de tiempo y espacio utilizado.
Los algoritmos implementados serán probados en 2 ecuaciones distintas siendo la ecuación 4 y 5

$$0 = x^2 - 60 \tag{4}$$

$$0 = x^3 - 2x^2 + ln(2x + 1) \tag{5}$$

Para cada ecuación, se comparan los 4 algoritmos aplicados buscando el menor valor de error obtenido, los errores también serán comparados con los errores a priori para el caso del algoritmo 1, 2 y 3, considerando el valor de e_n, donde n es el número de iteraciones, para los casos que se deba calcular en base a valores de error anterior, se considerará el primer valor de la lista de errores para realizar el cálculo en el caso de Newton-Raphson y Secante. Además, se revisará el tiempo empleado en cada algoritmo y su espacio utilizado, para ambas ecuaciones.

Para el caso del sistema de ecuaciones con Newton-Multivariable se realizará el cálculo del inverso del jacobiano para poder aplicarlo y encontrar los valores que soluciones el sistema de ecuaciones

IV. Resultados

Se logró realizar la implementación de los 4 algoritmos para la encontrar la raíz a una ecuación para las dos ecuaciones mencionadas 4 y5, representando los resultados de los errores mediante gráficos. Respecto a los valores encontrados, tanto para la ecuación 4 como para la ecuación

5, el método Newton-Raphson logra encontrar el valor en la menor cantidad de iteraciones, esto se puede apreciar en las imágenes 5 y 6 (con 40 iteraciones como tope), donde la variable independiente es el número de iteraciones y la dependiente es el valor de los errores.

https://www.overleaf.com/download/project/61b2d6577b0efb733a92f3(
e1679fb327689fde/output/output.pdf?compileGroup=standardclsiserver
reg-e2-e-h5qvpopupDownload=true

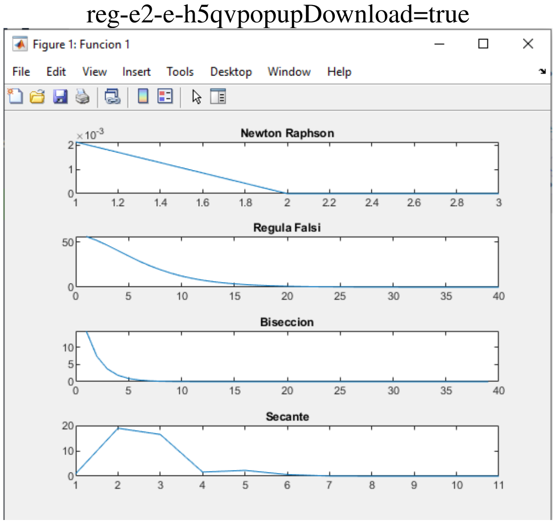

Figura 5: Valores de los errores de la ecuación 1 con n = 40

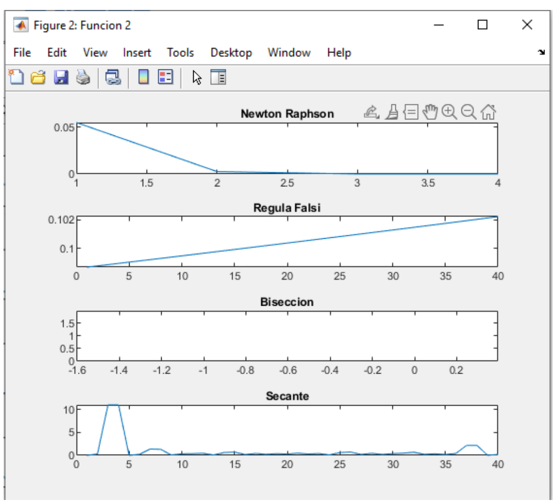

Figura 6: Valores de los errores de la ecuación 2 con n = 40

En cuanto a los valores de error mínimo encontrado con cada método, como se puede apreciar en la imagen 7, el método Regula-Falsi se escapa de los valores cercanos a 0 que tienen los demás métodos, esto dado a que Regula-Falsi requiere de un gran número de iteraciones para encontrar el valor de la solución final, en cambio los demás métodos encontraron el valor en una iteración anterior y dado si encontró el valor, el mínimo valor de error debe ser 0.

A continuación se mostrará, la comparación de los errores mínimos con los errores a priori de la primera ecuación 8 y de la segunda ecuación 9; se puede apreciar que el cual hubo

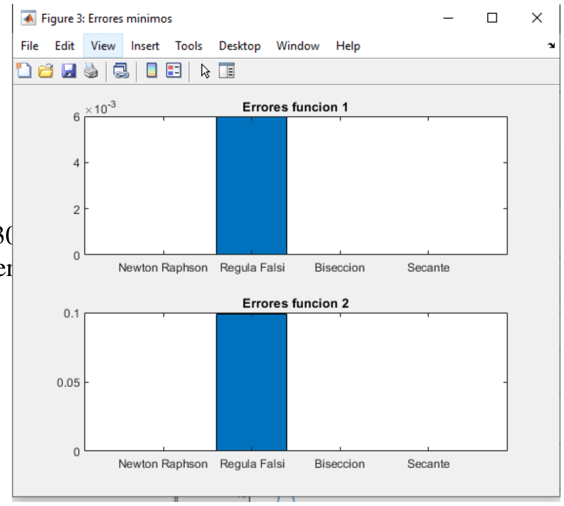

Figura 7: Errores mínimos para ambas ecuaciones

mayor diferencia entre el error mínimo y el error a priori, fue en el método de la bisección; en el resto de métodos dan números tan pequeños que al normalizarlos a 4 dígitos significativos, se aproxima a 0; es importante aclarar que no se pudo aplicar el error a posteriori para el algoritmo de Regula-Falsi.

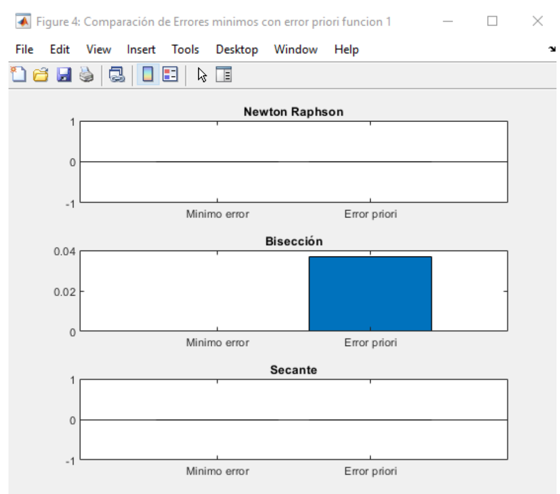

Figura 8: Comparación Error Min y Error a Priori Ecuación 1

Por ultimo analizaremos los espacios y tiempos utilizados, para el espacio como se podía apreciar en las imágenes 5 y 6, el método Newton-Raphson utilizaba menor cantidad de iteraciones en comparación al resto para llegar a un error igual a 0, de esta forma se traduce en menor número de instrucciones y por lo tanto menor número de espacio utilizado En cuanto al tiempo se creó que el gráfico de la imagen 10, para comparar los tiempos empleados en cada método utilizando las funciones tic-toc de Matlab, estas funciones fueron llamadas antes y después del llamado de cada método, como se puede apreciar para ambas ecuaciones el método más rápido fue el de Newton-Raphson, seguido de la bisección, tercero el método de la secante y finalmente el regula-Falsi siendo el más lento de los 4; un detalle a

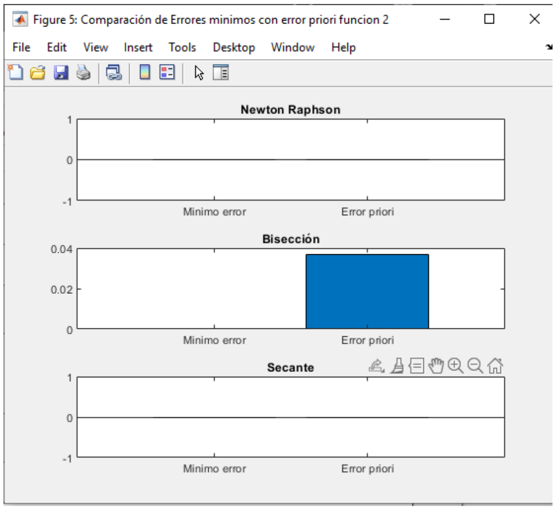

Figura 9: Comparación Error Min y Error a Priori Ecuación 2

considerar es que dado la forma de la ecuación 5 que incluye logaritmo natural, aquellos valores aproximados que provocarán un número menor o igual a 0 dentro del logaritmo provocaría que fallará.

La implementación del algoritmo numérico para hallar la solución de un sistema de ecuaciones, Newton multivariable, no logró ser implementado con éxito.

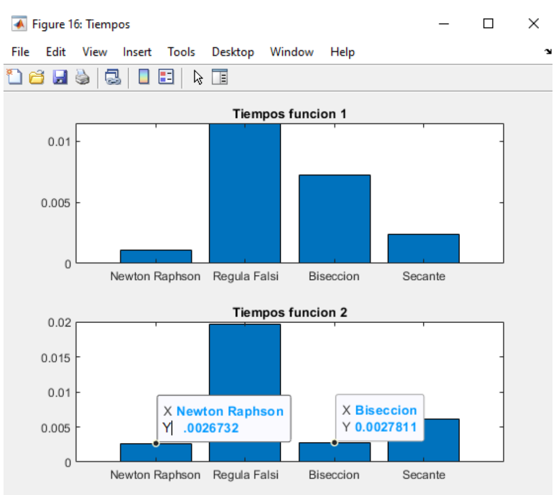

Figura 10: Comparación de tiempos de ejecución para ambas ecuaciones

.

V. Conclusiones

Durante este documento se logró apreciar que 4 de 5 algoritmos numéricos fueron implementados correctamente en el Matlab, siendo estos 4 los referentes a hallar la raíz de una ecuación, se logró establecer que el método más eficiente en cuanto a tiempo y espacio, es el método de Newton-Raphson y aquel que pese a su implementación más sencilla, como es Regula-Falsi, demuestra utilizar más tiempo de ejecución, con respecto a la ecuación 2, se debería incluir alguna restricción para que dentro del logaritmo no existan números menores o iguales a 0 en futuras ocasiones. Con respecto al uso del error mínimo y priori como medidas, se logró utilizar para evaluar y contrastar los distintos métodos, en búsqueda de aquel que presentará el menor en caso de que no se haya encontrado una solución. Otro elemento a considerar para futuras ocasiones es implementación del método de Newton multivariable ya que esté no sé logró implementar correctamente, en resumen, el método de newton-Raphson pesé a requerir la derivada de la función es aquel que será más eficiente dentro de los algoritmos numéricos para encontrar la raíz de una ecuación

Referencias

[1] Mathworks, "¿qué es matlab?" 7/12/2021. [Online]. Available: https://la.mathworks.com/products/matlab.html

[2] J. Andres, "Consulta codi a priori," 7/12/2021. [Online]. Available: https://es.scribd.com/document/412570440/Consulta-Codi-a-Priori

[3] . REAL ACADEMIA ESPAÑOLA, "Algoritmo," *Diccionario de la lengua española, 23.ª ed., [versión 23.4 en línea].*, 7/12/2021.

[4] ——, "Numérico," *Diccionario de la lengua española, 23.ª ed., [versión 23.4 en línea].*, 7/12/2021.

[5] O. Rojas, "Unidad ii: Resolución de ecuaciones no lineales," 7/12/2021. [Online]. Available: https://uvirtual.usach.cl/moodle/pluginfile.php/410917/mod_resource/content/1/II_NoLineal.pdf

Contribución

Conceptualización, M.V.C. y J.G.; Metodología, M.V.C. y J.G.G.; software, M.V.C.; validación, J.G.G.; análisis, M.V.C.; investigación, M.V.C. ; recursos, J.G.G; preparación de datos, M.V.C. y J.G.G.; escritura de borrador, M.V.C. y J.G.G.; revisión de la escritura y formalización, M.V.C.; formato y visualización, J.G.G; supervisión, M.V.C.; Todos los autores y autoras han leído y están de acuerdo con la entrega de este documento, así como el trato de principios éticos.