Documentación: Métodos numéricos raíces reales

D. R. Ramírez, C. Castrillón, J. M. Torres

Febrero de 2021

1 Introducción

El método de la Posición Falsa (también llamado método *Regula Falsi*) es un método numérico que permite aproximar raíces reales de funciones matemáticas continuas en un intervalo cerrado. La solución aproximada depende de la tolerancia que se proporcione como criterio de parada o, en su defecto, un número máximo de iteraciones definido por el usuario del método. A continuación se revelará detalladamente las condiciones necesarias para poder hacer uso del método; la demostración geométrica y algebráica del algoritmo construido; y las comparaciones con otros métodos numéricos que cumplen la misma función. Los resultados expuestos en este documento han sido obtenidos mediante la implementación del algoritmo en el lenguaje de programación Python, con la ayuda de módulos externos como NumPy.

Figure 1: Logo de Python

2 Método de la posición falsa (Regula Falsi)

2.1 Condiciones para aplicar el método

Las condiciones que se deben cumplir para utilizar el método a cabalidad son las siguientes:

- Debe haber un intervalo cerrado en el cual evaluar la función (dentro del que se va a desarrollar la búsqueda de raíces).
- Las imágenes correspondientes a los valores extremos de dicho intervalo deben ser de distinto signo: uno negativo; uno positivo. Esto con el fin de garantizar que, efectivamente, existe al menos una raíz en el intervalo.
- Debe existir al menos una de dos condiciones de parada: una tolerancia dada, o una cantidad máxima de iteraciones a ejecutar, en caso de que la anterior no se cumpla.

2.2 Explicación geométrica del algoritmo

Dadas las condiciones mencionadas con anterioridad, el método se describe de la siguiente manera: la recta que inicia en a y termina en b; y pasa por el punto c (intersecto con el eje x), se aproxima a la función (f(x)) dada. En consecuencia al cambio de tramo —i.e. cambio en el valor de uno de los extremos del intervalo en que se está evaluando la función— que se hace tras cada iteración del algoritmo, la recta cambia su longitud y, de mayor importancia, su pendiente. Por consiguiente, tal punto de intersección (c) varía, y su proximidad a la raíz de la función presente en el intervalo, incrementa. Para ilustrar lo que está sucediendo, se observa que la recta se desplaza hasta que el punto c se acerca tanto como sea posible (teniendo en cuenta la tolerancia dada) a la solución. El valor de c se obtiene mediante despeje al hallar la equivalencia por semejanza de triángulos entre la sección positiva de la recta, y la sección negativa de la misma.

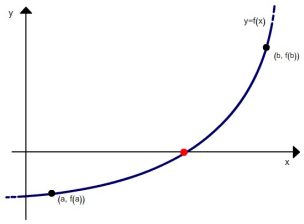

Figure 2: Primera captura Regula Falsi

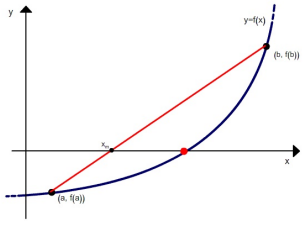

Figure 3: Segunda captura Regula Falsi

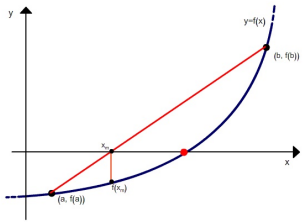

Figure 4: Tercera captura Regula Falsi

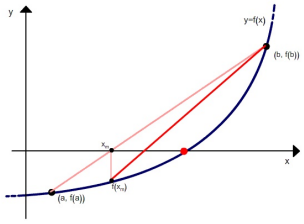

Figure 5: Cuarta captura Regula Falsi

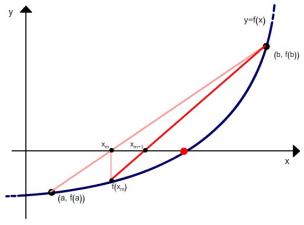

Figure 6: Quinta captura Regula Falsi

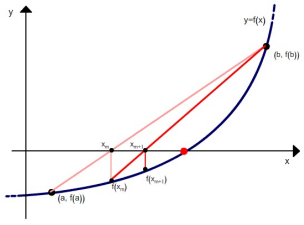

Figure 7: Sexta captura Regula Falsi

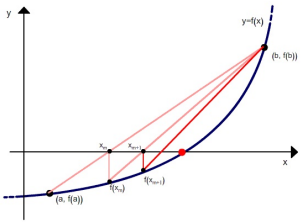

Figure 8: Séptima captura Regula Falsi

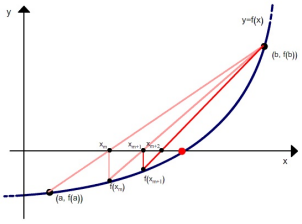

Figure 9: Octava captura Regula Falsi

2.3 Diagrama de flujo

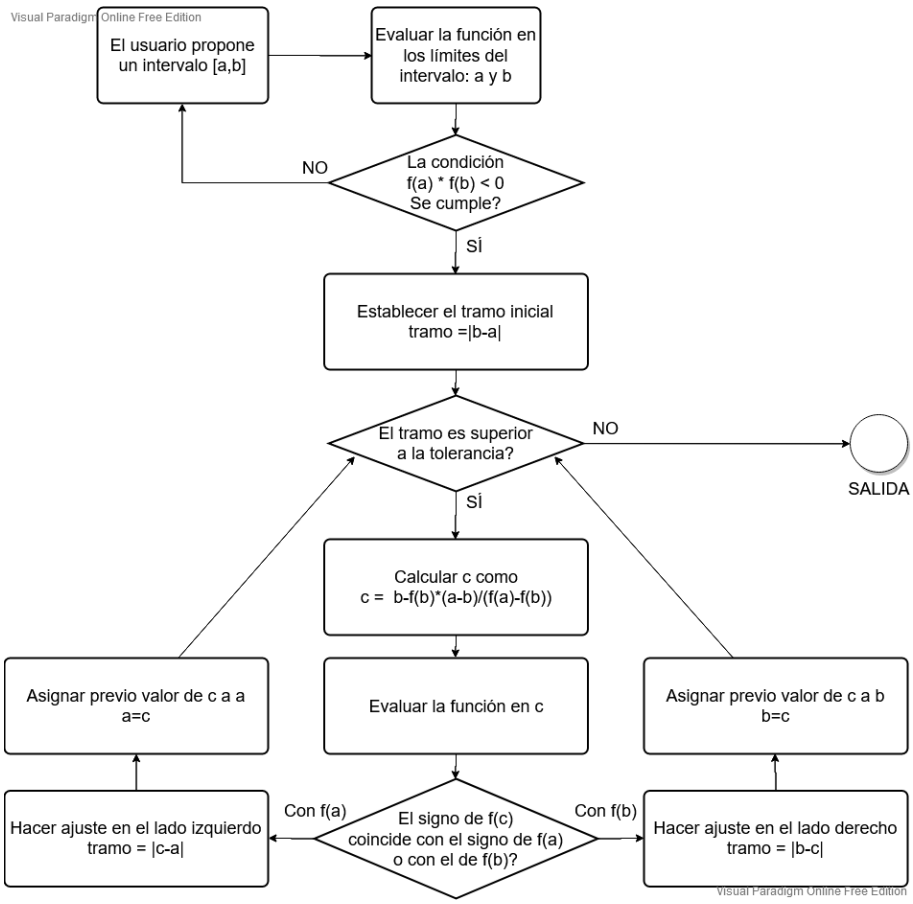

Figure 10: Diagrama de flujo del algoritmo

2.4 Resultados

Los resultados obtenidos dependieron en parte de la tolerancia aplicada. Todas las tolerancias utilizadas en el programa de Python fueron 1E-8, 1E-16, 1E-32 y 1E-56. Los resultados obtenidos con cada una de las tolerancias pueden observarse en la salida del programa. A continuación se muestra la tabla de resultados obtenida para una tolerancia de 1E-8.

- Para la función $f(x) = cos^2(x) - x^2$

	Posición falsa		
Función	Iteración	Aproximación	Error
$f(x) = \cos^2(x)-x^2$	0	-1.6811130581	6.81E-01
	1	-11.9089131021	1.02E+01
	2	2.3874745675	1.43E+01
	3	0.8950464215	1.49E+00
	4	0.7625016127	1.33E-01
	5	0.7432837765	1.92E-02
	6	0.7398571068	3.43E-03
	7	0.7392277049	6.29E-04
	8	0.7391114856	1.16E-04
	9	0.7390900048	2.15E-05
	10	0.7390860338	3.97E-06
	11	0.7390852997	7.34E-07
	12	0.7390851640	1.36E-07
	13	0.7390851389	2.51E-08
	14	0.7390851343	4.64E-09

Figure 11: Tabla de resultados para la primera función

- Para la función $f(x) = xsen(x) - 1$

	Posición falsa		
Función	Iteración	Aproximación	Error
$f(x) = x \cdot \text{sen}(x)-1$	0	-0.5132786531	4.87E-01
	1	0.6867011013	1.20E+00
	2	1.2227923574	7.77E-01
	3	1.1105698721	4.24E-01
	4	1.1141897329	1.09E-01
	5	1.1141571457	3.26E-05
	6	1.1141571409	4.81E-09

Figure 12: Tabla de resultados para la segunda función

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

	Posición falsa		
Función	Iteración	Aproximación	Error
$f(x) = x^3-2x^2+(4/3)x-(8/27)$	0	0.9841269841	1.02E+00
	1	0.9705093834	1.36E-02
	2	0.9586419910	1.19E-02
	3	0.9481678462	1.05E-02
	4	0.9388259887	9.34E-03
	5	0.9304202620	8.41E-03
	6	0.9227997706	7.62E-03
	7	0.9158461416	6.95E-03
	8	0.9094649433	6.38E-03
	9	0.9035797379	5.89E-03
	10	0.8981278580	5.45E-03
	11	0.8930573430	5.07E-03
	12	0.8883246751	4.73E-03
	13	0.8838930779	4.43E-03
	14	0.8797312186	4.16E-03
	15	0.8758122056	3.92E-03
	16	0.8721128031	3.70E-03
	17	0.8686128114	3.50E-03
	18	0.8652945708	3.32E-03
	...	...	...
	55	0.8015062452	9.75E-04

Figure 13: Tabla de resultados para la tercera función

- Para el problema del paracaidista, correspondiente a la función $f(x) = (9.8 * \frac{68.1}{x} * (1 - e^{\frac{-x}{(68.1*10)}})) - 40$

	Posición falsa		
Función	Iteración	Aproximación	Error
f(x) = (9.8*68.1/x)*(1-e^(-x/68.1*10))-40	0	8.529055440	9.53E+00
	1	12.100636010	3.57E+00
	2	13.604891409	1.50E+00
	3	14.260163104	6.55E-01
	4	14.549261541	2.89E-01
	5	14.677484022	1.28E-01
	6	14.734483854	5.70E-02
	7	14.759847893	2.54E-02
	8	14.771139506	1.13E-02
	9	14.776167320	5.03E-03
	10	14.778406250	2.24E-03
	11	14.779403304	9.97E-04
	12	14.779847326	4.44E-04
	13	14.780045066	1.98E-04
	14	14.780133127	8.81E-05
	15	14.780172344	3.92E-05
	16	14.780189809	1.75E-05
	17	14.780197587	7.78E-06
	18	14.780201051	3.46E-06
	19	14.780202593	1.54E-06
	20	14.780203280	6.87E-07
	21	14.780203586	3.06E-07
	22	14.780203722	1.36E-07
	23	14.780203783	6.07E-08
	24	14.780203810	2.70E-08
	25	14.780203822	1.20E-08
	26	14.780203827	5.36E-09

Figure 14: Tabla de resultados para la cuarta función

- Para la ecuación histórica, $f(x) = x^3 - 2x - 5$

	Posición falsa		
Función	Iteración	Aproximación	Error
$f(x) = x^3-2x-5$	0	3.0000000000	1.00E+00
	1	-0.2000000000	8.00E-01
	2	0.5155279503	7.16E-01
	3	1.1843668600	6.69E-01
	4	1.6617402469	4.77E-01
	5	1.9150042387	2.53E-01
	6	2.0251186675	1.10E-01
	7	2.0684926643	4.34E-02
	8	2.0848849245	1.64E-02
	9	2.0909813952	6.10E-03
	10	2.0932351170	2.25E-03
	11	2.0940664036	8.31E-04
	12	2.0943727710	3.06E-04
	13	2.0944856471	1.13E-04
	14	2.0945272298	4.16E-05
	15	2.0945425479	1.53E-05
	16	2.0945481907	5.64E-06
	17	2.0945502693	2.08E-06
	18	2.0945510350	7.66E-07
	19	2.0945513170	2.82E-07
	20	2.0945514209	1.04E-07
	21	2.0945514592	3.83E-08
	22	2.0945514733	1.41E-08
	23	2.094551479	5.19E-09

Figure 15: Tabla de resultados para la quinta función

Nótese que para la tercera función, el resultado está lejos de ser una aproximación válida como respuesta. En esta función, la raíz es un número periódico, y en este caso, el método sigue iterando de forma indefinida y nunca se detiene. Para manejar esta última excepción, se colocó un límite de iteraciones correspondiente al número máximo de iteraciones del método de la bisección, como fue recomendado. En la siguente imagen, que muestra la salida del programa elaborado en Python, se puede apreciar como se aproxima cada vez más a la raíz esperada de la tercera función.

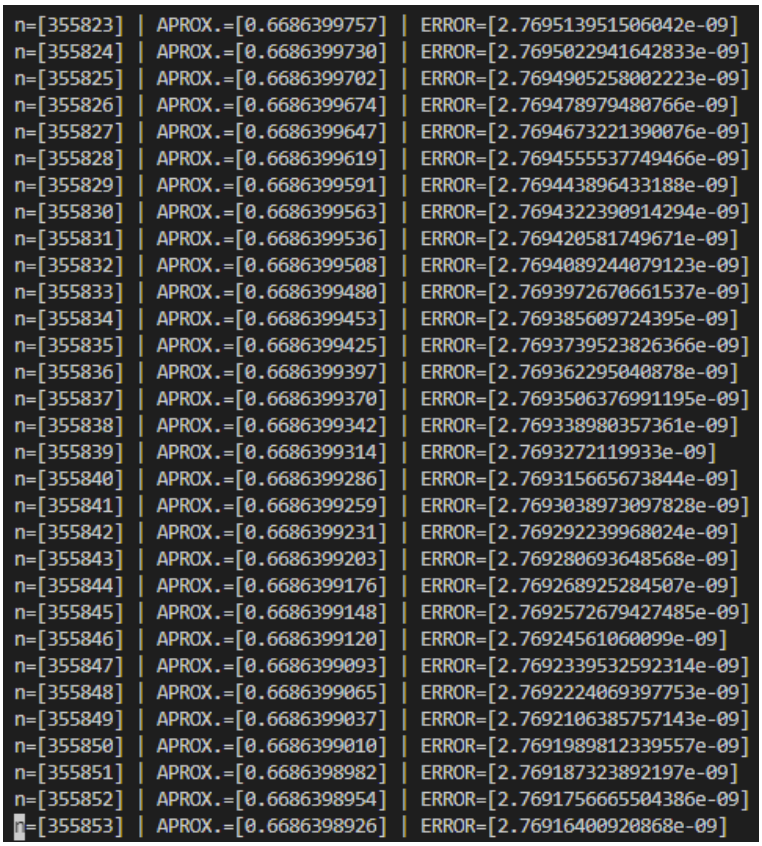

Figure 16: Comportamiento del método para la tercera función

2.4.1 Comportamiento del método

1. Para la función $f(x) = cos^2(x) - x^2$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** La pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el intérprete pueden soportar, pues a medida de que se realizan operaciones aritméticas y se obtienen más de los decimales permitidos, se va perdiendo precisión. Aunque, esto último no es un verdadero problema en la pérdida de significancia, pues Python tiene capcacidad de almacenar 28 posiciones decimales. En caso de que esto llegase a presentar un problema para otras funciones con las que se trabaje, se podría optar por usar el tipo "decimal" de Python.
   - **Número de iteraciones:** El método necesitó de 14 iteraciones para poder llegar a una aproximación óptima. El rendimiento del método

en cuanto al número de iteraciones que tiene que hacer para poder llegar a la solución aproximada depende plenamente de la tolerancia escogida. Se puede notar que entre 2 de las tolerancias más pequeñas (1E-32 y 1E-56), no se nota realmente un cambio en cuanto al número de iteraciones.

- **Convergencia:** Para esta función, el algoritmo garantiza la convergencia. Sin embargo, la aproximación obtenida en la segunda iteración no es muy coherente con las obtenidas en el resto de iteraciones. Si se observa detenidamente, esta segunda aproximación implica una disminución drástica respecto al rango de valores que se llevaba hasta el momento y con las aproximaciones posteriores. La ultima iteración resulta en una aproximación precisa que equivale a 0.7390851343

2. Para la función $f(x) = xsen(x) - 1$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** Como en el caso de la primera función, la perdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el interprete pueden soportar.
   - **Número de iteraciones:** El método necesitó de 6 iteraciones para poder llegar a una aproximación óptima. El numero de iteraciones con tolerancia 1E-32 y 1E-56 es exactamente el mismo. Adicionalmente, el hecho de que esta función sea periódica no implica realmente ningún cambio en cuanto al número de iteraciones, se puede observar que con esta función se toma incluso menos iteraciones para llegar a la solución que en cualquiera de los otros dos casos presentes.
   - **Convergencia:** Para esta función se logra la convergencia. Los valores se van acercando unos a otros cada vez más con una amplitud más reducida hasta que se llega al valor deseado. Las aproximaciones en cada iteración tienen coherencia con base en el resultado que se quiere obtener al final. La ultima iteración resulta en una aproximación precisa que equivale a 1.1141571409

3. Para la función $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** La pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el interprete pueden soportar.
   - **Número de iteraciones:** En este último caso, se tuvo que recurrir a la inclusión de una condición adicional en la sentencia que evalúa la tolerancia (i.e. la condición de parada). El resultado, en este caso, también es un número real, pero corresponde a un número periódico, lo que no sucedía con las funciones anteriores. Como el número que

se debe obtener al final es periódico, el método de la posición falsa, sin la condición adicional que limita el numero de iteraciones, sigue tratando de aproximar la solución hasta que se llega a 0,6 (periódico), sin dejar de iterar. Para manejar esta exepeción, se buscó el numero máximo de iteraciones para dar con la solución mediante el método de la bisección, y se puso ese valor como límite de iteración. Este último valor corresponde a 56 iteraciones.

- **Convergencia:** Para esta función definitivamente hay convergencia, después de cierto número de iteraciones, el resultado aproximado se estabiliza, pero no deja de iterar a no ser que se incluyan condiciones adicionales.

4. Para el problema del paracaidista, cuya función puede ser descrita como $v(c) = 0 = 9.8 * \frac{68.1}{c} * (1 - e^{-}(\frac{c}{68.1}) * 10)$, el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** La pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el intérprete pueden soportar
   - **Número de iteraciones:** El número de iteraciones máximo, correspondiente a los casos cuyo error relativo es 1E*-8, 1E*-16, 1E*-32, y 1E*-56, es igual a 19.
   - **Convergencia:** Para esta función, definitivamente hay convergencia, despues de cierto número de iteraciones el resultado aproximado se estabiliza y se aproxima a 14.7802038317.

5. Para la función $f(x) = x^3 - 2x - 5$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** Como en el caso del paracaidista, la pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación puede soportar.
   - **Número de iteraciones:** El método necesitó de 24 iteraciones para poder llegar a una aproximación óptima.
   - **Convergencia:** Para esta última función, definitivamente hay convergencia, despues de cierto número de iteraciones el resultado aproximado se estabiliza y se aproxima a 2.094551479.

2.4.2 Observaciones del método cuando la función tiene dos raices

Si la función llega a tener más de 1 raíz, el método encuentra la solución acotada entre el intervalo; y si llega a existir más de 1 raíz dentro del intervalo, el método de la posición falsa retornará la raíz más próxima al punto de intersección (con el eje horizontal) inicial.

2.4.3 Observaciones del método cuando la función es periódica, par o impar

Si la función es periódica y el intervalo utilizado en el algoritmo abarca más de 1 solución, entonces se retornará la primera solución que se encuentre, independientemente de que las funciones periódicas suelan tener infinitas soluciones en todo su dominio. Adicionalmente, para la función periódica con la que se trabajó, es importante mencionar que fue la que menos iteraciones tomó para llegar a la aproximación óptima. Por otro lado, si la función es par, la observación sería que para que se pueda escojer un intervalo válido para el método, el rango de la función debe albergar tanto valores negativos como positivos.

2.4.4 Gráficas de relación entre Ei+1 y Ei

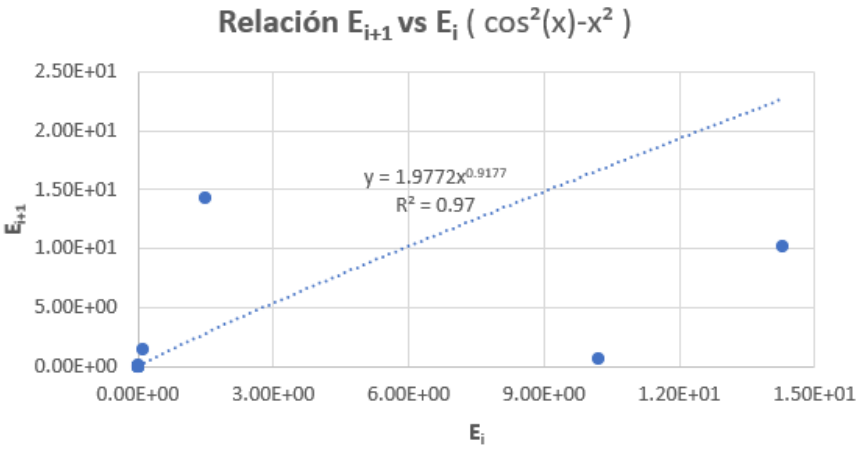

Figure 17: Gráfica que relaciona Ei+1 con Ei para la primera función

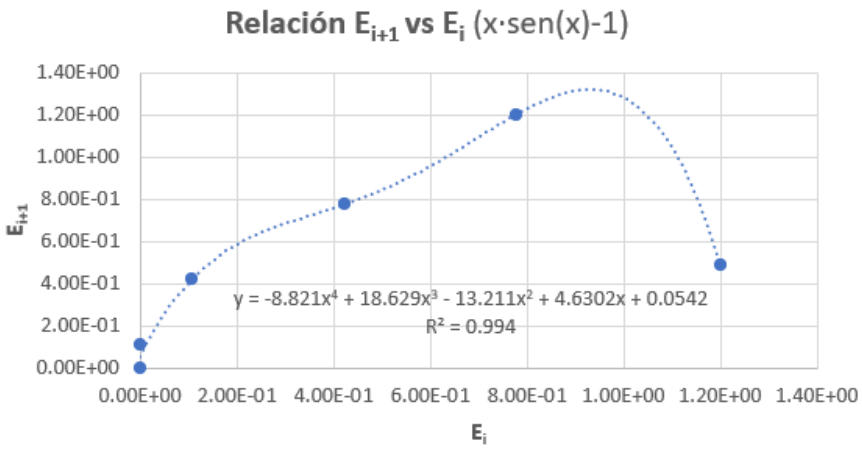

Figure 18: Gráfica que relaciona Ei+1 con Ei para la segunda función

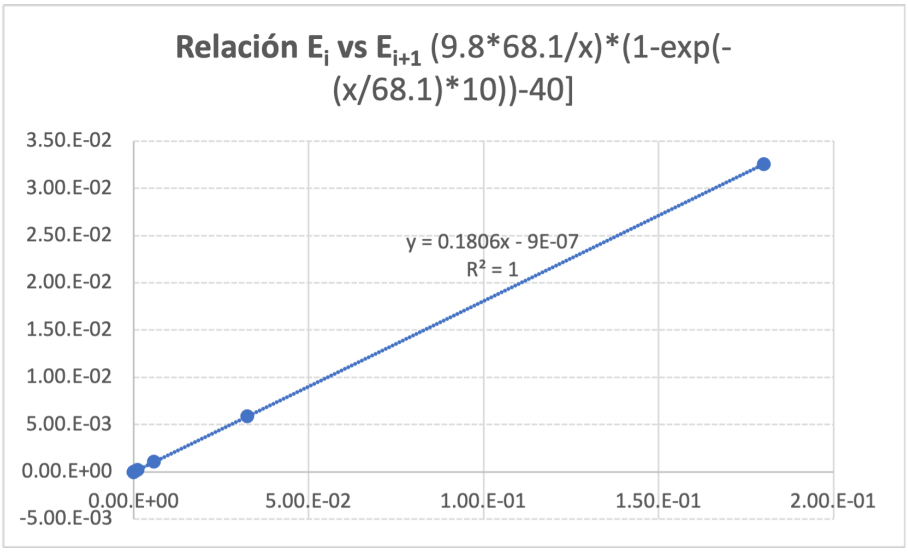

Figure 19: Gráfica que relaciona Ei+1 con Ei para la tercera función

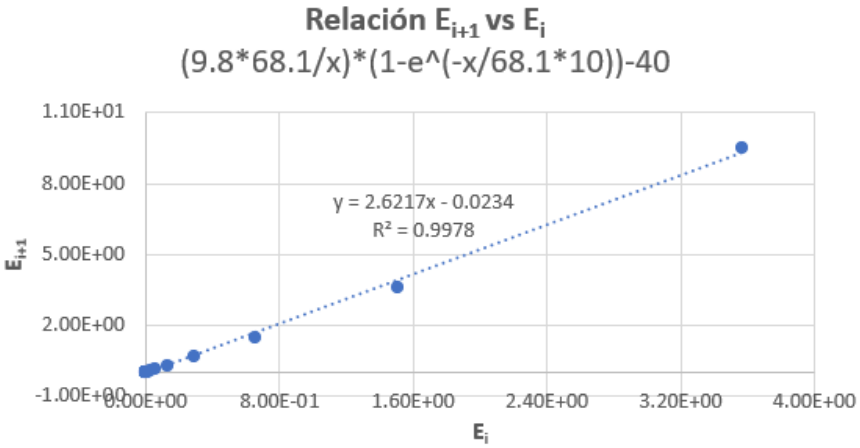

Figure 20: Gráfica que relaciona Ei+1 con Ei para la cuarta función

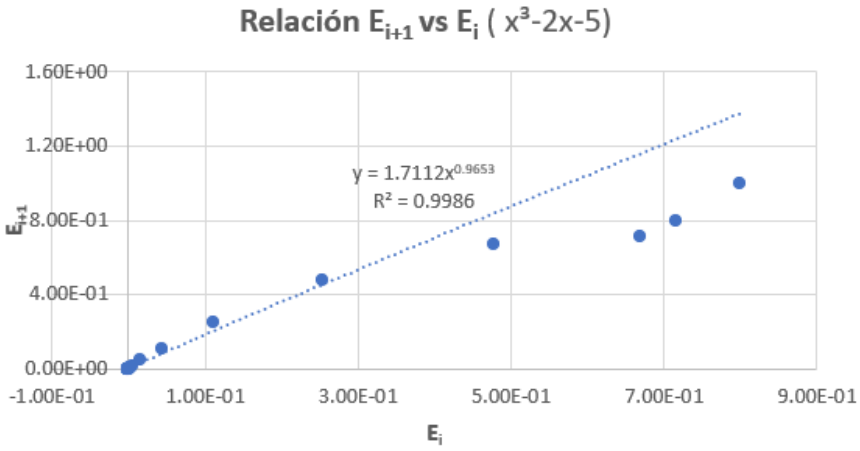

Figure 21: Gráfica que relaciona Ei+1 con Ei para la quinta función

2.5 Gráfica para visualizar el comportamiento del método

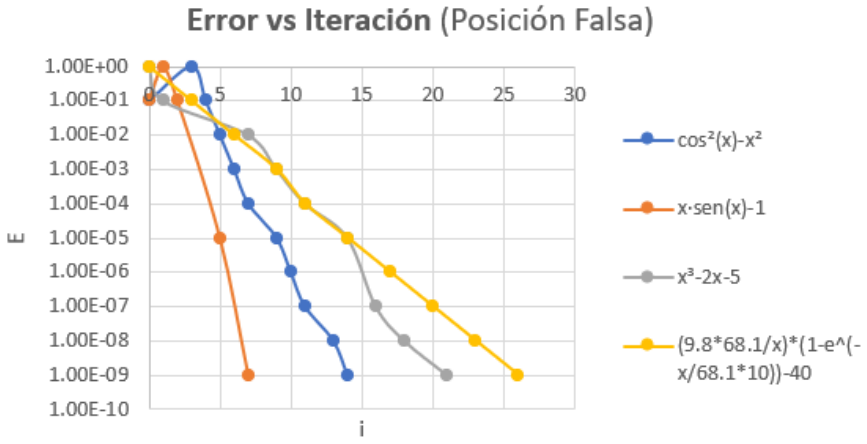

Figure 22: Gráfica error-iteración que revela el comportamiento del método por cada función

A partir de la Figura 22, se puede concluir que el método fue más eficiente con la segunda función ($f(x) = xsen(x) - 1$) a comparación del resto de funciones analizadas. Por otro lado, la aproximación de la raíz de la última función ($v(c) = 0 = 9.8 * \frac{68.1}{c} * (1 - e^{-}(\frac{c}{68.1}) * 10)$, es la que mayor cantidad de iteraciones toma en estimar.

2.6 Comparación con método de la bisección

El método de la bisección arroja los siguientes resultados para la función $f(x) = x^3 - x^2 + \frac{4}{3}x - \frac{8}{27}$: **Número de iteraciones:** 56 iteraciones **Aproximación final:** 0.6666717529296875 A diferencia del método de la posición falsa, el método de la bisección, en este caso, no necesita de una cota superior explícita para la cantidad de iteraciones que debe desarrollar, su parámetro de parada, en cambio, es el error permitido para el ejercicio (en este caso igual a 1E-56). Se concluye, entonces, que el método de la bisección, debido precisamente al hecho de converger a la solución aproximada, es más eficiente que el método de la Posición Falsa.

2.7 Comparación con método de Taylor

El método de Taylor no es un método numérico para encontrar raíces de una función, y aunque puede ser utilizado para aproximar el valor correspondiente para el que la función se hace igual a cero (i.e. la raíz), la comparación con

los métodos de búsqueda de raíces resulta poco relevante dada su distinta naturaleza.

3 Método de delta cuadrado de Aitken

3.1 Condiciones para aplicar el método

El Proceso de Delta Cuadrado de Aitken es un método numérico cuyo objetivo es acelerar la convergencia de una sucesión de convergencia lineal. En concordancia con el alcance y el objetivo de este documento, se estudiará en particular si se puede aplicar el proceso al método de Posición Falsa y, de ser así, cómo aplicarlo.

3.2 Diagrama de flujo

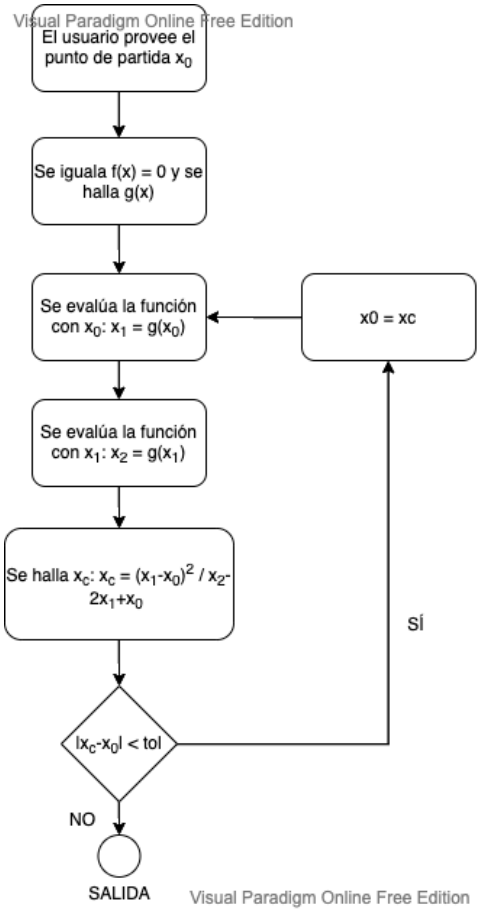

Figure 23: Diagrama de flujo del algoritmo

3.3 Resultados

Los resultados que se han obtenido dan cuenta de la mejora que este proceso de aceleración brinda para métodos numéricos de convergencia lineal. A continuación se muestra las tablas de aproximaciones y errores para cada iteración, en cada una de las funciones expuestas con anterioridad.

- Para la función $f(x) = cos^2(x) - x^2$

Función	Iteración	Aproximación	Error
$f(x) = \cos^2(x) - x^2$	0	0.7280103614676170	2.72E-01
	1	0.7390669669086730	1.11E-02
	2	0.7390851331660750	1.82E-05
	3	0.7390851332151600	4.91E-11

Figure 24: Tabla de resultados para la primera función

- Para la función $f(x) = xsen(x) - 1$

$f(x) = x \cdot \text{sen}(x) - 1$	0	1.11872920740582	1.19E-01
	1	1.11416320423672	4.57E-03
	2	1.11415714088267	6.06E-06
	3	1.11415714087193	1.07E-11
	4	1.11415714087193	0.00E+00

Figure 25: Tabla de resultados para la segunda función

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

$f(x) = x^3 - 2x^2 + (4/3)x - (8/27)$	0	0.844444444444444	1.56E-01
	1	0.782370945060495	6.21E-02
	2	0.743996079422068	3.84E-02
	3	0.718643733605188	2.54E-02
	4	0.701601657259061	1.70E-02
	5	0.690110927446719	1.15E-02
	6	0.682373223212298	7.74E-03
	7	0.677174502904048	5.20E-03
	8	0.673689012353133	3.49E-03
	9	0.671356069255352	2.33E-03
	10	0.669796494856593	1.56E-03
	11	0.668754795963442	1.04E-03
	12	0.668059533604268	6.95E-04
	13	0.667596419265461	4.63E-04
	14	0.667288739831335	3.08E-04
	15	0.667099642367595	1.89E-04
	16	0.666798827765142	3.01E-04

Figure 26: Tabla de resultados para la tercera función

- Para el problema del paracaidista, correspondiente a la función

f(x) = (9.8*68.1/x)*(1-e^(-x/68.1*10))-40	1	14.7857785	7.86E-01
	2	14.78020884	5.57E-03
	3	14.78020859	2.47E-07

Figure 27: Tabla de resultados para la cuarta función

- Para la ecuación historica

$f(x) = x^3-2x-5$	0	2.137096774	1.37E-01
	1	2.102249297	3.48E-02
	2	2.094809605	7.44E-03
	3	2.094551773	2.58E-04
	4	2.094551482	2.92E-07
	5	2.094551482	3.73E-13

Figure 28: Tabla de resultados para la quinta función

3.3.1 Comportamiento del método

1. Para la función $f(x) = cos^2(x) - x^2$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** La pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el intérprete pueden soportar, pues a medida de que se realizan operaciones aritméticas y se obtienen más de los decimales permitidos, se va perdiendo precisión. Aunque, esto último no es un verdadero problema en la pérdida de significancia, pues Python tiene capcacidad de almacenar 28 posiciones decimales. En caso de que esto llegase a presentar un problema para otras funciones con las que se trabaje, se podría optar por usar el tipo "decimal" de Python.
   - **Número de iteraciones:** El método necesitó de 4 iteraciones para poder llegar a una aproximación óptima. El rendimiento del método depende plenamente del punto de partida que se escoja, más aún que la tolerancia escogida. Al escoger un punto de partida mas alejado, gasta mas iteraciones para llegar al resultado.
   - **Convergencia:** Para esta funcion, el algoritmo garantiza la convergencia. Sin embargo, al emplear la fórmula del método en la quinta iteración, el denominador tiende a 0 y este deja de funcionar.

2. Para la función $f(x) = xsen(x) - 1$ el algoritmo se comportó de la siguiente manera:

- **Pérdida de significancia:** Como en el caso de la primera funcion, la perdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el intérprete pueden soportar.
- **Número de iteraciones:** El método gasta 4 iteraciones con una tolerancia de 1E-8, y una más para 1E-16, ya que en la última iteración el error es 0 .
- **Convergencia:** Para esta función, las aproximaciones en cada iteración tienen coherencia con base en el resultado que se quiere obtener y al final la diferencia entre el último resultado y el inmediatamente anterior es 0.

3. Para la función $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** La pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el interprete pueden soportar.
   - **Número de iteraciones:** Para este caso, el número de iteraciones fue 17 y no continuó debido a que el denominador del metodo tendía a 0. Por otro lado, la aproximación no es tan precisa (da 0,6679...). Comparándola con el resultado ideal, está alejada.
   - **Convergencia:** Hay convergencia aunque al final el metodo deja de funcionar.

4. Para el problema del paracaidista, cuya función puede ser descrita como $v(c) = 0 = 9.8 * \frac{68.1}{c} * (1 - e^{-}(\frac{c}{68.1}) * 10)$, el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** La pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación y el intérprete pueden soportar
   - **Número de iteraciones:** El número de iteraciones máximo, correspondiente a los casos cuyo error relativo es 1E*-8, 1E*-16, 1E*-32, y 1E*-56, es igual a 19.
   - **Convergencia:** Para esta última función definitivamente hay convergencia, y su resultado aproximado es 14.780208593679466.

5. Para la función $f(x) = x^3 - 2x - 5$ el algoritmo se comportó de la siguiente manera:

   - **Pérdida de significancia:** Como en los demás casos, la pérdida de significancia se vería reflejada esencialmente en el número de posiciones decimales que el lenguaje de programación puede soportar.
   - **Número de iteraciones:** El método gasta 6 iteraciones con una tolerancia de 1E-8, y una más para 1E-16, ya que en la última iteración el error es igual a 0.

- **Convergencia:** Para esta función, la aproximación en cada iteración tiene coherencia, teniendo en cuenta el resultado que se quiere obtener, y, al final, la diferencia entre el último resultado y el inmediatamente anterior es 0.

3.3.2 Observaciones del método cuando la función tiene dos raices

Si la función tiene dos raíces, el método presenta aquella cuyo resultado es positivo.

3.3.3 Observaciones del método cuando la función es periodica, par o impar

Si la función es periódica, y el término de partida utilizado en el algoritmo es el ideal, funciona bien, dando resultados muy aproximados, pero para uno de los casos el denominador tiende a 0 y el método ya no funciona.

3.3.4 Gráficas de relación entre Ei+1 y Ei

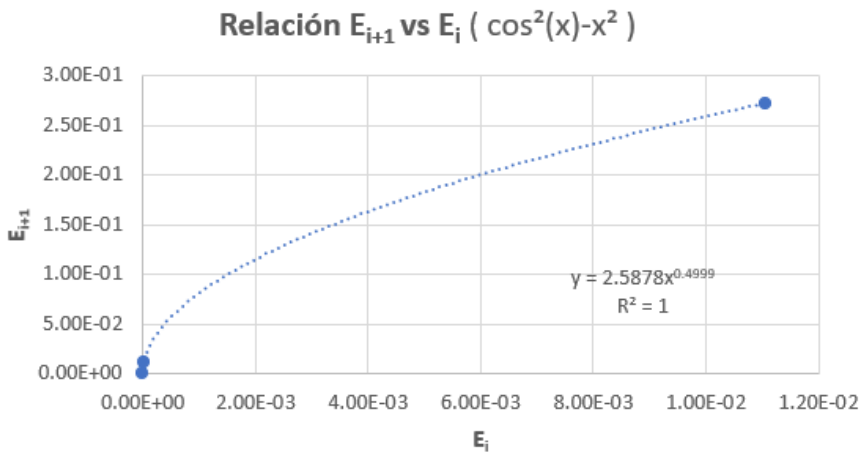

Figure 29: Gráfica que relaciona Ei+1 con Ei para la primera función

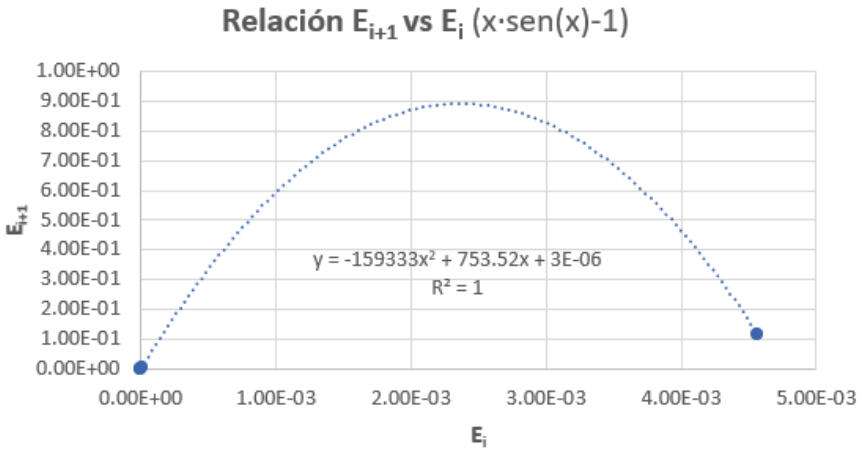

Figure 30: Gráfica que relaciona Ei+1 con Ei para la segunda función

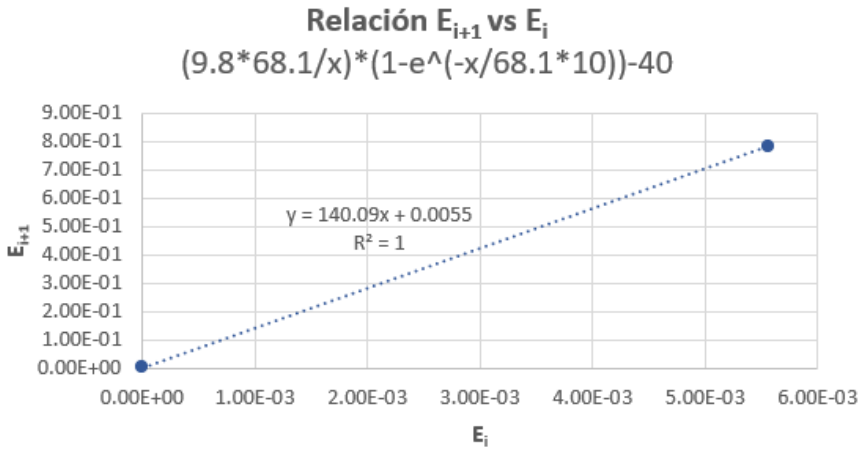

Figure 31: Gráfica que relaciona Ei+1 con Ei para la cuarta función

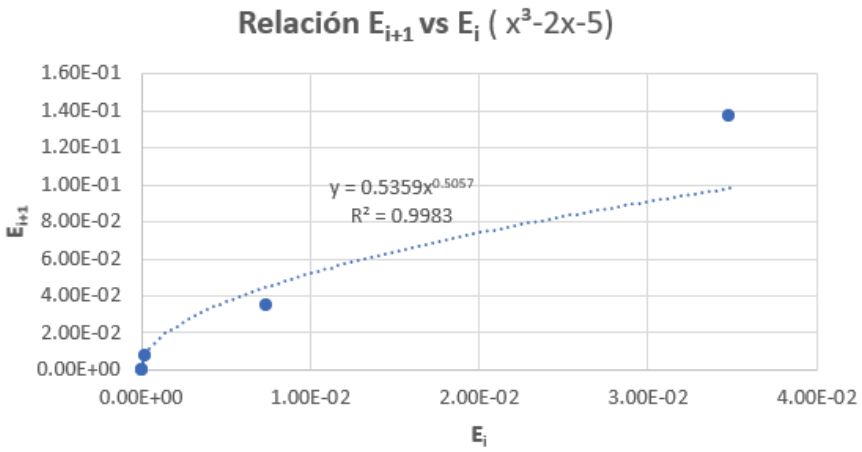

Figure 32: Gráfica que relaciona Ei+1 con Ei para la quinta función

3.4 Gráfica para visualizar el comportamiento del método

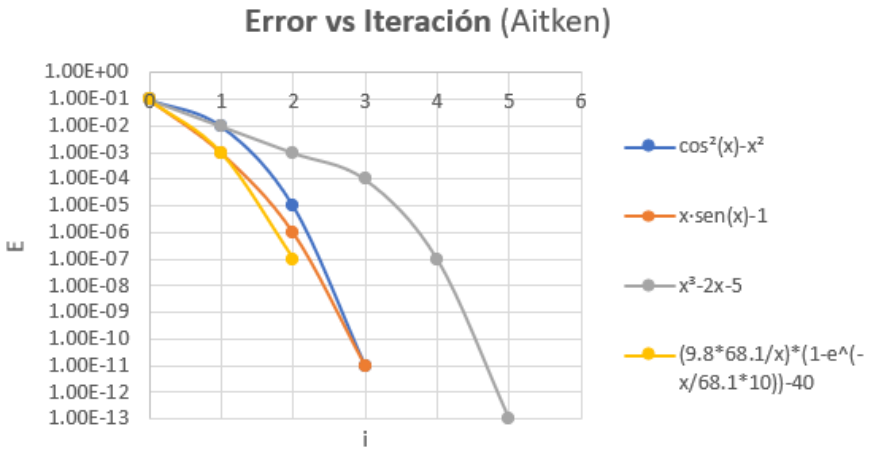

Figure 33: Gráfica error-iteración que revela el comportamiento del método por cada función

Como se puede observar en la Figura 28, el metodo fue más eficiente con la función $v(c) = 0 = 9.8 * \frac{68.1}{c} * (1 - e^{-}(\frac{c}{68.1}) * 10$ y menos eficiente con la función $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$.

3.5 Comparación con método de la bisección

El método de la bisección arroja los siguientes resultados para la función $f(x) = x^3 - x^2 + \frac{4}{3}x - \frac{8}{27}$: **Número de iteraciones:** 56 iteraciones **Aproximación final:** 0.6666717529296875 En comparación con el método de la bisección, el método de delta cuadrado de Aitken, en este caso, es mucho más eficiente, pues toma menos de un tercio del tiempo en llegar a una aproximación aceptable (con un error de 3.01E-4).

3.6 Comparación con método de Taylor

El método de Taylor no es un método numérico para encontrar raíces de una función, y aunque puede ser utilizado para aproximar el valor correspondiente para el que la función se hace igual a cero (i.e. la raíz), la comparación con los métodos de búsqueda de raíces resulta poco relevante dada su distinta naturaleza.

4 Comparación entre los dos métodos estudiados

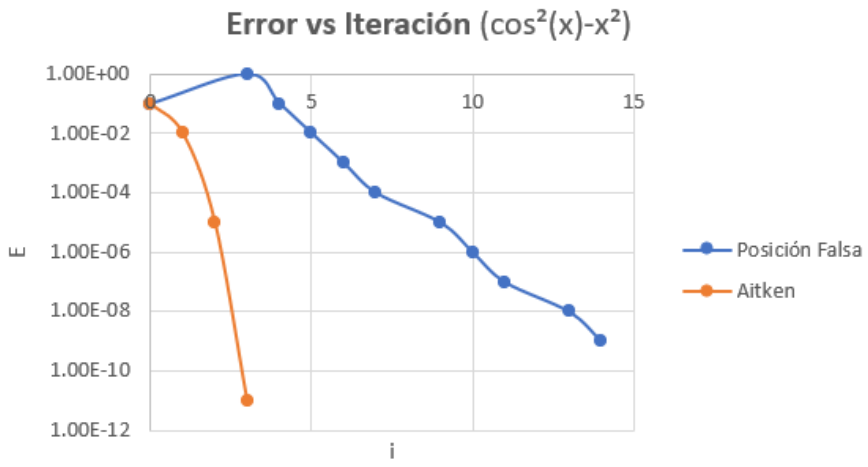

Figure 34: Gráfica Error vs Iteración para la primera función

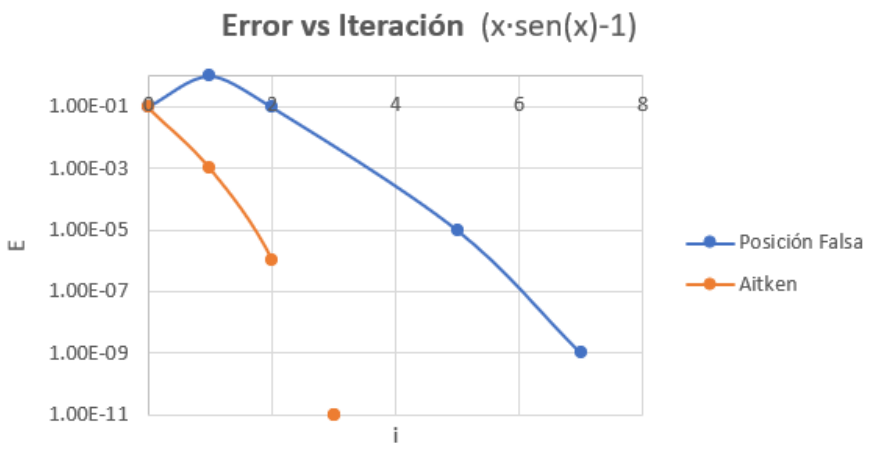

Figure 35: Gráfica Error vs Iteración para la segunda función

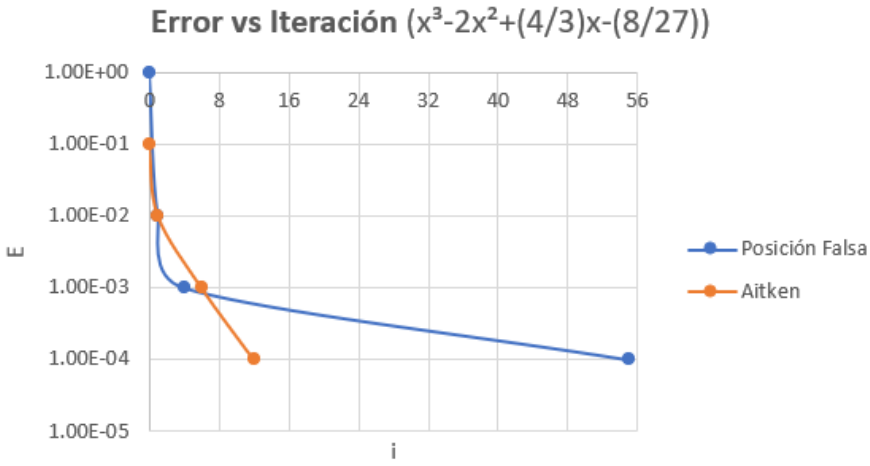

Figure 36: Gráfica Error vs Iteración para la tercera función

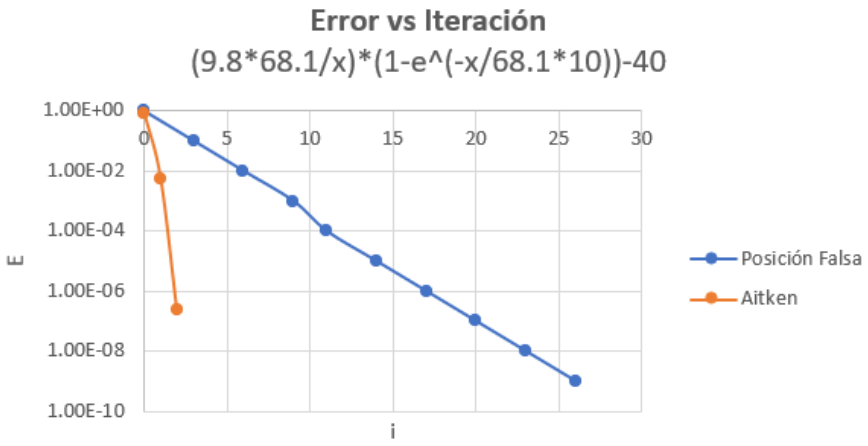

Figure 37: Gráfica Error vs Iteración para la cuarta función

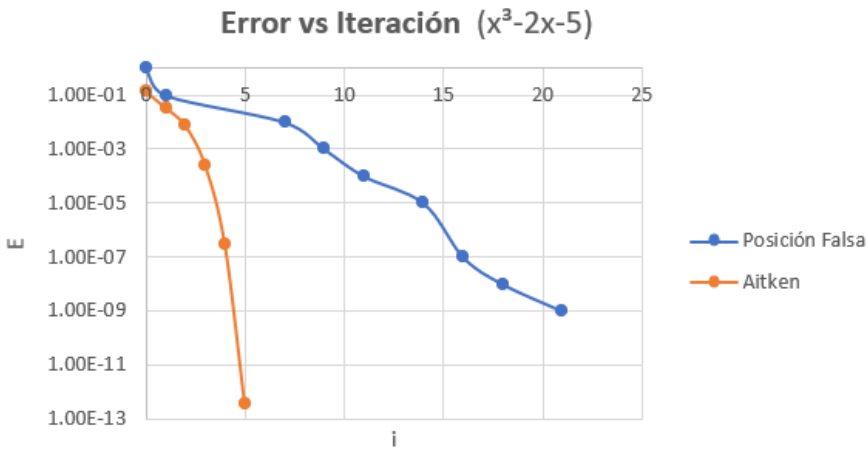

Figure 38: Gráfica Error vs Iteración para la quinta función