

Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Alumno: Clyde Neil Paricahua Pari

Trabajo Encargado N°06 - Métodos Numéricos para Raíces

Método de Bisección

Descripción El método de bisección es un procedimiento iterativo para encontrar raíces de una función continua $f(x)$ en un intervalo cerrado $[a, b]$. Se fundamenta en el teorema del valor intermedio, que garantiza la existencia de una raíz si la función cambia de signo en los extremos del intervalo. Su fórmula para encontrar el punto medio en cada iteración es:

$$c_n = \frac{a_n + b_n}{2}$$

Su convergencia es lineal y garantizada, aunque más lenta que otros métodos.

Código en JavaScript

```
1
2 const readline = require('readline');
3 const math = require('mathjs');
4
5 // Crear la interfaz para leer y escribir en la consola
6 const rl = readline.createInterface({
7   input: process.stdin,
8   output: process.stdout
9 });
10
11 function runBisectionConsole() {
12
13   rl.question("Ingrese la función f(x): ", (funcStr) => {
14     if (!funcStr) {
15       console.log("No se ingresó ninguna función.");
16       rl.close();
17       return;
18     }
19
20     // Se eliminan todos los espacios de la entrada del usuario.
21     const cleanFuncStr = funcStr.replace(/\s*/g, '').replace(/\s/g, '');
22
23
24     let f;
25     try {
26
27       const node = math.parse(cleanFuncStr);
28       const code = node.compile();
29       f = x => code.evaluate({ x: x });
30     } catch (error) {
31       console.error("Error en la función ingresada:", error.message);
32       rl.close();
33       return;
34     }
35
36   });
37 }
```

```
36 rl.question(" Desea encontrar una raíz con el método de Bisección? (s/n): ",
37   (op) => {
38     if (op.toLowerCase() !== 's') {
39       console.log("No se aplicó el método de Bisección.");
40       rl.close();
41       return;
42     }
43
44
45     rl.question("Ingrese el extremo izquierdo del intervalo (a): ", (a_input) =>
46       {
47         rl.question("Ingrese el extremo derecho del intervalo (b): ", (b_input)
48           => {
49           const a_start = parseFloat(a_input);
50           const b_start = parseFloat(b_input);
51
52           if (isNaN(a_start) || isNaN(b_start)) {
53             console.error("Error: Los valores del
54               intervalo deben ser numéricos.");
55             rl.close();
56             return;
57           }
58
59           if (f(a_start) * f(b_start) > 0) {
60             console.warn("\n No hay cambio de signo en el intervalo
61               . Intente con otro.");
62             rl.close();
63             return;
64           }
65
66           const tol = 1e-6;
67           const max_iter = 100;
68           let a = a_start;
69           let b = b_start;
70           let converged = false;
71
72           console.log("\nIteración | a | b | c | f(c) | Error");
73           console.log("-----");
74
75           for (let i = 1; i <= max_iter; i++) {
76             const c = (a + b) / 2;
77             const fc = f(c);
78             const error = Math.abs(b - a) / 2;
79
80             const iterText = i.toString().padStart(9);
81             const aText = a.toFixed(6).padStart(12);
82             const bText = b.toFixed(6).padStart(12);
83             const cText = c.toFixed(6).padStart(12);
84             const fcText = fc.toFixed(6).padStart(12);
85             const errorText = error.toFixed(6).padStart(12);
86             console.log(`${iterText} | ${aText} | ${bText} | ${cText}
87               | ${fcText} | ${errorText}`);
88
89             if (Math.abs(fc) < tol || error < tol) {
90               console.log(`\nRaíz aproximada encontrada: ${c.toFixed(6)}
91                 `);
92               console.log(` Iteraciones realizadas: ${i}`);
93               converged = true;
94               break;
95             }
96           }
97         }
98       )
99     )
100   }
```

```

94
95         if (f(a) * fc < 0) {
96             b = c;
97         } else {
98             a = c;
99         }
100     }
101
102     if (!converged) {
103         console.warn('\n No se alcanzó la convergencia
104             despu s de ${max_iter} iteraciones. ');
105     }
106
107     rl.close();
108     });
109 });
110 });
111 });
112 }
113
114 runBisectionConsole();

```

Ejemplo de Ejecución

Para la función:

$$f(x) = x^3 - 2x - 5$$

y el intervalo inicial $[-5, 5]$, se obtiene el siguiente resultado:

```

olioweb/programs/Iterativos /Biseccion.js"
Ingrese la función f(x): x^3 - 2*x - 5
¿Desea encontrar una raíz con el método de Bisección? (s/n): s
Ingrese el extremo izquierdo del intervalo (a): -5
Ingrese el extremo derecho del intervalo (b): 5

```

Iteración	a	b	c	f(c)	Error
1	-5.000000	5.000000	0.000000	-5.000000	5.000000
2	0.000000	5.000000	2.500000	5.625000	2.500000
3	0.000000	2.500000	1.250000	-5.546875	1.250000
4	1.250000	2.500000	1.875000	-2.158203	0.625000
5	1.875000	2.500000	2.187500	1.092529	0.312500
6	1.875000	2.187500	2.031250	-0.681610	0.156250
7	2.031250	2.187500	2.109375	0.166836	0.078125
8	2.031250	2.109375	2.070313	-0.266864	0.039063
9	2.070313	2.109375	2.089844	-0.052406	0.019531
10	2.089844	2.109375	2.099609	0.056614	0.009766
11	2.089844	2.099609	2.094727	0.001954	0.004883
12	2.089844	2.094727	2.092285	-0.025263	0.002441
13	2.092285	2.094727	2.093506	-0.011664	0.001221
14	2.093506	2.094727	2.094116	-0.004857	0.000610
15	2.094116	2.094727	2.094421	-0.001452	0.000305
16	2.094421	2.094727	2.094574	0.000251	0.000153
17	2.094421	2.094574	2.094498	-0.000600	0.000076
18	2.094498	2.094574	2.094536	-0.000175	0.000038
19	2.094536	2.094574	2.094555	0.000038	0.000019
20	2.094536	2.094555	2.094545	-0.000068	0.000010
21	2.094545	2.094555	2.094550	-0.000015	0.000005
22	2.094550	2.094555	2.094553	0.000012	0.000002
23	2.094550	2.094553	2.094551	-0.000002	0.000001
24	2.094551	2.094553	2.094552	0.000005	0.000001

```

Raíz aproximada encontrada: 2.094552
Iteraciones realizadas: 24

```

Figura 1: Enter Caption

La raíz aproximada obtenida es:

$$x \approx 2,094552$$

Método de Punto Fijo

Descripción El método de Punto Fijo busca soluciones a ecuaciones de la forma $f(x) = 0$ reescribiéndolas en una forma equivalente $x = g(x)$. Partiendo de un valor inicial x_0 , se genera una secuencia de aproximaciones usando la fórmula de iteración:

$$x_{n+1} = g(x_n)$$

Para asegurar la convergencia, es crucial que el valor absoluto de la derivada de la función de iteración sea menor que 1, es decir, $|g'(x)| < 1$, en un intervalo que contenga la raíz.

Código en JavaScript

```
1 const readline = require('readline');
2 const math = require('mathjs');
3
4 const rl = readline.createInterface({
5   input: process.stdin,
6   output: process.stdout
7 });
8
9 function runFixedPointConsole() {
10   console.log("--- M TODO DEL PUNTO FIJO ---");
11   console.log("Recuerde que f(x) = 0 se reescribe como x = g(x)");
12
13   rl.question("Ingrese la funci n original f(x): ", (funcStr) => {
14     rl.question("Ingrese la funci n iterativa g(x): ", (gStr) => {
15       if (!funcStr || !gStr) {
16         console.log("Debe ingresar ambas funciones, f(x) y g(x).");
17         rl.close();
18         return;
19       }
20
21       // Limpia la entrada del usuario para eliminar "f(x) =" o "g(x) ="
22       let finalFuncStr = funcStr.includes('=') ? funcStr.split('=')[1] : funcStr;
23       let finalGStr = gStr.includes('=') ? gStr.split('=')[1] : gStr;
24
25       const cleanFuncStr = finalFuncStr.replace(/\*\*/g, '^').replace(/\s/g, '');
26       const cleanGStr = finalGStr.replace(/\*\*/g, '^').replace(/\s/g, '');
27
28       let f, g;
29       try {
30         const fNode = math.parse(cleanFuncStr);
31         const gNode = math.parse(cleanGStr);
32         const fCode = fNode.compile();
33         const gCode = gNode.compile();
34         f = x => fCode.evaluate({ x: x });
35         g = x => gCode.evaluate({ x: x });
36       } catch (error) {
37         console.error("Error en la sintaxis de una de las funciones:", error.
38           message);
39         rl.close();
40         return;
41       }
42
43       rl.question(" Desea aplicar el m todo de Punto Fijo? (s/n): ", (op) => {
44         if (op.toLowerCase() !== 's') {
45           console.log("No se aplic el m todo de Punto Fijo.");
46           rl.close();
47           return;
48         }
49       });
50     });
51   });
52 }
```

```
49
50     rl.question("Ingrese el valor inicial x0: ", (x0_input) => {
51         let x0 = parseFloat(x0_input);
52
53         if (isNaN(x0)) {
54             console.error("Error: El valor inicial debe ser num rico.");
55             rl.close();
56             return;
57         }
58
59         const tol = 1e-6;
60         const max_iter = 100;
61         let converged = false;
62
63         console.log("\nIteraci n |      x0 |      g(x0) |      f(x0)|
64             Error");
65         console.log("-----");
66
67         for (let i = 1; i <= max_iter; i++) {
68             const x1 = g(x0);
69             const error = Math.abs(x1 - x0);
70             const f_val = f(x0);
71
72             const iterText = i.toString().padStart(9);
73             const x0Text = x0.toFixed(6).padStart(11);
74             const x1Text = x1.toFixed(6).padStart(13);
75             const fText = f_val.toFixed(6).padStart(15);
76             const errorText = error.toExponential(2).padStart(10);
77
78             console.log(`${iterText} | ${x0Text} | ${x1Text} | ${fText} | ${
79                 errorText}`);
80
81             if (error < tol) {
82                 console.log(`\nRa z aproximada encontrada: ${x1.toFixed(6)
83                     }`);
84                 console.log(`Iteraciones realizadas: ${i}`);
85                 converged = true;
86                 break;
87             }
88
89             x0 = x1;
90         }
91
92         if (!converged) {
93             console.warn(`\nNo se alcanz la
94                 convergencia despu s de ${max_iter} iteraciones.`);
95         }
96
97         rl.close();
98     });
99 }
100
101 runFixedPointConsole();
```

Ejemplo de Ejecución

Para la función:

$$f(x) = x^2 - x - 2 \rightarrow g(x) = \sqrt{x + 2}$$

y el valor inicial $x_0 = 0,5$, se obtiene el siguiente resultado:

```

Recuerde que f(x) = 0 se reescribe como x = g(x)
Ingresa la función original f(x): x^2 - x - 2
Ingresa la función iterativa g(x): sqrt(x + 2)
¿Desea aplicar el método de Punto Fijo? (s/n): s
Ingresa el valor inicial x0: 0.5

```

Iteración	x0	g(x0)	f(x0)	Error
1	0.500000	1.581139	-2.250000	1.08e+0
2	1.581139	1.892390	-1.081139	3.11e-1
3	1.892390	1.972914	-0.311251	8.05e-2
4	1.972914	1.993217	-0.080524	2.03e-2
5	1.993217	1.998304	-0.020303	5.09e-3
6	1.998304	1.999576	-0.005087	1.27e-3
7	1.999576	1.999894	-0.001272	3.18e-4
8	1.999894	1.999973	-0.000318	7.95e-5
9	1.999973	1.999993	-0.000080	1.99e-5
10	1.999993	1.999998	-0.000020	4.97e-6
11	1.999998	2.000000	-0.000005	1.24e-6
12	2.000000	2.000000	-0.000001	3.11e-7

```

Raíz aproximada encontrada: 2.000000
Iteraciones realizadas: 12

```

Figura 2: Tabla de iteraciones generada por el programa.

La raíz aproximada obtenida es:

$$x \approx 2$$

Método de la Secante

Descripción El método de la secante es una alternativa al método de Newton-Raphson que no requiere el cálculo de la derivada. En su lugar, aproxima la tangente por una línea secante que pasa por los dos últimos puntos de la iteración. Requiere dos puntos iniciales, x_0 y x_1 . Su fórmula de iteración es:

$$x_{n+1} = x_n - f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

Su velocidad de convergencia es superlineal, siendo más rápida que la bisección. Sin embargo, puede divergir si los puntos iniciales no son bien elegidos.

Código en JavaScript

```

1
2 const readline = require('readline');
3 const math = require('mathjs');
4
5 // Crear la interfaz para leer y escribir en la consola
6 const rl = readline.createInterface({
7   input: process.stdin,

```

```
8   output: process.stdout
9   });
10
11  function runSecantConsole() {
12    rl.question("Ingrese la funci n f(x): ", (funcStr) => {
13      if (!funcStr) {
14        console.log("No se ingres ninguna funci n.");
15        rl.close();
16        return;
17      }
18
19
20      const cleanFuncStr = funcStr.replace(/\*\*/g, '^').replace(/\s/g, '');
21
22      let f;
23      try {
24        const node = math.parse(cleanFuncStr);
25        const code = node.compile();
26        f = x => code.evaluate({ x: x });
27      } catch (error) {
28        console.error("Error en la funci n ingresada:", error.message);
29        rl.close();
30        return;
31      }
32
33      rl.question(" Desea encontrar una ra z con el m todo de la Secante? (s/n): ",
34        (op) => {
35          if (op.toLowerCase() !== 's') {
36            console.log("No se aplic el m todo de la Secante.");
37            rl.close();
38            return;
39          }
40
41          rl.question("Ingrese el primer valor inicial x0: ", (x0_input) => {
42            rl.question("Ingrese el segundo valor inicial x1: ", (x1_input) => {
43              let x0 = parseFloat(x0_input);
44              let x1 = parseFloat(x1_input);
45
46              if (isNaN(x0) || isNaN(x1)) {
47                console.error("Error: Los valores iniciales deben ser num ricos
48                  .");
49                rl.close();
50                return;
51              }
52
53              const tol = 1e-6;
54              const max_iter = 100;
55              let converged = false;
56
57              console.log("\nIteraci n | x0 | x1 | f(x0) | f(x1) | x2 | Error");
58              console.log("-----");
59
60              for (let i = 1; i <= max_iter; i++) {
61                const f0 = f(x0);
62                const f1 = f(x1);
63
64                if (Math.abs(f1 - f0) < 1e-15) { // Evitar divisi n por cero
65                  console.log(`\nDivisi n por cero en la iteraci n ${i}. El
66                    m todo se detiene.`);
67                  break;
68                }
69              }
70            }
71          }
72        }
73      )
74    }
75  }
76}
```

```

67         const x2 = x1 - f1 * (x1 - x0) / (f1 - f0);
68         const error = Math.abs(x2 - x1);
69
70         const iterText = i.toString().padStart(9);
71         const x0Text = x0.toFixed(6).padStart(11);
72         const x1Text = x1.toFixed(6).padStart(11);
73         const f0Text = f0.toFixed(6).padStart(15);
74         const f1Text = f1.toFixed(6).padStart(15);
75         const x2Text = x2.toFixed(6).padStart(11);
76         const errorText = error.toExponential(2).padStart(10);
77
78         console.log(`${iterText} | ${x0Text} | ${x1Text} | ${f0Text}
79         | ${f1Text} | ${x2Text} | ${errorText}`);
80
81         if (error < tol) {
82             console.log(`\nRa z aproximada encontrada: ${x2.toFixed(6)}
83             `);
84             console.log(`Iteraciones realizadas: ${i}`);
85             converged = true;
86             break;
87         }
88
89         x0 = x1;
90         x1 = x2;
91     }
92
93     if (!converged) {
94         console.warn(`\nNo se alcanz la convergencia despu s de ${
95         max_iter} iteraciones.`);
96     }
97
98     rl.close();
99 }
100
101 }
102
103 runSecantConsole();

```

Ejemplo de Ejecución

Para la función:

$$f(x) = x^3 - x - 1$$

y los valores iniciales $x_0 = 1$ y $x_1 = 2$, se obtiene el siguiente resultado:

La raíz aproximada obtenida es:

$$x \approx 1,324718$$

Método de Regula Falsi (Falsa Posición)

Descripción El método de Regula Falsi combina la robustez de la bisección con la velocidad de la secante. Requiere un intervalo inicial $[a, b]$ donde la función cambie de signo, lo que garantiza la convergencia. Calcula la siguiente aproximación trazando una línea recta entre los puntos $(a, f(a))$ y $(b, f(b))$, usando la fórmula:

$$c_n = b_n - f(b_n) \cdot \frac{b_n - a_n}{f(b_n) - f(a_n)}$$

Generalmente es más rápido que la bisección, pero puede volverse lento si uno de los extremos del intervalo se estanca durante las iteraciones.

H

```

Ingrese la función f(x): x**3 - x - 1
¿Desea encontrar una raíz con el método de la Secante? (s/n): s
Ingrese el primer valor inicial x0: -2
Ingrese el segundo valor inicial x1: 2

```

Iteración	x0	x1	f(x0)	f(x1)	x2	Error
1	-2.000000	2.000000	-7.000000	5.000000	0.333333	1.67e+0
2	2.000000	0.333333	5.000000	-1.296296	0.676471	3.43e-1
3	0.333333	0.676471	-1.296296	-1.366909	-5.965904	6.64e+0
4	0.676471	-5.965904	-1.366909	-207.372591	0.720545	6.69e+0
5	-5.965904	0.720545	-207.372591	-1.346449	0.764243	4.37e-2
6	0.720545	0.764243	-1.346449	-1.317874	2.779577	2.02e+0
7	0.764243	2.779577	-1.317874	17.695578	0.903931	1.88e+0
8	2.779577	0.903931	17.695578	-1.165337	1.019819	1.16e-1
9	0.903931	1.019819	-1.165337	-0.959175	1.558994	5.39e-1
10	1.019819	1.558994	-0.959175	1.230081	1.256047	3.03e-1
11	1.558994	1.256047	1.230081	-0.274439	1.311307	5.53e-2
12	1.256047	1.311307	-0.274439	-0.056478	1.325627	1.43e-2
13	1.311307	1.325627	-0.056478	0.003879	1.324707	9.20e-4
14	1.325627	1.324707	0.003879	-0.000049	1.324718	1.14e-5
15	1.324707	1.324718	-0.000049	-0.000000	1.324718	9.69e-9

```

Raíz aproximada encontrada: 1.324718
Iteraciones realizadas: 15

```

Figura 3:

Código en JavaScript

```

1
2 const readline = require('readline');
3 const math = require('mathjs');
4
5 const rl = readline.createInterface({
6   input: process.stdin,
7   output: process.stdout
8 });
9
10 function runRegulaFalsiConsole() {
11   console.log("--- M TODO DE REGULA FALSI (FALSA POSICI N) ---");
12
13   rl.question("Ingrese la funci n f(x): ", (funcStr) => {
14     if (!funcStr) {
15       console.log("No se ingres ninguna funci n.");
16       rl.close();
17       return;
18     }
19
20     const cleanFuncStr = funcStr.replace(/\*/g, '^').replace(/\s/g, '');
21
22     let f;
23     try {
24       const node = math.parse(cleanFuncStr);
25       const code = node.compile();
26       f = x => code.evaluate({ x: x });
27     } catch (error) {
28       console.error("Error en la sintaxis de la funci n:", error.message);
29       rl.close();
30       return;
31     }
32
33     rl.question(" Desea aplicar el m todo de Regula Falsi? (s/n): ", (op) => {

```

```
34     if (op.toLowerCase() !== 's') {
35         console.log("No se aplicó el método de Regula Falsi.");
36         rl.close();
37         return;
38     }
39
40     rl.question("Ingrese el valor de a (extremo izquierdo): ", (a_input) => {
41         rl.question("Ingrese el valor de b (extremo derecho): ", (b_input) => {
42             let a = parseFloat(a_input);
43             let b = parseFloat(b_input);
44
45             if (isNaN(a) || isNaN(b)) {
46                 console.error("Error: Los valores del intervalo deben ser
47                     numéricos.");
48                 rl.close();
49                 return;
50             }
51
52             // Comprobación del cambio de signo, requisito fundamental del
53             // método
54             if (f(a) * f(b) > 0) {
55                 console.warn("\nLa función no cambia de signo en el intervalo.
56                     No se puede aplicar el método.");
57                 rl.close();
58                 return;
59             }
60
61             const tol = 1e-6;
62             const max_iter = 100;
63             let converged = false;
64             let c_old = a; // Para calcular el error
65
66             console.log("\nIteración | a | b | c | f(c) | Error");
67             console.log("-----");
68
69             for (let i = 1; i <= max_iter; i++) {
70                 const fa = f(a);
71                 const fb = f(b);
72
73                 if (Math.abs(fa - fb) < 1e-15) {
74                     console.log(`\nDivisión por cero inminente en la iteración
75                         ${i}. El método se detiene.`);
76                     break;
77                 }
78
79                 // Fórmula de Regula Falsi
80                 const c = b - (fb * (b - a)) / (fb - fa);
81                 const fc = f(c);
82                 const error = Math.abs(c - c_old);
83                 c_old = c;
84
85                 const iterText = i.toString().padStart(9);
86                 const aText = a.toFixed(6).padStart(11);
87                 const bText = b.toFixed(6).padStart(11);
88                 const cText = c.toFixed(6).padStart(11);
89                 const fcText = fc.toFixed(6).padStart(13);
90                 const errorText = error.toExponential(2).padStart(10);
91
92                 console.log(`${iterText} | ${aText} | ${bText} | ${cText} | ${fcText} | ${errorText}`);
93
94                 if (Math.abs(fc) < tol || error < tol) {
```

```

93         console.log(`\nRaíz aproximada encontrada: ${c.toFixed(6)
94             }`);
95         console.log(`Iteraciones realizadas: ${i}`);
96         converged = true;
97         break;
98     }
99     // Actualización de los intervalos, manteniendo el cambio de
100     signo
101     if (fa * fc < 0) {
102         b = c;
103     } else {
104         a = c;
105     }
106 }
107 if (!converged) {
108     console.warn(`\nNo se alcanzó la convergencia después de ${
109         max_iter} iteraciones.`);
110 }
111 rl.close();
112 });
113 });
114 });
115 });
116 }
117
118 runRegulaFalsiConsole();

```

Ejemplo de Ejecución

Para la función:

$$f(x) = x^3 - 2x - 1$$

y el intervalo inicial $[2, 3]$, se obtiene el siguiente resultado:

Ingrese la función f(x): $x^3 - 2x - 5$
 ¿Desea aplicar el método de Regula Falsi? (s/n): s
 Ingrese el valor de a (extremo izquierdo): 2
 Ingrese el valor de b (extremo derecho): 3

Iteración	a	b	c	f(c)	Error
1	2.000000	3.000000	2.058824	-0.390800	5.88e-2
2	2.058824	3.000000	2.081264	-0.147204	2.24e-2
3	2.081264	3.000000	2.089639	-0.054677	8.38e-3
4	2.089639	3.000000	2.092740	-0.020203	3.10e-3
5	2.092740	3.000000	2.093884	-0.007451	1.14e-3
6	2.093884	3.000000	2.094305	-0.002746	4.22e-4
7	2.094305	3.000000	2.094461	-0.001012	1.55e-4
8	2.094461	3.000000	2.094518	-0.000373	5.72e-5
9	2.094518	3.000000	2.094539	-0.000137	2.11e-5
10	2.094539	3.000000	2.094547	-0.000051	7.77e-6
11	2.094547	3.000000	2.094550	-0.000019	2.86e-6
12	2.094550	3.000000	2.094551	-0.000007	1.05e-6
13	2.094551	3.000000	2.094551	-0.000003	3.88e-7

Raíz aproximada encontrada: 2.094551
 Iteraciones realizadas: 13

Figura 4:

La raíz aproximada obtenida es:

$$x \approx 2,094551$$