Algorithm 1 INCREMENTAL SEARCH

```
\overline{\textbf{Input: } f, x_0, h, n_{max}}
Output: a, b, iter
    x_{ant} \leftarrow x_o
    f_{ant} \leftarrow f(x_{ant})
    x_{act} \leftarrow x_{ant} + h
    f_{act} \leftarrow f(a_{ct})
    For i \leftarrow 1, n_{max} \ \mathbf{do}
        if f_{ant} * f_{act} < 0 Then
            break
        End if
        x_{ant} \leftarrow x_{act}
       f_{ant} \leftarrow f_{act}
x_{act} \leftarrow x_{ant} + h
f_{act} \leftarrow f(x_{act})
    End For
    a \leftarrow x_{ant}
    b \leftarrow x_{act}
    iter \leftarrow i
    {\bf Return}\ a,b,iter
```

Algorithm 2 BISECTION

```
Input: f, a, b, tol, n_{max}
Output: x, iter, err
 1: fa \leftarrow f(a)
 2: f_{pm} \leftarrow (a+b)/2
 3: f_{pm} \leftarrow f(pm)
 4: E \leftarrow 1000
 5: cont \leftarrow 1
 6: While (E > tol) & (cont < N_{max}) do
 7:
       if (f_a * f_{pm} < 0) Then
          b \leftarrow pm
 8:
       else if
 9:
10:
          a \leftarrow pm
11:
       End if
12:
       p_0 \leftarrow pm
       pm \leftarrow (a+b)/2
13:
14:
       f_{pm} \leftarrow f(pm)
15:
       E \leftarrow |pm - p_0|
16:
       cont \leftarrow cont + 1
17: End While
18: x \leftarrow pm
19: iter \leftarrow cont
20: err \leftarrow E
21: Return x, iter, err
```

Algorithm 3 FALSE RULE

```
Input: f, a, b, tol, N_{max}
Output: x, iter, err
 1: f_a \leftarrow f(a)
 2: f_b \leftarrow f(b)
3: f_{pm} \leftarrow f(pm)
4: E \leftarrow 1000
 5: cont \leftarrow 1
 6: While (E > tol) & (cont < N_{max}) do
        if (f_a * f_{pm} < 0) Then
           b \leftarrow pm
 8:
        else if
 9:
10:
          a \leftarrow pm
        End if
11:
12:
        p_0 \leftarrow pm
        pm \leftarrow (f(b) * a - f(a) * b) / f(b) - f(a))
13:
        f_{pm} \leftarrow f(pm)
14:
        \vec{E} \leftarrow |pm - p_0|
15:
        cont \leftarrow cont + 1
16:
17: End While
18: x \leftarrow pm
19: iter \leftarrow cont
20: err \leftarrow E
21: Return x, iter, err
```

Algorithm 4 FIXED POINT

```
Input: g, x_0, tol, N_{max}
Output: x, iter, err
 1: x_{ant} \leftarrow x0
 2: E \leftarrow 1000
 3: cont \leftarrow 1
 4: While (E > tol) & (cont < N_{max}) do
        x_{act} \leftarrow g(x_{ant})
 5:
        E \leftarrow |x_{act} - x_{ant}|
 6:
        cont \leftarrow cont + 1
 7:
        x_{ant} \leftarrow x_{act}
 9: End While
10: x \leftarrow x_{act}
11: iter \leftarrow cont
12: err \leftarrow E
13: Return x, iter, err
```

Algorithm 5 NEWTON

```
\overline{\textbf{Input:} f, df, x_0, tol, N_{max}}
Output: x, iter, err
 1: x_{ant} \leftarrow x0
 2: f_{ant} \leftarrow f(x_{ant})
 3: E \leftarrow 1000
 4: cont \leftarrow 0
 5: While (E > tol) & (cont < N_{max}) do
         x_{act} \leftarrow (x_{ant} - f_{ant})/(df(x_{ant}))
         f_{act} \leftarrow f(x_{act})
 7:
         E \leftarrow |x_{act} - x_{ant}|
 8:
         cont \leftarrow cont + 1
10:
         x_{ant} \leftarrow x_{act}
         f_{ant} \leftarrow f_{act}
11:
12: End While
13: x \leftarrow x_{act}
14: iter \leftarrow cont
15: err \leftarrow E
16: Return x, iter, err
```

Algorithm 6 SEC

```
\overline{\textbf{Input: } f, x_0, x_1, tol, N_{max}}
Output: x, iter, err
 1: f_0 \leftarrow f(x_0)
 2: f_1 \leftarrow f(x_1)
 3: E \leftarrow 1000
 4: cont \leftarrow 1
 5: While (E > tol) & (cont < N_{max}) do
         x_{act} \leftarrow x1 - f1 * (x_1 - x_0)/(f_1 - f_0)
 7:
         f_{act} \leftarrow f(x_{act})
         E \leftarrow |x_{act} - x_1|
 8:
 9:
         cont \leftarrow cont + 1
10:
         x_0 \leftarrow x_1
         f_0 \leftarrow f_1
11:
         x_1 \leftarrow x_{act}
12:
         f_1 \leftarrow f_{act}
14: End While
15: x \leftarrow x_{act}
16: iter \leftarrow cont
17: err \leftarrow E
18: Return x, x_{act}, err
```

Algorithm 7 MULTIPLE ROOTS

```
Input: f, f', f'', x0, tol, N_{max}
Output: x, iter, err
 1: x_{ant} \leftarrow x0
 2: f_{ant} \leftarrow f(act)
 3: E \leftarrow 1000
 4: cont \leftarrow 0
 5: While (E > tol) & (cont < N_{max}) do
        x_{act} \leftarrow (x_{ant} - f_{ant}) * f''(x_{ant}) / (f'(x_{ant})^2 - f_{ant} * f''(x_{ant}))
        fa_{ct} \leftarrow f(x_{act})
        E \leftarrow |x_{act} - x_{ant}|
 8:
        cont \leftarrow cont + 1
 9:
10:
        x_{ant} \leftarrow x_{act}
        f_{ant} \leftarrow f_{act}
11:
12: End While
13: x \leftarrow x_{act}
14: iter \leftarrow cont
15: err \leftarrow E
16: Return x, iter, err
```

Algorithm 8 GAUSSIAN ELIMINATION

```
Input: A, b

Output: x

1: n \leftarrow size(A, 1)

2: M \leftarrow [Ab]

3: For i \leftarrow 0, n - 1 do

4: For j \leftarrow i + 1, n do

5: if M(j, i) \neq 0 Then

6: M(j, i : n + 1) \leftarrow M(j, i : n + 1) - (M(j, i)/M(i, i)) * M(i, i : n + 1)

7: End if

8: End For

9: End For

10: x \leftarrow backsubstitution(M)

11: Return x
```

Algorithm 9 GAUSSIAN ELIMINATION WITH PARCIAL PIVOTING

```
Input: A, b
Output: x
  n \leftarrow size(A, 1)
  M \leftarrow [Ab]
  For i \leftarrow 1, n-1 do
     [aux0, aux] \leftarrow max|M(i+1:n,1)|
     if aux_0 > |M(i,i)| Then
       aux_2 \leftarrow M(i + aux, i : n + 1)
       M(aux+i,i:n+1) \leftarrow M(i,i:n+1)
        M(i, i: n+1) \leftarrow aux_2
     End if
     For j \leftarrow i+1, n do
       if M(j,i) \neq 0 Then
          M(j, i: n+1) \leftarrow M(j, i: n+1) - (M(j, i)/M(i, i)) * M(i, i: n+1)
       End if
     End For
  End For
  x \leftarrow backsubstitution(M)
  \textbf{Return} \ \ x
```

Algorithm 10 GAUSSIAN ELIMINATION WITH TOTAL PIVOTING

```
Input: A, b
Output: x
  n \leftarrow size(A, 1)
  M \leftarrow [Ab]
  aux \leftarrow []
  For i \leftarrow 1, n-1 do
     [a,b] \leftarrow find(|M(i:n,i:n)|) \equiv max(|M(i:n,i:n)|) Cambio de columna
     if b(1) + i - 1 \neq i Then
       aux \leftarrow + = [ib(1) + i + 1]
        aux_2 \leftarrow M(:, b(1) + i + 1)
        M(:,b(1)+i-1) \leftarrow
        M(:,i) \leftarrow
     End if Cambio de fila
     if a(1) + i \neq i Then
        aux_2 \leftarrow M(i + a(1) - 1, i : n + 1)
        M(a(1) + i - 1, i : n + 1) \leftarrow M(i, i : n + 1)
        M(i, i: n+1) \leftarrow aux_2
     End if
     For j \leftarrow i+1, n do
       if M(j,i) \neq 0 Then
          M(j, i: n+1) \leftarrow M(j, i: n+1) - (M(j, i)/M(i, i)) * M(i, i: n+1)
        End if
     End For
  End For
  x \leftarrow backsubstitution(M)
  Return x
```