Root-Finding Algorithms

1. Bisection Method

Input: f(x), interval [a, b], tolerance e

Condition: f(a) * f(b) < 0

Steps:

1. Check if f(a) * f(b) < 0. If not, root not guaranteed.
2. Repeat until |b - a| < e:
   - c = (a + b) / 2
   - If f(c) == 0, return c
   - Else if f(a) * f(c) < 0, b = c
   - Else, a = c
3. Return c as the root.

2. False Position (Regula Falsi) Method

Input: f(x), interval [xl, xu], tolerance e

Condition: f(xl) * f(xu) < 0

Steps:

1. Check if f(xl) * f(xu) < 0. If not, root not guaranteed.
2. Repeat until |f(xr)| < e:
   - xr = xu - (f(xu) * (xl - xu)) / (f(xl) - f(xu))
   - If f(xr) == 0, return xr
   - Else if f(xl) * f(xr) < 0, xu = xr
   - Else, xl = xr
3. Return xr as the root.

3. Newton-Raphson Method

Input: f(x), f'(x), initial guess x0, tolerance e

Steps:

1. Set x = x0
2. Repeat until |f(x)| < e:

- x = x - f(x) / f'(x)
- If f'(x) == 0, stop (division by zero)

3. Return x as the root.

4. Secant Method

Input: f(x), initial guesses x0, x1, tolerance e

Steps:

1. Set x0, x1
2. Repeat until |f(x2)| < e:
   - x2 = x1 - f(x1) * (x1 - x0) / (f(x1) - f(x0))
   - Update x0 = x1, x1 = x2
3. Return x2 as the root.