Algorithm 1: Bisection method

Data: a, b, N, δ, ϵ
Result: Mid point of bracketing interval
$a_0 = a$, $b_0 = b$, $k = 0$;
while $k < N$ **do**
    $c_k = \frac{1}{2}(a_k + b_k)$;
    if $|f(c_k)| < \delta$ **then**
        Return c_k;
    end
    if $signf(c_k) \neq signf(b_k)$ **then**
        $a_{k+1} = c_k$;
        $b_{k+1} = b_k$;
    else
        $a_{k+1} = a_k$;
        $b_{k+1} = c_k$;
    end
    $k = k + 1$;
    if $b_k - a_k < \epsilon$ **then**
        Return $\frac{1}{2}(a_k + b_k)$;
    end
end

Algorithm 2: Bisection method

Data: a, b, N, δ, ϵ
Result: Mid point of bracketing interval
$k = 0$;
while $k < N$ **do**
  $c = \frac{1}{2}(a + b)$;
  if $|f(c)| < \delta$ **then**
    Return c;
  end
  if $signf(c) \neq signf(b)$ **then**
    $a = c$;
  else
    $b = c$;
  end
  $k = k + 1$;
  if $b - a < \epsilon$ **then**
    Return $\frac{1}{2}(a + b)$;
  end
end

Algorithm 3: Newton-Raphson method

Data: x, N, δ, ϵ
Result: Root
$k = 0$, $f = f(x)$;
while $k < N$ **do**
    $df = f'(x)$;
    $e = -f/df$;
    $x = x + e$;
    $f = f(x)$;
    if $|e| < |x|\epsilon$ *or* $|f| < \delta$ **then**
        Return x;
    end
    $k = k + 1$;
end