A Fast Algorithm for Solving Nonlinear Equations

Xiaodong Wang¹, Lei Wang ²
¹ Quanzhou Normal University, Quanzhou 362000, China
² Georgia Institute of Technology, Atlanta GA 30332, USA

Abstract. In this work, we develop a simple yet practical algorithm for finding a root of a real function \( f'(x) \) with a good local convergence. The algorithm uses a continued fraction interpolation method that can be easily implemented in software packages for achieving desired convergence orders. For the general \( n \)-point formula, the order of convergence rate of the presented algorithm is \( \tau_n \), the unique positive root of the equation \( x^n - x^{n-1} - \cdots - x - 1 = 0 \). Computational results ascertain that the developed algorithm is efficient and demonstrate equal or better performance as compared with other well known methods.

Keywords: Iterative methods, higher order, convergence rate, algorithm

1. Introduction

One of the most frequently occurring problems in scientific work is to locate a real root \( \alpha \) of a nonlinear equation
\[
f'(x) = 0
\] (1)

Various problems arising in diverse disciplines of engineering, sciences and nature can be described by a nonlinear equation of the form (1). For example, many optimization problems and boundary value problems appearing in many applied areas are reduced to solving the preceding equation. Therefore solving the equation (1) is a very important task and there exists numerous methods for solving the above equation [9, 13, 6, 11, 12]. Though there exists many iterative methods but still the Newton's method, one of the best known and the probably the oldest, is extensively used for solving the nonlinear equation (1) [3, 5, 10]. Many methods have been developed which improve the convergence rate of the Newton's method [4, 10, 7, 1, 2, 8].

In this paper, we present an iterative algorithm for the problem based on the continued fraction interpolation method. For the general iteration scheme of \( n \) points, the algorithm constructs the \( n+1 \)th point according to the information given by the previous \( n \) points. Thus the algorithm has a good local convergence. Furthermore, the scheme of the algorithm is very simple and easy to implement on a computer. The numerics experiments indicate that the scheme of the algorithm even for three points converges very fast.

2. The Design of the Algorithm

Let \( f \in C^1([a,b]), y(x) = f'(x) \). We want to find \( x^* \in (a,b) \) such that \( y(x^*) = f'(x^*) = 0 \). Suppose we are given \( n \) approximated values \( x_0, x_1, \cdots, x_{n-1} \) of \( x^* \) and the function \( f' \) with corresponding derivatives \( y_i = y(x_i), 0 \leq i < n \).

In order to find the real root of \( y \), we construct the following scheme of \( n \) points based on the continued fraction interpolation of \( y = f'(x) \) as follows.
The $k$th order difference of function $f$ can be defined recursively as follows.

$$\Phi_{k}(y)=\chi$$

$$\Phi_{k}(y_{1}, \cdots, y_{k}, y)=\frac{y-y_{i}}{\Phi_{k-1}(y_{1}, \cdots, y_{k-1}, y)-\Phi_{k-1}(y_{i}, \cdots, y_{i})}, k=1,2, \cdots$$

It follows from the inverse interpolation condition $\psi(y_{i})=x_{i}, 0 \leq i < n$ that

$$\begin{align*}
a_{0} &= x_{n-1} = \Phi_{0}(y_{n-1}) \\
a_{i} &= \Phi_{i}(y_{n-1}, \cdots, y_{n-i}), i = 1, 2, \cdots, n-1
\end{align*}$$

It follows that

$$\psi(0) = a_{0} = \frac{y_{n-1}}{a_{1}} = \frac{y_{n-2}}{y_{2}} \cdots \frac{y_{1}}{a_{n-1}}$$

According to the formula described above, we can design a very simple iteration algorithm to find $x^{*}$ as follows.

**Algorithm 2.1: Inverse-Interpolation($f$)**

**Input** $x_{0}, x_{1}, \cdots, x_{n-1}, \epsilon$

**for** $i \leftarrow 0$ to $n-1$

**do** $y_{i} \leftarrow f'(x_{i})$

**while** $y_{n-1} > \epsilon$

$$\begin{align*}
g(y_{n-1}, \cdots, y_{1}) \\
x_{n} \leftarrow a_{0} = \frac{y_{n-1}}{a_{1}} = \frac{y_{n-2}}{y_{2}} \cdots \frac{y_{1}}{a_{n-1}} \\
d \leftarrow \\
y_{n} \leftarrow f'(x_{n}) \\
n \leftarrow n + 1
\end{align*}$$

**Output** $(x_{n}, y_{n}, f(x_{n}))$

where the function $g(y_{n-1}, \cdots, y_{1})$ is used to compute the coefficients $a_{0}, \cdots, a_{n-1}$ according to the formula (4).

### 3. The Convergence of the Algorithm
Suppose \( f \in C^{(i)}[a,b], x \in [a,b] \). If for \( x_i \in [a,b], 0 \leq i \leq k \), the limit \( \lim_{x \to x_i} f'(x) = \Phi_k(x) \) exists, then the function \( f \) is \( k \) th inverse differentiable and \( \Phi_k(x) \) is called the \( k \) th inverse derivative of \( f \) at \( x \).

Let \( x^* \in [a,b] \) and \( f'(x^*) = 0 \). The function type \( T^{(n)}[a,b] \subseteq C^{(i)}[a,b] \) can be defined as

\[
T^{(n)}[a,b] = \{ f \mid f \text{ kth inverse differentiable in } [a,b], \Phi_k(x^*) \neq 0, 1 \leq k \leq n \}
\]

Suppose for any function \( f \in T^{(n)}[a,b] \), the point sequence generated by algorithm 2.1 be \( \{x_k\}_{k=1}^{\infty} \). From the computing steps of the algorithm we know \( x_{k+1} \) can be formulated as

\[
x_{k+1} = a_0 - \frac{y_k}{a_1} - \frac{y_{k-1}}{a_2} - \cdots - \frac{y_{k-n+2}}{a_{n-1}} - \frac{y_{k-n+1}}{a_n}
\]

where \( a_0 = x_k, \Phi_k(y_i) \), and \( a_i = \Phi_k(y_k, \ldots, y_{k-i}), i = 1, 2, \ldots, n-1 \).

Now, we can compute \( \psi_i(y) \), the \( n+1 \) points inverse continued fraction interpolation of \( y = f'(x) \) at \( x^*, x_{k-n+1}, \ldots, x_k \) as follows

\[
\psi_i(y) = a_0' + \frac{y - y_k}{a_1'} + \frac{y - y_{k-1}}{a_2'} + \cdots + \frac{y - y_{k-n+2}}{a_{n-1}'} + \frac{y - y_{k-n+1}}{a_n'}
\]

It follows from the inverse interpolation condition \( \psi_i(y_i) = x_i, k-n+1 \leq i \leq k \), and \( \psi_i(0) = x^* \) that \( a_i' = a_i, 0 \leq i < n \), and \( a_n' = \Phi_n(y_k, \ldots, y_{k-n+1}, 0) \).

It follows that

\[
\psi_i(0) = a_0 - \frac{y_k}{a_1} - \frac{y_{k-1}}{a_2} - \cdots - \frac{y_{k-n+2}}{a_{n-1}} - \frac{y_{k-n+1}}{a_n}
\]

Denote \( x^* = \frac{P_i}{Q_i} \), and \( x_{k+1} = \frac{P_{n-1}}{Q_{n-1}} \). It is not difficult to see that

\[
|x_{k+1} - x^*| = \left| \frac{P_{n-1}}{Q_{n-1}} - \frac{P_n}{Q_n} \right| = \left| \frac{\prod_{i=1}^{n} y_{k-n+i}}{Q_{n-1} Q_n} \right|
\]

It follows that

\[
|x_{k+1} - x^*| = \left| \prod_{i=1}^{n} x_{k-n+i} - x^* \right| \left| \prod_{i=1}^{n} \Phi_i(y_{k-n+i}, 0) \right| \frac{Q_{n-1} Q_n}{P_{n-1} P_n}
\]
Theorem 1 Let \( f \in T^n[a,b] \). There must be \( \varepsilon > 0 \) and \( \delta > 0 \) such that when \( \{x_{n-k+1}, \ldots, x_k\} \subset o(x^*, \delta) \), \( |Q_{n-1}Q_n| \geq \varepsilon \).

Theorem 2 Let \( f \in T^n[a,b] \). There must be \( \delta > 0 \) such that when \( \{x_0, \ldots, x_{n-1}\} \subset o(x^*, \delta) \), the sequence \( \{x_n\} \) generated by the algorithm 2.1 converges to \( x^* \).

4. References


