Technique of Initial Solution Finding
for the Newton Method

Sa-at Muangchan
Department of Mathematics and Statistics
Faculty of Science and Technology
Sakon Nakhon Rajabhat University
Sakon Nakhon 47000, Thailand
msaat@hotmail.com

Supunnee Sompong
Department of Mathematics and Statistics
Faculty of Science and Technology
Sakon Nakhon Rajabhat University
Sakon Nakhon 47000, Thailand
s_sanpinij@yahoo.com

Abstract
The aim of this research is to proposed the computational algorithms of initial solution finding for the Newton method by using the technique of linear and parabolic regressions. The numerical results of example non-linear equations were presented and performed on Matlab program.

Keywords: non-linear equations, regression, Newton method

1 Introduction

Many problems of mathematics involve, at some point or another, solving an equation for an unknown quantity. Let’s us consider a continuously non-linear equation

\[ f(x) = 0. \]  \hspace{1cm} (1)

Let’s suppose that we would like to approximate the solution to the equation (1) and let’s also suppose that we have somehow found an initial approximation to this solution say, \( x_0 \). This initial approximation is probably not all that good and so we would like to find a better approximation.
By the Newton method ([2], [4]), we can approximate the solution of the equation (1) to within a tolerance error of beginning with an initial guess \(x_0\), we compute the sequence of approximations \(x_1, x_2, x_3, \ldots\) by using the below formula

\[
x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}
\]

where \(i = 0, 1, 2, \ldots\)

However, Newton’s Method fails to produce a solution, if there is no solution to be found or the initial solution is not good enough for the method. In the practice, the initial solution \(x_0\) is really important for the Newton method. Some initial solutions will lead to the exact numerical solution. But some initial solutions can make the method diverges.

Hence, in this paper, we concentrate and propose the initial solution finding for the Newton method by using the technique of linear and parabolic regressions.

## 2 Main Results

Let’s us consider the nonlinear equation \(f(x) = 0\) on the interval \([A, B]\). Then we divide this interval into \(n\) subintervals with the step-length \(h = (B - A)/n\) and we have the points \(x_0, x_1, x_2, \ldots, x_n\) such as below:

\[
x_k = x_{k-1} + h = x_0 + kh \quad \text{for} \quad k = 1, 2, \ldots, n - 1
\]

where \(x_0 = A\) and \(x_n = B\).

Hence, we can calculate the value of \(y_k\) for \(k = 0, 1, 2, \ldots, n\) as the following formula:

\[
y_k = f(x_k) \quad \text{for} \quad k = 0, 1, 2, \ldots, n.
\]

To find the appropriately initial solution, we construct the following algorithms by the linear and parabolic regressions.

**Algorithm1:** To find the appropriately initial solution for the Newton method, we can apply the parabolic regression to the points \((x_k, y_k)\) for \(k = 0, 1, 2, \ldots, n\) and then we can find the least-square parabola that fits all the points with the following equation:

\[
y = ax^2 + bx + c
\]

To find the coefficients \(a, b\) and \(c\) in equation (5), we can state as the following:
For $k = 0, 1, 2, ..., n$ we have
\[
\begin{align*}
ax_k^2 + bx_k + c &= y_k \\
ax_k^3 + bx_k^2 + cx_k &= x_ky_k \\
ax_k^4 + bx_k^3 + cx_k^2 &= x_k^2y_k.
\end{align*}
\]  
\tag{6}

Then, we can summarize the above system as the follows:
\[
\begin{align*}
a \sum_{k=0}^{n} x_k^2 + b \sum_{k=0}^{n} x_k + n &= \sum_{k=0}^{n} y_k \\
a \sum_{k=0}^{n} x_k^3 + b \sum_{k=0}^{n} x_k^2 + \sum_{k=0}^{n} x_k &= \sum_{k=0}^{n} x_ky_k \\
a \sum_{k=0}^{n} x_k^4 + b \sum_{k=0}^{n} x_k^3 + \sum_{k=0}^{n} x_k^2 &= \sum_{k=0}^{n} x_k^2y_k.
\end{align*}
\]  
\tag{7}

For convenience, we can rewrite the system (7) as below:
\[
\begin{bmatrix}
1 & \beta & \alpha \\
\xi & 1 & \beta \\
\delta & \xi & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
\varepsilon_1 \\
\varepsilon_2 \\
\varepsilon_3
\end{bmatrix}
\]  
\tag{8}

where
\[
\beta = \frac{\sum_{k=0}^{n} x_k}{\sum_{k=0}^{n} x_k^2}, \quad \alpha = \frac{n}{\sum_{k=0}^{n} x_k^2}, \quad \xi = \frac{\sum_{k=0}^{n} x_k^3}{\sum_{k=0}^{n} x_k^2}, \quad \delta = \frac{n}{\sum_{k=0}^{n} x_k^2}, \quad \varepsilon_1 = \frac{\sum_{k=0}^{n} y_k}{\sum_{k=0}^{n} x_k^2},
\]
\[
\varepsilon_2 = \frac{\sum_{k=0}^{n} x_ky_k}{\sum_{k=0}^{n} x_k^2} \quad \text{and} \quad \varepsilon_3 = \frac{\sum_{k=0}^{n} x_k^2y_k}{\sum_{k=0}^{n} x_k^2}.
\]

To solve the solution of the above system, we use the row eliminations for the first and second rows. Hence, we have
\[
\begin{bmatrix}
1 - \alpha\delta & \beta - \alpha\xi & 0 \\
\xi - \beta\delta & 1 - \beta\xi & 0 \\
\delta & \xi & 1
\end{bmatrix}
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
= 
\begin{bmatrix}
\varepsilon_1 - \alpha \varepsilon_3 \\
\varepsilon_2 - \beta \varepsilon_3 \\
\varepsilon_3
\end{bmatrix}.
\]  
\tag{9}

From (9), we can consider the following system of two equations
\[
(1 - \alpha\delta) a + (\beta - \alpha\xi) b = \varepsilon_1 - \alpha \varepsilon_3
\]
\[(\xi - \beta \delta) a + (1 - \beta \xi) b = \varepsilon_2 - \beta \varepsilon_3.\]

Thus, we have
\[b = \frac{(\xi - \beta \delta) (\varepsilon_1 - \alpha \varepsilon_3) - (1 - \alpha \delta) (\varepsilon_2 - \beta \varepsilon_3)}{(\xi - \beta \delta) (\beta - \alpha \xi) - (1 - \alpha \delta) (1 - \beta \xi)}\]

\[(10)\]

\[a = \frac{(\varepsilon_1 - \alpha \varepsilon_3) - (\beta - \alpha \xi) b}{1 - \alpha \delta}\]

\[(11)\]

where \(\alpha \delta \neq 1\) and \(\beta \xi \neq 1\). From the third row of system (9), we have
\[c = \varepsilon_3 - \delta a - \xi b\]

\[(12)\]

Then, we consider the x-intercepts of the regression of \(f(x)\). That are
\[x_0^* = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\]

\[(13)\]

which we will use as the initial solution of the Newton method.

Thus, we can state the algorithm for finding the initial solution and the solution of the equation (1) by the Newton method as follows:

**Computational Algorithm 1 (ALG1):**

Step 1: Choose \(n\) large enough and compute \(x_k = x_{k-1} + h\) for \(k = 1, 2, \ldots, n - 1\) where \(h = (B - A)/n\), \(x_0 = A\) and \(x_n = B\).

Step 2: Compute \(y_k = f(x_k)\) for \(k = 0, 1, 2, \ldots, n\).

Step 3: Compute the values \(b, a\) and \(c\) from (10), (11) and (12), respectively.

Step 4: Compute the initial values of \(x_0^*\) from the equation (13).

Step 5: If \(x_0^* \notin [A, B]\), STOP: there are no solution on \([A, B]\).

Step 6: If \(x_0^* \in [A, B]\) then GOTO step 6.1.

Step 6.1: Set \(i = 0\) and tolerance of error\((TOL \approx 10^{-8})\).

Step 6.2: If \(|f(x_i^*)| < TOL\), STOP: the solution is \(x^* = x_i^*\).

Step 6.3: Compute
\[x_{i+1}^* = x_i^* - \frac{f(x_i^*)}{f'(x_i^*)}\]

and set \(i = i + 1\). GOTO Step 6.2.
Algorithm 2: This algorithm, we applied a linear regression to the data 
\((x_k, y_k)\) for \(k = 0, 1, 2, \ldots, n\). With this method, we compute the coefficients
\(m\) (slope) and \(c\) (y-intercept) of the straight line equation
\[ y = mx + c \] (14)
such that the sum of the squares of the errors will be minimum. We derive
the values of \(m\) and \(c\) that will make the equation of the straight line to the
best fit the data \((x_k, y_k)\) for \(k = 0, 1, 2, \ldots, n\). So, we have
\[ y_k = mx_k + c \]
for \(k = 0, 1, 2, \ldots, n\). Similarly as the above algorithm, we have the following
system:
\[
\begin{align*}
\left(\sum_{k=0}^{n} x_k^2\right) m + \left(\sum_{k=0}^{n} x_k\right) c &= \sum_{k=0}^{n} x_k y_k \\
\left(\sum_{k=0}^{n} x_k\right) m + (n + 1) c &= \sum_{k=0}^{n} y_k.
\end{align*}
\] (15)

Hence, we can solve the above system (5) by Cramer’s rule as follows:
\[
m = \frac{\left| \begin{array}{cc} \sum_{k=0}^{n} x_k y_k & \sum_{k=0}^{n} x_k \\
\sum_{k=0}^{n} y_k & n + 1 \end{array} \right|}{\left| \begin{array}{cc} \sum_{k=0}^{n} x_k^2 & \sum_{k=0}^{n} x_k y_k \\
\sum_{k=0}^{n} x_k & n + 1 \end{array} \right|} \quad \text{and} \quad c = \frac{\left| \begin{array}{cc} \sum_{k=0}^{n} x_k^2 & \sum_{k=0}^{n} x_k \\
\sum_{k=0}^{n} x_k & n + 1 \end{array} \right|}{\left| \begin{array}{cc} \sum_{k=0}^{n} x_k^2 & \sum_{k=0}^{n} x_k y_k \\
\sum_{k=0}^{n} x_k & n + 1 \end{array} \right|}.
\] (16)

Thus, we have the straight line \(y = mx + c\) which approximates the function
\(f(x)\) on the interval \([A, B]\). If we set \(y = 0\), then the x-intercept is \(x_0^* = -\frac{c}{m}\)
where \(m \neq 0\). Then, we can construct the algorithm for find the initial
solutions of non-linear equation on the interval \([A, B]\) as follows:

**Computational Algorithm 2 (ALG2):**

Step 1: Choose \(n\) large enough and compute \(x_k = x_{k-1} + h\) for
\(k = 1, 2, \ldots, n - 1\) where \(h = (B - A)/n\), \(x_0 = A\) and \(x_n = B\).
Step 2: Compute \(y_k = f(x_k)\) for \(k = 0, 1, \ldots, n\).
Step 3: Compute the values $m$ and $c$ from (16).

Step 4: Compute the initial values of $x_0^* = -\frac{c}{m}$ where $m \neq 0$.

Step 5: If $x_0^* \notin [A, B]$, STOP: there are no solution on $[A, B]$.

Step 6: If $x_0^* \in [A, B]$ then GOTO step 6.1.

Step 6.1: Set $i = 0$ and tolerance of error ($TOL \approx 10^{-8}$).

Step 6.2: If $|f(x_i^*)| < TOL$, STOP: the solution is $x^* = x_i^*$.

Step 6.3: Compute

$$x_{i+1}^* = x_i^* - \frac{f(x_i^*)}{f'(x_i^*)}$$

and set $i = i + 1$. GOTO Step 6.2.

3 Numerical Results

In this section, we tested the algorithms with the specific examples ([1], [5]) on Matlab program. For the accuracy, we use tolerance error ($TOL$) less than $0.5 \times 10^{-8}$. The tested nonlinear equations are below:

\[
\begin{align*}
  f_1(x) &= e^{-x} + \cos(x), & f_4(x) &= x^2 - e^x - 3x + 2, \\
  f_2(x) &= 10xe^{-x^2} - 1, & f_5(x) &= \sin(x) - \frac{x}{2}, \\
  f_3(x) &= x^3 - 2x + 2, & f_6(x) &= \sin(x) - x^5 + x^3 - 1.
\end{align*}
\]

The results of the algorithms are shown in the following tables.

4 Conclusion and Remarks

From the numerical results, both of our algorithms can find the appropriately initial solution which reduced the iterations number of Newton method. In practice, the number of subinterval $n$ is also important. If the value of $n$ is too small then the algorithms may fail to find an appropriately initial solution and in the other hand, if the number of $n$ is too large then the computer will spend more time for fit the data. In the future work, we may apply these algorithms to find all solutions on the interval of nonlinear equations.
Table 1: Numerical results of the Newton method, Algorithm1 and Algorithm2.

| Problem | Algorithms | Interval | $x_0^*$ | No.iterations | $|f(x^*)|$ |
|---------|------------|----------|---------|--------------|------------|
| $f_1(x)$ | Newton     | -        | 0       | 4            | 1.868383e-010 |
|         | ALG1       | (-2.2)   | 1.61565347 | 3             | 6.036838e-014 |
|         | ALG2       | (-2.2)   | 1.54889649 | 2             | 1.046885e-012 |

| $f_2(x)$ | Newton     | -        | 1       | 4            | 8.670376e-011 |
|          | ALG1       | (-1,1)   | -0.24779658 | 3             | 7.808199e-013 |
|          | ALG2       | (-1,1)   | 0.11527742  | 2             | 2.908962e-011 |

| $f_3(x)$ | Newton     | -        | -1      | 7            | 4.340706e-010 |
|          | ALG1       | (-3,0)   | -2.47610403 | 6             | 4.176293e-011 |
|          | ALG2       | (-3,0)   | -2.37004234 | 5             | 7.851186e-011 |

| $f_4(x)$ | Newton     | -        | 0       | 3            | 1.007128e-011 |
|          | ALG1       | (-2,2)   | -0.43105390 | 3             | 2.513791e-12  |
|          | ALG2       | (-2,2)   | 0.33650523  | 2             | 3.783142e-12  |

| $f_5(x)$ | Newton     | -        | -2      | 12           | 1.431096e-010 |
|          | ALG1       | (-3,-1)  | -1.81322143 | 7             | 2.271047e-011 |
|          | ALG2       | (-3,-1)  | -1.59337320 | 4             | 3.459277e-011 |

| $f_6(x)$ | Newton     | -        | 0       | 75           | 6.138201e-012 |
|          | ALG1       | (-1,1)   | -0.46745581 | 53            | 7.904788e-014 |
|          | ALG2       | (-1,1)   | 0.91885053  | 51            | 4.072298e-013 |

References


Received: May, 2011