ComTech: Computer, Mathematics and Engineering Applications, 13(1), June 2022, 45−52 P-ISSN: 2087-1244
DOI: 10.21512/comtech.v13i1.7322 E-ISSN: 2476-907X

The Solution of Non-Linear Equations System Containing Interpolation Functions by Relaxing the Newton Method

Nur Rokhman[1]*, Erwin Eko Wahyudi[2], and Janoe Hendarto[3]

[1-3]Department of Computer Sciences and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada
Sekip Utara, Bulaksumur, Daerah Istimewa Yogyakarta 55281, Indonesia
[1]nurrokhman@ugm.ac.id; [2]erwin.eko.w@ugm.ac.id; [3]jhendarto@ugm.ac.id

Received: 30th April 2021/ **Revised:** 24th June 2021/ **Accepted:** 25th June 2021

How to Cite: Rokhman, N., Wahyudi, E. E., & Hendarto, J. (2022). The Solution of Non-Linear Equations System Containing Interpolation Functions by Relaxing the Newton Method. *ComTech: Computer, Mathematics and Engineering Applications, 13*(1), 45−52. https://doi.org/10.21512/comtech.v13i1.7322

Abstract **-** Many world phenomena lead to non-linear equations systems. For some applications, the non-linear equations which construct the non-linear equations system are interpolation functions. However, the interpolation functions are usually not represented as mathematical expressions but as computer programs in specific programming languages. The research proposed using the relaxed Newton method to solve the non-linear equations system that contained interpolation functions. The interpolation functions were represented in the R programming language. Then, the experiment used the Spline interpolation function to construct a two-dimensional non-linear equations system. Eleven initial guesses, maximum of ten-time iterations, and 10^{-7} precision were applied. The solution of the non-linear equations system and the iteration needed on each initial guess were observed. The experiment shows that the proposed method works well for solving the non-linear equations system constructed by Spline interpolation functions. By observing the initial guesses used in the experiment, there are four possible results: true solution, spurious solution, false solution, and no solution. Applying 11 initial guesses have five initial guesses resulting in true solutions, one initial guess in spurious solution, three initial guesses in false solutions, and one initial guess in no solution. The discussions imply that this method can be generalized to the three-dimensional non-linear equations system or higher dimensions.

Keywords: non-linear equation, interpolation function, Newton method

I. INTRODUCTION

Many world phenomena lead to the problem of non-linear equations systems. The methods of solving non-linear equations systems are very interesting. There are many solutions of the non-linear equations system which are developed from the Newton method. These solutions solve explicit functions which are represented in mathematical notations. On the other hand, the higher implementation of computer technology has raised the development of the mathematical function, which is represented in a computer program, such as interpolation functions. However, to the best of researchers' knowledge, no method has been developed to solve non-linear equations systems with function represented as a computer program.

Interpolation can be viewed as a process of determining a function that passes through the interpolation points. In a complex computation that involves many interpolation points, the interpolation function is not presented explicitly in mathematical notation. The interpolation function is presented in computational steps as a computer program in such cases. The interpolation functions may work simultaneously and build a non-linear equations system. Currently, the methods to solve non-linear equations system work based on explicit functions. However, these methods cannot be used to solve non-linear equations systems that contain interpolation function since the interpolation function is represented as computational steps. Hence, a certain method is needed to solve the non-linear equation system with the interpolation functions since the functions are not presented explicitly in mathematical notation but computational steps.

*Corresponding Author

In general, the solution of a non-linear equation system by using the Newton method and its variants requires an initial value and the derivative of the non-linear equations involved in the system. The solution of a non-linear equation system that does not need any derivative has been carried out by Rokhman (2013). In a non-linear equation that contains interpolation functions, the explicit formulation of the derivative cannot be obtained. Hence, the Newton method is unable to be used. The study of the solution of non-linear equations which contains the interpolation function has been carried out by Rokhman (2017).

The problem of solving the system of non-linear equations can be viewed as a matter of multi-objective optimization. Every equation in the system states an objective function that aims to minimize the difference between the right and left terms of the corresponding equation (Grosan & Abraham, 2008). A new algorithm based on the method of Adomian decomposition convergence basis for solving functional equations is presented. This algorithm can account for all the real answers of a system if a suitable and primary approximation is chosen (Hosseini & Kafash, 2010).

An iterative method for solving the problem of the non-linear equation numerically is suggested. A mathematical proof supports the proposed iteration through the n-dimensional Taylor expansion (Montazeri et al., 2012). Then, a new algorithm is proposed for the solutions of systems of non-linear equations which use a combination of the gradient and the Newton methods. A novel dynamic combinatory is developed to determine the contribution of the methods in the combination. The numerical results prove that the proposed combination algorithm is generally more robust and efficient than other methods on some important and difficult problems (Taheri & Mammadov, 2012).

Three algorithms that work based on a generalization of fourth-order Jarratt formula iterative methods for solving a system of non-linear equations of n-dimension with n-variables are proposed. These new algorithms may be viewed as extensions and generalizations of the existing methods for solving the system of non-linear equations (Khirallah & Hafiz, 2013). Then, a new fourth-order Jarratt-type method for solving systems of non-linear equations is proposed. The local convergence order of this method is proven (Ghorbanzadeh & Soleymani, 2015). A generalization of a family of methods for solving non-linear equations with unknown multiplicities to the system of non-linear equations is also proposed by using a non-zero multi-variable auxiliary function. Different auxiliary functions give different families of the iterative method to find roots with unknown multiplicities (Ahmad et al., 2016).

Moreover, a new two-step Newton-type method with third-order convergence for solving systems of non-linear equations is proposed. The new method works based on the integral interpolation of the Newton method. Its cubic convergence and error equation is proven theoretically and demonstrated numerically (Liu & Fang, 2015). There is also an improvement to the recently introduced factored method for the solution of non-linear equations. In this method, the original system is transformed by adding an offset to all unknowns (Ruiz-Oltra et al., 2016).

A new alternative approximation based on the quasi-Newton approach for solving systems of non-linear equations using the average of midpoint and the Simpson quadrature is conducted. This method reduces the number of iterations to reach a solution (Mohammad & Waziri, 2015). Then, according to Abubakar and Waziri (2016), an enhancement of a three-step Newton-like method where the Jacobian inverse is approximated into a diagonal matrix is proposed. Numerical results prove that the method in terms of CPU time and the number of iterations is very encouraging.

Then, some researchers also propose a fifth-order convergence three-step method for solving a system of non-linear equations. The efficiency index of this method is better than the other three-step methods. The advantages of the method lie in the feature that this technique not only achieves an approximate solution with high accuracy but also improves the calculation speed (Ahmadi et al., 2016).

Two multi-step derivative-free iterative methods for solving a system of non-linear equations are also presented. These methods have high computational efficiency and low computational cost (Wang & Fan, 2016). It has been shown that the proposed method for reformulation and rearrangement of the NLE systems can reduce the dimension of the system, remove discontinuities, and considerably simplify the task of finding initial estimates for the unknowns so that solution is obtained (Shacham & Brauner, 2017).

There are also three higher-order iterative methods of convergence order four, five, and six for solving a system of non-linear equations. New weight functions are included in the second term of the Newton method to get higher-order methods. Higher-order accuracy is achieved with only one inverse of the Jacobian matrix (Madhu & Jayaraman, 2017). Moreover, two basic methods of approximating the solutions of non-linear systems of algebraic equations are considered. The Steepest Descent method obtains a good and sufficient initial guess. The Broyden method replaces the Newton method. The obtained results suggest that the number is reduced compared to the Newton method in the same problem (Mahwash & Gyang, 2018).

A new iterative method based on the quasi-Newton approach for solving systems of non-linear equations, especially large scale, is also proposed. It uses the weighted combination of the Trapezoidal and Simpson quadrature rules (Osinuga & Yusuff, 2018). According to Dwail and Shiker (2021), the Trust Region Method (TRM) is combined with Broyden–Fletcher–Goldfarb–Shanno (BFGS) technique to find the solution to the non-linear systems. Next, two Broyden-like methods are developed from a weighted combination of quadrature rules, namely the

Trapezoidal, Simpson 3/8, and Simpson 1/3 quadrature rules (Isaac et al., 2021).

Based on Halilu and Waziri (2020), there is a derivative-free method for solving large-scale systems of non-linear equations via a double direction approach. The acceleration parameter used in this approach approximated the Jacobian matrix to form a derivative-free method. Then, a new line search algorithm is suggested for solving non-linear equations systems that combine a monotone technique into a modified line search rule. These techniques show that these methods solve the non-linear system with less time and effort without evaluating the derivative (Hashim et al., 2019).

Solving a non-linear equations system using the Newton method needs the derivative of each function to construct the non-linear equations system. The research proposes a novel way to solve a non-linear equations system constructed by the interpolation functions. The interpolation functions are represented in computer programs whose derivatives cannot be represented as explicit functions. The absence of derivative function causes Newton method not to work on solving the non-linear equations system, constructed by the interpolation functions.

II. METHODS

In the Newton method, a numerical solution of a non-linear equation $f(x) = 0$ is found by iterating an initial guess x_0 that moves closer to the solution by using Equation (1). The iteration is stopped when the error tolerance (ε) is reached. Otherwise, no solution can be found. Relaxing the Newton method can be carried out by using the error tolerance (ε) to determine the derivative value of the function. It means that no function derivative is needed (Rokhman, 2011). The Newton method for solving a non-linear equation system needs derivative of the functions constructing the system. The derivative-free method to solve the non-linear equations system can be developed by relaxing the Newton method (Rokhman, 2013). The relaxed Newton method can also be used to solve a non-linear equation containing interpolation functions. An interpolation function is represented in computer program. In this case, no explicit function of its derivative like Equation (1) does not work (Rokhman, 2017).

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad n = 0,1,.. \tag{1}$$

A non-linear equations system with n variables $x_1, \ldots, x_n$ is constructed by n non-linear functions $F_1, \ldots F_n$, generally written as in Equation (2). The solution of the non-linear equations system is the value of variables $x_1, \ldots, x_n$ which are true for all non-linear functions.

$$\begin{aligned} F_1(x_1,\ldots,x_n) &= 0 \\ &\vdots \\ F_n(x_1,\ldots,x_n) &= 0 \end{aligned} \tag{2}$$

Suppose x, F(x), F′(x) are defined as in Equation (3), Equation (4), and Equation (5), respectively with $\frac{\partial F_p}{\partial x_q}$ as a partial derivative of function F_p to variable x_q where $p = 1,\ldots n$, $q = 1, \ldots n$. The matrix in Equation (5) is known as the Jacobian matrix. Then, the Newton method for solving non-linear equation systems can be written as Equation (6) (Atkinson & Han, 2004).

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{3}$$

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} F_1(x_1,\ldots,x_n) \\ \vdots \\ F_n(x_1,\ldots,x_n) \end{bmatrix} \tag{4}$$

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix} \tag{5}$$

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \left(\mathbf{F}'\left(\mathbf{x}^{(n)}\right)\right)^{-1} \mathbf{F}\left(\mathbf{x}^{(n)}\right) \tag{6}$$

The research proposes the application of relaxing the Newton method for solving non-linear equations systems which contain interpolation functions. A non-linear equation system that will be solved in the research is presented in Equation (7). Each $f_1(x)$ and $f_2(x)$ is the results of two-dimensional Spline interpolation function.

$$\begin{aligned} y &= f_1(x) \\ y &= f_2(x) \end{aligned} \tag{7}$$

Basically, a solution of this non-linear equations system is an intersection point of $f_1(x)$ and $f_2(x)$. The non-linear equations systems in Equation (7) can be rewritten, as shown in Equation (8).

$$F(x) = \begin{bmatrix} f_1(x) - y \\ f_2(x) - y \end{bmatrix} \tag{8}$$

Suppose it is $F_1(x, y) - f_1(x) - y$ and $F_2(x, y) - f_2(x) - y$. Then, the non-linear equations systems are a two-dimensional system. It contains two functions, $F_1(x, y)$ and $F_2(x, y)$. The Jacobian matrix of the non-linear equations system can be built as presented in Equation (9).

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial x} & -1 \\ \frac{\partial F_2}{\partial x} & -1 \end{bmatrix} \tag{9}$$

Using a small value of ε, the relaxed Newton method can be used to construct Jacobian matrices of Equation (9) and gives a new Jacobian matrix as presented in Equation (10).

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{F_1(x+\varepsilon,y) - F_1(x,y)}{\varepsilon} & -1 \\ \frac{F_2(x+\varepsilon,y) - F_2(x,y)}{\varepsilon} & -1 \end{bmatrix} \quad (10)$$

Suppose ∂x and ∂y are the difference of x and y variables, respectively. The relaxing of Newton method for solving non-linear equation systems can be constructed, as shown in Equation (11).

$$\begin{bmatrix} \partial x \\ \partial y \end{bmatrix} = -\left(\begin{bmatrix} \frac{F_1(x+\varepsilon,y) - F_1(x,y)}{\varepsilon} & -1 \\ \frac{F_2(x+\varepsilon,y) - F_2(x,y)}{\varepsilon} & -1 \end{bmatrix} \right)^{-1} \begin{bmatrix} F_1(x,y) \\ F_2(x,y) \end{bmatrix} \quad (11)$$

The initial guess for the solutions of the non-linear equations system is (x_0, y_0). Then, the improvement to this initial guess is carried out by using Equation (12). The difference of x and y variables on each iteration, namely ∂x and ∂y can be obtained from the Equation (13). These processes are repeated until the desired error tolerance (ε) is reached.

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} \partial x \\ \partial y \end{bmatrix}, \quad n = 0,1,\ldots \quad (12)$$

An algorithm to find the solution of the non-linear equations system, which contains the Spline interpolation function, can be constructed by using Equations (12) and (13). The algorithm inputs are two sets of interpolation points, a precision value called *eps*, an initial guess (x_0, y_0), and a maximum number of iterations called *maxIter*. When the error tolerance value is reached at t iteration, the output of the algorithm is a point (x_t, y_t). This point is the solution of the non-linear equations system containing the Spline interpolation function.

The first step of the algorithm constructs the interpolation functions. Suppose $f(x)$ and $g(x)$ are the interpolations functions. The second step calculates the derivatives of the $f(x)$ and $g(x)$ at the initial guess by using the relaxed Newton method. These values are used to construct the Jacobian matrix. The improvement values of the initial guess, called *dx* and *dy*, are calculated using the inverse of the Jacobian matrix and the values of the $f(x)$ and $g(x)$ at the initial guess (x_0, y_0). The iteration process is completed by improving the initial guess using *dx* and *dy*, but the error tolerance still needs to be checked. The solution is found if both *dx* and *dy* are less than *eps*. Otherwise, the solution is not yet found, and the processes are repeated with the new value of (x, y).

Under some conditions, the value of the interpolation function may be undefined, or the error tolerance cannot be reached after several *maxIter* iterations. The processes must be stopped, and no solution is found. These conditions are represented in the seventh and eighth steps. The complete algorithm is depicted in Figure 1.

$$\begin{bmatrix} \partial x \\ \partial y \end{bmatrix} = -\left(\begin{bmatrix} \frac{F_1(x_n+\varepsilon,y_n) - F_1(x_n,y_n)}{\varepsilon} & -1 \\ \frac{F_2(x_n+\varepsilon,y_n) - F_2(x_n,y_n)}{\varepsilon} & -1 \end{bmatrix} \right)^{-1} \begin{bmatrix} F_1(x,y) \\ F_2(x,y) \end{bmatrix} \quad (13)$$

Input : two sets of interpolation points,
Precision as **eps**,
Initial guess as (x_0, y_0),
Maximum number of itertion as **maxIter**
Output: (x,y)
Steps :
1. Do the interpolation. f(x) and g(x) as the result
$X = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$
t = 0
found = false
2. dfx = $(f(x_t+eps)-f(x_t))/eps$
dgx = $(g(x_t+eps)-g(x_t))/eps$
$J = \begin{bmatrix} dfx & -1 \\ dgx & -1 \end{bmatrix}$
3. A = $\begin{bmatrix} f(x_t) \\ g(x_t) \end{bmatrix}$
4. D = $\begin{bmatrix} dx \\ dy \end{bmatrix} = -J^{-1}A$
5. $X = X + D$
t = t + 1
6. If $|f(x_t) - g(x_t)| <$ **eps** and then **found** = true
7. If $f(x_t)$ or $g(x_t)$ undefined then **Break**
8. Repeat from step 2 until **found** or (**t** >= **maxIter**)
9. If **found** then the result is (x_t, y_t)

Figure 1 Algorithm of Solving Non-Linear System Containing Interpolation Functions

The experiment is carried out by implementing the algorithm in the R programming language. In this programming language, the construction of the Spline interpolation function is done easily by calling the splinefun function. The research objects are two interpolation functions used by Dontchev et al. (2002). These functions are used to construct the non-linear equations system. The solutions of the system are the intersection points of the interpolation functions. Then, some initial points are used in the experiments. The final result and the number of iterations to obtain the final result of each initial point are observed.

III. RESULTS AND DISCUSSIONS

The research objects are the interpolations points as used in Dontchev et al. (2002). The non-linear equations system is constructed from the interpolation function of interpolation points {(0,0, 0,0), (0,05, 0,7), (0,1, 1,0), (0,2, 1,0), (0,8, 0,3), (0,85, 0,05), (0,9, 0,1), (1,0, 1,0)} and the interpolation points {(0,0, 0,0), (0,1, 0,9), (0,2, 0,95), (0,3, 0,9), (0,4, 0,1), (0,5, 0,05), (0,6, 0,05), (0,8, 0,2), (1,0, 1,0)}. The interpolations are carried out by using the Spline interpolation function. Then, these interpolation functions construct the non-linear equations system which will be solved by using the relaxing of the Newton method.

Suppose the first interpolation gives a function called $f(x)$. The second interpolation has a function called $g(x)$. The graph of these functions is shown in Figure 2. The non-linear equations system to be solved is presented in Equation (14). By observing the graph in Figure 2, the solutions of the non-linear equations system consist of five points in which the x value lies between 0 and 1, namely (0,0, 0,0) and (1,0, 1,0), which are the interpolation points, and three intersection points.

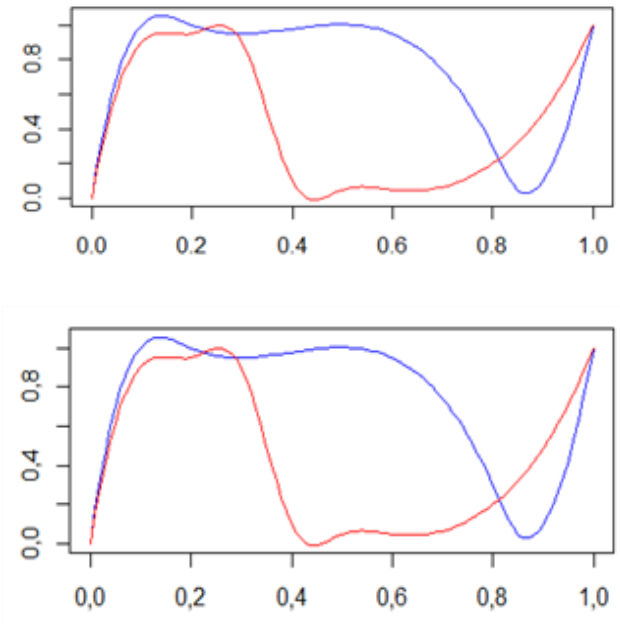

Figure 2 Graph of the Interpolation Functions

$$y = f(x)$$
$$y = g(x) \qquad (14)$$

Then, eleven initial guesses, namely (0,0, 0,0), (0,1, 1,0), (0,2, 2,0), (0,3, 3,0), (0,4, 4,0), (0,5, 5,0), (0,6, 6,0), (0,7, 7,0), (0,8, 8,0), (0,9, 9,0), and (1,0, 10,0) are used in the experiment. The maximum iteration is set to 10. The precision value is set to 0,0000001. Then, the algorithm implemented in R programming language is shown in Figure 3.

The proposed method is used to find the solutions to non-linear equations system containing two Spline interpolation functions in Equation (16). For example, there are five points of the intersection between $f(x)$ and $g(x)$. Even though the points of (0,0, 0,0) and (1,0, 1,0) are trivial solutions, these points are the interpolations points of both interpolation functions.

The performance of relaxing the Newton method for solving non-linear equation system has been observed. Two parameters are observed: the iteration needed to find the solutions (n) and the solutions (x_n, y_n). The experiment results are shown in Table 1. The initial guesses of (0,0, 0,0) and (1,0, 10,0) need only one iteration to get the solution since these initial guesses are the interpolation points of the interpolation functions. Then, the initial guesses of (0,2, 2,0) and (0,4, 4,0) have almost the same solution namely (0,2259318, 0,9756243) and (0,2259319, 0,9756243) respectively. It is the first intersection point. However, the required iterations are different. The initial guess of (0,2, 2,0) requires three times of iterations. Meanwhile, the initial guess of (0,4, 4,0) requires five times of iterations. The detailed observation in processing the initial guess of (0,2, 0,2) and (0,4, 0,4) is shown in Tables 2 and 3, respectively.

```
maxIter <- 10
eps <- 0.0000001
x1 <- c(0.0,0.05,0.1,0.2,0.8,0.85,0.9,1.0)
y1 <- c(0.0,0.7,1.0,1.0,0.3,0.05,0.1,1.0)
f <- splinefun(x1,y1)
x2 <- c(0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.8,1.0)
y2 <- c(0.0,0.9,0.95,0.9,0.1,0.05,0.05,0.2,1.0)
g <- splinefun(x2,y2)
expDataX <-c(0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0
.9,1.0)
expDataY <-c(0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0
,9.0,10.0)
for (i in 1:length(expDataX))
{
  x <- matrix(c(expDataX[i],expDataY[i]),nrow=2,
ncol=1, byrow=TRUE)
  iter <-0
  repeat
  {
    iter <-iter + 1
    dfdx <- (f(x[1]+eps)-f(x[1]))/eps
    dfdy <- -1
    dgdx <- (g(x[1]+eps)-g(x[1]))/eps
    dgdy <- -1
    jacobi <- matrix(c(dfdx,dfdy,dgdx,dgdy),
nrow=2,ncol=2,byrow=TRUE)
    A <- matrix(c(f(x[1])-x[2],g(x[1])-x[2]))
    dxdy <- -solve(jacobi)%*%A
    x <- x+dxdy
    if (abs(f(x[1])-g(x[1]))<eps)
    {
      found<-TRUE
      break
    }
    if (iter >= maxIter) break
  }
print(c(expDataX[i],expDataY[i],iter,x[1],x[2]))
}
```

Figure 3 R Program to Find the Solution

Table 1 The Experimental Results

No	(x_0, y_0)	n	(x_n, y_n)
1	(0,0, 0,0)	1	(0,0, 0,0)
2	(0,1, 1,0)	9	(-1,029962e-08, -7,092949e-08)
3	(0,2, 2,0)	3	(0,2259318, 0,9756243)
4	(0,3, 3,0)	4	(0,2879449, 0,9510676)
5	(0,4, 4,0)	5	(0,2259319, 0,9756243)
6	(0,5, 5,0)	4	(0,8124135, 0,2264703)
7	(0,6, 6,0)	7	(1,824736, 22,33810)
8	(0,7, 7,0)	10	(-0,0312172, -0,2897813)
9	(0,8, 8,0)	2	(0,8124135, 0,2264695)
10	(0,9, 9,0)	8	(1,824736, 22,338101)
11	(1,0, 10,0)	1	(1,0, 1,0)

Table 2 The Detail Observation of Initial Guess of (0,2, 0,2)

n	(x_n, y_n)	$\lvert f(x_n) - g(x_n)\rvert$
1	(0,2293685, 0,9668224)	0,006472153
2	(0,2258565, 0,9756006)	0,00014464
3	(0,2259318, 0,9756243)	5,269591e-08

Table 3 The Detail Observation of Initial Guess of (0,4, 0,4)

n	(x_n, y_n)	$\lvert f(x_n) - g(x_n)\rvert$
1	(0,2468062, 0,9178059)	0,03282423
2	(0,2162326, 0,9777921)	0,01918141
3	(0,2258095, 0,9750844)	0,00023499
4	(0,2259318, 0,9756243)	1,385406e-07
5	(0,2259319, 0.9756243)	1,876277e-14

The initial guess of (0,3, 3,0) gives solution of (0,2879449, 0,9510676) after four times iterations. The detailed observation in processing the initial guess of (0,3, 0,3) is shown in Table 4. According to Figure 2, this initial point finds the second intersection point. Table 1 shows the fact that only initial guess (0,3, 0,3) finds the second intersection point.

Table 4 The Detail Observation of Initial Guess of (0,3, 0,3)

n	(x_n, y_n)	$\lvert f(x_n) - g(x_n)\rvert$
1	(0,2902525, 0,9505274)	0,007918907
2	(0,2880567, 0,9510382)	0,0003654196
3	(0,2879452, 0,9510675)	9,246800e-07
4	(0,2879449, 0,9510676)	8,073542e-12

The initial guesses of (0,5, 5,0) and (0,8, 8,0) have almost the same solution, namely (0,8124135, 0,2264703) and (0,8124135, 0,2264695), respectively. However, the required iterations are different. The initial guess of (0,5, 5,0) requires four times of iterations. Meanwhile, the initial guess of (0,8, 8,0) needs two times of iterations. The detailed observation in processing initial guesses (0,5, 5,0) and (0,8, 8,0) are shown in Tables 5 and 6, respectively.

Table 5 The Detail Observation of Initial Guess of (0,5, 0,5)

n	(x_n, y_n)	$\lvert f(x_n) - g(x_n)\rvert$
1	(1,4095988, 0,9933773)	4,5814418
2	(0,81692703, -7,30583402)	0,03670788
3	(0,8123981, 0,2262701)	0,0001253189
4	(0,8124135, 0,2264703)	9,604448e-11

Table 6 The Detail Observation of Initial Guess of (0,8, 0,8)

n	(x_n, y_n)	$\lvert f(x_n) - g(x_n)\rvert$
1	(0,8127117, 0,2258902)	0,002429513
2	(0,8124135, 0,2264695)	7,154819e-08

Table 5 shows the fact that the final iteration of the initial guess of (0,5, 0,5) finds the solution, but there is a point moves out from the interpolation range. So, the solution is not the real true solution. It is called a spurious solution. Table 6 shows the fact that the initial guess (0,8, 0,8) gives real true solution. There is no iteration point which moves out from the interpolation range by using this initial guess.

Next, initial guesses of (0,1, 1,0), (0,6, 6,0), and (0,9,9,0) give solutions as following (-1,029962e-08, -7,092949e-08), (1,824736, 22,33810), and (1,824736, 2233810) with the final precisions 2,729921e-08, 1,350031e-13, and 9,237056e-14, respectively. These are false solutions since they move out from the interpolation range. However, the initial guess of (0,7, 7,0) has no result. The process reaches the maximum number of iterations but does not meet the required precision. The final value of this initial guess is $(x_{10}, y_{10}) = -0{,}0312172, -0{,}2897831)$. The value of $f(0{,}0312172)$ is $-0{,}7068187$. The value of $g(0{,}0312172)$ is $-0{,}5983234$. Both values give $|f(x_{10}) - g(x_{10})| > \varepsilon$ that means no solution is found.

Based on the experiment results, there are four categories of possible results for implementing the relaxing Newton method for solving non-linear equations system. They are true solution, spurious solution, false solution, and no solution. The initial guesses of (0,0, 0,0), (0,2, 2,0), (0,3, 3,0), (0,4, 4,0), (0,8, 8,0), and (1,0, 10,0) result in true solutions on

this experiment. Then, the initial guess of (5,0, 5,0) gives a spurious solution. On the other hand, the initial guesses of (0,1, 1,0), (0,6, 6,0), and (0,9, 9,0) result in false solutions. Last, the initial guess of (0,7, 7,0) does not find the solution.

The relaxing of the Newton method can also be implemented for solving non-linear equations system which contains the three-dimensional interpolation functions. For this case, suppose Equations (7), (8), and (9) are advanced by Equations (15), (16), and (17), respectively. The initial guess is (x_1^0, x_2^0, y^0).

$$\begin{aligned} y &= f_1(x_1, x_2) \\ y &= f_2(x_1, x_2) \\ y &= f_3(x_1, x_2) \end{aligned} \tag{15}$$

$$\begin{aligned} F_1(x_1, x_2, y) &= f_1(x_1, x_2) - y \\ F_2(x_1, x_2, y) &= f_2(x_1, x_2) - y \\ F_3(x_1, x_2, y) &= f_3(x_1, x_2) - y \end{aligned} \tag{16}$$

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & -1 \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & -1 \\ \frac{\partial F_3}{\partial x_1} & \frac{\partial F_3}{\partial x_2} & -1 \end{bmatrix} \tag{17}$$

It applies the relaxed Newton method via introducing the error tolerance (ε) for approximating the partial derivative of F_1, F_2, and F_3 functions to the x_1 $\frac{\partial F_1}{\partial x_2}$ x_2 variables. As an example, the partial derivative means the ratio between the change of F_1 if there is ε change in x_2 variable and the change of x_2, that is ε. The values of each partial derivative in Equation (17) can be determined using Equations (18)−(23).

$$\frac{\partial F_1}{\partial x_1} = \frac{F_1(x_1 + \varepsilon, x_2, y) - F_1(x_1, x_2, y)}{\varepsilon} \tag{18}$$

$$\frac{\partial F_1}{\partial x_2} = \frac{F_1(x_1, x_2 + \varepsilon, y) - F_1(x_1, x_2, y)}{\varepsilon} \tag{19}$$

$$\frac{\partial F_2}{\partial x_1} = \frac{F_2(x_1 + \varepsilon, x_2, y) - F_2(x_1, x_2, y)}{\varepsilon} \tag{20}$$

$$\frac{\partial F_2}{\partial x_2} = \frac{F_2(x_1, x_2 + \varepsilon, y) - F_2(x_1, x_2, y)}{\varepsilon} \tag{21}$$

$$\frac{\partial F_3}{\partial x_1} = \frac{F_3(x_1 + \varepsilon, x_2, y) - F_3(x_1, x_2, y)}{\varepsilon} \tag{22}$$

$$\frac{\partial F_3}{\partial x_2} = \frac{F_3(x_1, x_2 + \varepsilon, y) - F_3(x_1, x_2, y)}{\varepsilon} \tag{23}$$

The general form of the relaxing Newton method for solving non-linear equations system is presented in Equation (24) and Equation (25). The solution of Equation (25) gives the values of ∂x_1, ∂x_2, and ∂y, which are used to improve the values of x_1^n, x_2^n, and y^n, respectively.

$$\begin{bmatrix} x_1^{n+1} \\ x_2^{n+1} \\ y^{n+1} \end{bmatrix} = \begin{bmatrix} x_1^n \\ x_2^n \\ y^n \end{bmatrix} + \begin{bmatrix} \partial x_1 \\ \partial x_2 \\ \partial y \end{bmatrix} \tag{24}$$

$$\begin{bmatrix} \partial x_1 \\ \partial x_2 \\ \partial y \end{bmatrix} = -\left(\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & -1 \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & -1 \\ \frac{\partial F_3}{\partial x_1} & \frac{\partial F_3}{\partial x_2} & -1 \end{bmatrix} \right)^{-1} \begin{bmatrix} F_1(x_1^n, x_2^n, y^n) \\ F_2(x_1^n, x_2^n, y^n) \\ F_3(x_1^n, x_2^n, y^n) \end{bmatrix} \tag{25}$$

IV. CONCLUSIONS

Nowadays, many functions in the computer program form, like the interpolation function, have been developed. From the discussions, it can be concluded that the relaxing of the Newton method can be used to solve the non-linear equations system which contains the interpolation functions. The research shows the fact that the solution to systems constructed by these functions can be found by relaxing the Newton method. It means, the non-linear equations system containing such functions can be solved easily by relaxing the Newton methods. An initial guess may give a true solution, a spurious solution, a false solution, or no solution. However, there is no guarantee for an initial point to give the result a true solution, a spurious solution, a false solution, or no solution. The research discusses an experiment in two-dimensional data. Theoretically, it can be expanded to three-dimensional data or generalized to *n*-dimensional data.

The research is limited by the absence of an explicit form of the interpolation function. It makes the properties exploration of the non-linear equations system containing the interpolation function cannot be conducted yet. Future research can be addressed to explore the properties of such a non-linear equations system.

ACKNOWLEDGEMENT

The authors are indebted to the Department of Computer Science and Electronics which has given full support to the research. Hopefully, the research makes a major contribution to the advancement of numerical solutions to non-linear equations systems.

REFERENCES

Abubakar, A. B., & Waziri, M. Y. (2016). A matrix-free approach for solving systems of nonlinear equations. *Journal of Modern Methods in Numerical Mathematics*, *7*(1), 1–9.

Ahmad, F., Serra-Capizzano, S., Ullah, M. Z., & Al-Fhaid, A. S. (2016). A family of iterative methods for solving systems of nonlinear equations having unknown multiplicity. *Algorithms*, *9*(1), 1–10. https://doi.org/10.3390/a9010005

Ahmadi, M., Esmaeili, H., & Erfanifar, R. (2016). Solving system of nonlinear equations by using a new three-step method. *Control and Optimization in Applied Mathematics*, *1*(2), 53–62.

Atkinson, K., & Han, W. (2004). *Elementary numerical analysis*. Wiley.

Dontchev, A. L., Qi, H. D., Qi, L., & Yin, H. (2002). A Newton method for shape-preserving spline interpolation. *SIAM Journal on Optimization*, *13*(2), 588–602. https://doi.org/10.1137/S1052623401393128

Dwail, H. H., & Shiker, M. A. K. (2021). Using Trust Region method with BFGS technique for solving nonlinear systems of equations. *Journal of Physics: Conference Series*, *1818*, 1–9. https://doi.org/10.1088/1742-6596/1818/1/012022

Ghorbanzadeh, M., & Soleymani, F. (2015). A quartically convergent Jarratt-type method for nonlinear system of equations. *Algorithms*, *8*(3), 415–423. https://doi.org/10.3390/a8030415

Grosan, C., & Abraham, A. (2008). A new approach for solving nonlinear equations systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *38*(3), 698–714. https://doi.org/10.1109/TSMCA.2008.918599

Hashim, K. H., Dreeb, N. K., Dwail, H. H., Mahdi, M. M., Wasi, H. A., Shiker, M. A. K., & Hussein, H. A. (2019). A new line search method to solve the nonlinear systems of monotone equations. *Journal of Engineering and Applied Sciences*, *14*, 10080–10086. https://doi.org/10.36478/jeasci.2019.10080.10086

Halilu, A. S., & Waziri, M. Y. (2020). Solving systems of nonlinear equations using improved double direction method. *Journal of the Nigerian Mathematical Society*, *39*(2), 287–301.

Hosseini, M. M., & Kafash, B. (2010). An efficient algorithm for solving system of nonlinear equations. *Applied Mathematical Sciences*, *4*(3), 119–131.

Isaac, A., Stephen, T. B., & Seidu, B. (2021). A new Trapezoidal-Simpson 3/8 method for solving systems of nonlinear equations. *American Journal of Mathematical and Computer Modelling*, *6*(1), 1–8. https://doi.org/10.11648/j.ajmcm.20210601.11

Khirallah, M. Q., & Hafiz, M. A. (2013). Solving system of nonlinear equations using family of Jarratt methods. *International Journal of Differential Equations and Applications*, *12*(2), 69–83. https://doi.org/10.12732/ijdea.v12i2.931

Liu, Z., & Fang, Q. (2015). A new Newton-type method with third-order for solving systems of nonlinear equations. *Journal of Applied Mathematics and Physics*, *3*(10), 1256–1261. https://doi.org/10.4236/jamp.2015.310154

Madhu, K., & Jayaraman, J. (2017). Some higher order Newton-like methods for solving system of nonlinear equations and its applications. *International Journal of Applied and Computational Mathematics*, *3*, 2213–2230. https://doi.org/10.1007/s40819-016-0234-z

Mahwash, K. N., & Gyang, G. D. (2018). Numerical solution of nonlinear systems of algebraic equations. *International Journal of Data Science and Analysis*, *4*(1), 20–23. https://doi.org/10.11648/j.ijdsa.20180401.14

Mohammad, H., & Waziri, M. Y. (2015). On Broyden-like update via some quadratures for solving nonlinear systems of equations. *Turkish Journal of Mathematics*, *39*, 335–345. https://doi.org/10.3906/mat-1404-41

Montazeri, H., Soleymani, F., Shateyi, S., & Motsa, S. S. (2012). On a new method for computing the numerical solution of systems of nonlinear equations. *Journal of Applied Mathematics*, *2012*, 1–15. https://doi.org/10.1155/2012/751975

Osinuga, I. A., & Yusuff, S. O. (2018). Quadrature based Broyden-like method for systems of nonlinear equations. *Statistics, Optimization & Information Computing*, *6*(1), 130–138. https://doi.org/10.19139/soic.v6i1.471

Rokhman, N. (2011). Secton: A combination of Newton Method and Secant method for solving non linear equations. In *Seminar Nasional Aplikasi Teknologi Informasi (SNATI)* (pp. 50–52).

Rokhman, N. (2013). The implementation of Secton method for solving systems of non linear equations. In *International Conference on Engineering and Technology Development (ICETD)* (pp. 80–84).

Rokhman, N. (2017). Solving non-linear equations containing spline interpolation function by relaxing the Newton method. In *2017 Second International Conference on Informatics and Computing (ICIC)* (pp. 1–5). IEEE. https://doi.org/10.1109/IAC.2017.8280641

Ruiz-Oltra, J. M., Gómez-Quiles, C., & Gómez-Expósito, A. (2016). Offset-assisted factored solution of nonlinear systems. *Algorithms*, *9*(1), 1–14. https://doi.org/10.3390/a9010002

Shacham, M., & Brauner, N. (2017). Solving a system of nonlinear algebraic equations: You only get error messages-What to do next? *Chemical Engineering Education*, *51*(2), 75–82.

Taheri, S., & Mammadov, M. (2012). Solving systems of nonlinear equations using a globally convergent optimization algorithm. *Global Journal of Technology & Optimization*, *3*, 132–138.

Wang, X., & Fan, X. (2016). Two efficient derivative-free iterative methods for solving nonlinear systems. *Algorithms*, *9*(1), 1–10. https://doi.org/10.3390/a9010014