Derivative Free Iterative Method for Solving Nonlinear Equations with Stability Analysis

Muhammad Hassan, Muhammad Ali, Muhammad Waseem

Department of Mathematics and Statistics, University of Southern Punjab, Multan, Pakistan

Abstract: This work presented a novel derivative-free sixth-order iterative method for solving nonlinear equations. This method possesses an efficiency index of 1.56. Convergence analysis is provided in order to prove the reliability of our method. Numerical comparison with other existing methods is provided to demonstrate the efficiency of our method. Plots of Residual fall graphs and stability analysis of all methods have been presented in this work. These plots highlight the efficiency and robustness of our method.

Keywords: Non-Linear Equations, Stability Analysis, Residual Fall Graphs.

Email: mhassan4800@gmail.com

1. Introduction

Solving non-linear equations is a fundamental challenge in engineering, physics, and applied mathematics, as these equations frequently arise in real-world applications. Some of the uses are rocket speed calculation, calculation of eigenvalue in structural mechanics, modeling of compressibility of gases, mitochondrial aging analysis, simulation of simple harmonic oscillation, calculation of material temperature, calculation of drug plasma concentration, and design of intricate systems. Calculating zeros of nonlinear equations is one of the most important activities in this area. Solving polynomials for roots has been a basic problem in computational and applied mathematics, and its range is in all areas of modern science. In the majority of engineering fields, many problems can be reduced to nonlinear equations by mathematical means so that they can be solved numerically. Analytical solutions are not possible or practical in most cases and must be handled by iterative processes to estimate the solution. Another extremely important consideration in the selection of any iteration scheme is the choice of a suitable first estimate because the performance of the scheme, the order and rate of convergence, to a large extent, both depend on it.

Various schemes have been developed by researchers to calculate the roots of nonlinear equations. The schemes have been developed through methodologies like the Taylor series, the Homotopy analysis method, the Hermite interpolation method, and other mathematical methods.

One-step iterative process is another name used for one-point iterative process, but for those that cover two or more steps, multi-point iterative schemes. One of the principal challenges in deriving one-point higher-order iterative processes is determining higher-order derivatives of

a function, an attempt that would significantly complicate most nonlinear equations. For dealing with this constraint, numerical analysis researchers have concentrated on designing creative multi-point iteration techniques to realistically estimate solutions for nonlinear equations. One of the most popular methods is Newton's method.

The classical Newton's method is a widely recognized approach for solving nonlinear equations, derived from the first two terms of the Taylor series expansion of $f(x)$. The Taylor series around a point x_0 can be expressed as:

$$f(x) = f(x_0) + (x - x_0)f'(x_0). \quad (1)$$

where $f(x_0)$ and $f'(x_0)$ are the function value and its derivative at x_0, respectively. Setting $f(x) = 0$ and solving Eq. (1) for x, we obtain:

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}, \quad (2)$$

This can be written in its iterative form as:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad (3)$$

Newton's Method is a clever way to find the roots of a function (where it crosses zero). You start with a rough guess, say x_n, and then the method gives you a better guess, $x_{n+1,}$ by using a formula that "corrects" your initial estimate. Thanks to Traub's work back in 1964, we know this method is super-fast—it roughly doubles the number of correct digits with each step. That's why researchers have come up with **derivative-free methods**—smarter alternatives that work just as well (or even better in some cases) without needing derivatives. These are super handy in fields like engineering, finance, and AI, where you want speed and simplicity without sacrificing accuracy. Numerous such methods exist in the literature, as documented in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23].

In this paper, we introduce a new way to solve nonlinear equations that's both fast and practical. Our method builds on Traub's classic two-step approach (1964) [1] but adds a clever third step inspired by Newton's method. The best part? It *doesn't need derivatives* at all—yet it converges six times faster per step than basic methods. Derivatives can be a headache in real-world problems (think complex models or data-driven applications). By ditching them without sacrificing speed, our method offers a simpler, more efficient tool for scientists and engineers. It's like upgrading from a manual calculator to a high-speed solver—same accuracy, fewer steps.

The remainder of the paper is structured as follows. In Section 2, we present basic definitions for the estimation of error and computational order. Some existing methods showcasing the sixth-order convergence are presented in Section 3. Section 4 details the development of the proposed method, including its theoretical convergence analysis. Section 5 presents numerical experiments and comparisons with existing methods. In Section 5.1, we analyze the residual fall graphs and stability of the scheme. Finally, conclusions and future directions are discussed in Section 6.

2. Definitions

In this section, we define key concepts essential for understanding the analysis of iterative methods used to solve nonlinear equations.

2.1. Error Analysis

In numerical methods, errors tell us how close our approximations are to the real solution. Think of it like measuring how much your guess improves with each step—or how far it still is from the exact answer. By setting a "tolerance" (basically, how small an error we're willing to accept), we can stop calculations once the result is good enough.

2.2 Absolute Error

Absolute error, one of the most commonly used error measures, is the absolute difference between the exact value and the approximate value. Let E represent the absolute error, then:

$E = |Exact\ Value\ -\ Approximate\ Value|$.

Absolute error provides a direct measure of the magnitude of deviation from the exact solution.

2.2.1 Computational Order of Convergence (COC)

Computational order of convergence (COC) is a numerical indicator that quantifies how fast errors in an iterative process diminish. For iterative algorithms,

COC is given as:

$$COC \cong \frac{\ln\left|\frac{e^*_{m+1}}{e^*_m}\right|}{\ln\left|\frac{e^*_m}{e^*_{m-1}}\right|}$$

2.3 Stopping Criteria

In numerical computation, iterative methods need a stopping criterion to terminate the process at some level of accuracy. Stopping criteria are usually based on the following conditions:

$$|x_m - x_{m-1}| \leq \lambda,$$

Where λ is the tolerance. All of these conditions guarantee that iterative steps terminate as the solution converges to a predetermined degree of accuracy.

2.4. Order of Convergence

The order of convergence is the rate at which an iterative method converges to the solution. If a method converges with order q, then it is true that:

$$|x_{n+1} - \alpha| \leq C|x_n - \alpha|^q,$$

Where $C > 0$ is a constant, and $q > 1$ is the order of convergence. Higher-order methods converge more quickly and are thus preferable to employ in solving nonlinear equations.

2.5. Residual Error Analysis

Residual error is the size of the function value at the approximation solution. At a particular iteration x_n, the residual error can be written as:

$$R_n = |f(x_n)|$$

Residual error analysis plays an important role in testing the convergence and stability of an iterative method.

2.6. Graphical Representation of Errors

Graphical techniques like residual fall graphs are used to graphically illustrate the reduction of error with iterations. Such graphs convey information regarding the convergence and stability behavior of a numerical technique.

2.7. Stability Analysis

Stability is a critical characteristic of numerical algorithms. Stability guarantees that slight changes in the initial estimate or the intermediate steps do not result in divergence. In this work, we investigate the stability of our proposed algorithm by examining its behavior on a set of test functions.

3. Overview of Existing Methods

Prior to going through the discussion and derivation of the introduced technique, we first briefly provide an overview of some of the basic techniques found in the literature. These established methods will serve as benchmarks for comparison with the results obtained from our proposed approach. This comparison aims to highlight the effectiveness and potential advantages of our new method. In addition, we will evaluate each method's performance by analyzing the residual errors and presenting stability analysis graphs.

For comparison, we will consider the following methods:

1. Rafiullah and Haleem [24] proposed the following three-step iterative technique

without the evaluation of the second derivative of the function, and it contains two functions and two first derivatives per iteration. The method they devised contains the following iterative scheme:

$$w_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$v_k = x_k - \frac{f(x_k)}{f'(x_k)} + \frac{f(x_k)f'(w_k) - f'(x_k)}{2f'(x_k)^2}, \tag{3}$$

$$x_{k+1} = v_k + \frac{2f(v_k)f'(w_k)}{f'(v_k)^2 - 4f'(x_k)f'(w_k)^2}.$$

We call this method **RF.**

2. S.K. Parhi and D.K. Gupta [25] derived a sixth-order iterative method for solving nonlinear equations. The method has the following form:

$$w_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$v_k = w_k - \frac{2f(x_k)}{f'(x_k) + f'(w_k)},$$

$$x_{k+1} = -v_k - \frac{f(v_k)}{f'(v_k)} \frac{f'(x_k) + f'(w_k)}{3f'(w_k) - f'(x_k)}.$$

We call this method **SKP.**

3. Sukhjit Singh and D.K. Gupta [26] describe a novel iterative method of sixth order that was presented to determine the real roots of nonlinear equations. Starting with a suitable x_0, the approach generates a sequence of iterations that converges to the root. The method has the following form:

$$w_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$v_k = w_k - \frac{f(w_k)}{f'(x_k)} \frac{f(x_k) - \beta f(w_k)}{f(x_k) + (\beta - 2)f(w_k)},$$

$$x_{k+1} = v_k - \frac{f(v_k)}{f'(x_k)} \frac{f(x_k) - f(w_k) + \gamma f(v_k)}{f(x_k) - 3f(w_k) + \gamma f(v_k)}.$$

We call this method **SSE.**

4. Mirzaee and Hamzeh [27] proposed a sixth-order numerical iterative scheme as follows:

$$w_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$v_k = w_k - \frac{f(w_k)}{f'(x_k)} \frac{f(w_k) - f(x_k)}{2f(w_k) - f(x_k)},$$

$$x_{k+1} = v_k - \frac{f(v_k)f(x_k)(2f(w_k) - f(x_k))}{f'(x_k)(4f(w_k)f(x_k) - 2f(w_k)^2 - f(x_k)^2}.$$

We call this method **FME.**

5. Han Li [28] proposed a novel four-step iterative scheme as follows:

$$w_k = x_k + \frac{f(x_k)}{f'(x_k)},$$

$$v_k = w_k - \frac{f(w_k)}{f'(w_k)},$$

$$b_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$x_{k+1} = v_k + \frac{2f(x_k)^2}{f'(x_k)^2 - 4f'(x_k)f'(b_k) + f(b_k)^2}\frac{f(v_k)}{f'(x_k)},$$

We refer to this method as **HLI.**

6. Obadah Said Solaiman and Ishak Hashim [29] proposed two novel sixth-order iterative schemes using Hallye's method.

a) **Method 1**

$$w_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$x_{k+1} = w_k - \frac{f(w_k)}{f'(w_k)} - \frac{2f(w_k)^2 f'(w_k)f''(w_k)}{4f'(w_k)^4 - 4f(w_k)f'(w_k)^2 f''(w_k) + f(w_k)^2 f''^{(w_k)^2}}.$$

We call this method **OBS1.**

b) **Method 2**

$$w_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$x_{k+1} = w_k - \frac{f(w_k)}{f'(w_k)} - \frac{2f(w_k)^2 f'(w_k)Q(x_k, w_k)}{4f'(w_k)^4 - 4f(w_k)f'(w_k)^2 Q(x_k, w_k) + f(w_k)^2 Q(x_k, w_k)^2}.$$

Where $Q(x_k, w_k) = \frac{2}{x_k - w_k}\left[3\frac{f(x_k)-f(w_k)}{x_k - w_k} - 2f'(w_k) - f'(x_k)\right]$.

We call this method **OBS2.**

4. Development and Convergence Analysis of the Proposed Method

In this section, we present a new three-step, sixth-order iterative method. We begin by considering an existing method, as discussed in [1], and modify it to create a completely derivative-free approach. The formulation of the iterative

The expression is as follows:

$$b_k = x_k - \frac{f(x_k)}{f'(x_k)}.$$

$$v_k = b_k - \frac{f(b_k)}{f'(b_k)}$$

Introduce a third step in the above method as:

$$x_{k+1} = v_k - \frac{f(v_k)}{f'(v_k)}$$

We attempt to build a derivative-free algorithm from the above iteration scheme. To this effect, we suggest a new formula for w_n:

$$w_k = x_k - \beta f(x_k)$$

Next, we approximate the derivatives using first-order divided differences:

$$f'(x_k) \cong \frac{f(w_k) - f(x_k)}{w_k - x_k} = f[w_k, x_k].$$

$$f'(b_k) \cong \frac{f(b_k) - f(x_k)}{b_k - x_k} = f[b_k, x_k].$$

We also approximate $f'(v_k)$ using a linear combination of divided differences:

$$f'(v_k) \cong f[b_k, v_k] + f[x_k, b_k] - f[x_k, v_k].$$

4.1. Algorithm 1

The proposed method is defined as follows:

$$w_k = x_k - \beta f(x_k)$$

$$b_k = x_k - \frac{f(x_k)}{f[w_k, x_k]},$$

$$v_k = b_k + \frac{f(b_k)}{f[b_k, x_k]}$$

$$x_{k+1} = v_k + \frac{f(v_k)}{f[b_k, v_k] + f[x_k, b_k] - f[x_k, v_k]}.$$

We denote the above method as **MHW.** We set $\beta = 0.5$.

This method has a convergence order of sixth and an efficiency index of $p^{\frac{1}{d}} = 6^{\frac{1}{4}} = 1.565$.

4.2. Analysis of Convergence

Theoretical confirmation of the convergence property of the derived iterative process is made through the following theorem. Using the computational capabilities of MAPLE 2023, we rigorously demonstrate that the convergence order of Algorithm 1 is sixth-order.

Theorem 1:

Let $\alpha \in \alpha \in \alpha \in I$ be a simple root of a function $f: I \subseteq \mathbb{R} \to \mathbb{R}$ in an open interval I. If the initial approximation x_0 is sufficiently close to α, then the derivative-free method defined by Algorithm 1 converges with order six.

Proof:

We begin with the Taylor expansion of the function $f(x_k)$ around the root α:

$$f(x_k) = e_k + c_2 e_k^2 + \cdots + c_{12} e_k^{12}$$

Similarly, the expansion for w_k is:

$$w_k = e_k + c_2 e_k^2 + \cdots + c_9 e_k^2.$$

Next, we compute the expansion of $\theta(w_k)$:

$$f(w_k) = \frac{1}{2} e_k - \frac{1}{2} c_2 e_k^2 + \left(-\frac{3}{8} c_3 - \frac{1}{2} c_2^2\right) e_k^3 + \cdots$$

Now, we compute the divided difference:

$$\frac{f(w_k) - f(x_k)}{w_k - x_k} = 1 + \frac{3}{2} c_2 e_k + \left(\frac{7}{4} c_3 - \frac{1}{2} c_2^2\right) e_k^2 + \cdots$$

Similarly, the expansions for b_k and $f(b_k)$ are computed, and the corresponding divided differences are derived. Through a similar process, we can also expand v_k and calculate the necessary terms for evaluating the convergence order. The detailed expansions for each step indicate that the error in each iteration is dominated by terms of order e_k^6. Hence, the proposed method has convergence order six.

4.3. Comparison of Efficiency Indices

In the context of iterative methods, the efficiency index (EI) provides a standardized framework for comparing the computational efficiency of various methods. The EI integrates the order of convergence (r) and the number of functional evaluations (N_f), offering a comprehensive assessment. The relationship is mathematically expressed as:

$$EI = r^{\frac{1}{N_f}}.$$

This allows researchers to evaluate the relative performance of different iterative techniques by quantifying their computational efficiency. Table 1 presents a comparison of the efficiency indices for various methods.

Table 1. Comparison of Efficiencies of Various Methods

Methods	N_f	EI
RF	4	1.56
SKP	5	1.43
SSE	5	1.43
FME	4	1.56
HLI	6	1.33
OBS1	5	1.43
OBS2	4	1.56
MHW	4	1.56

5. Numerical Examples

In this section, we present numerical examples to illustrate the efficiency index of the MHW method. We compare the performance of several iterative methods, including Rafiullah (RF), S.K. Parhi (SKP), Sukhjit Singh (SSE), F. Mirzaee (FME), Han Li (HLI), Obadah (OBS1, OBS2), and MHW. For these examples, we set the tolerance value as $epsilon = E - 1000$.

Table 2. Test Functions

Functions	Exact Roots
$f_1 = xe^x + \ln(x^4 + x + 1)$	0
$f_2 = \sin(x) - 10x + 1$	0.1111
$f_3 = \cos(x) - 2x + 5$	2.2041
$f_4 = x^3 - 2$	1.2599
$f_5 = \ln(x) - \cos(x)$	1.3030
$f_6 = x^3 - 10$	2.1544
$f_7 = x^2 + \sin\left(\frac{x}{5}\right) - \frac{1}{4}$	0.4100

Table 3. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_1 and $x_0 = 0.5$.

Methods	N	N_f	$\lvert f(x_{n+1})\rvert$
RF	5	25	1×10^{-118}
SKP	6	36	9×10^{-1002}
SSE	5	30	0
FME	5	30	7×10^{-1002}
HLI	8	48	8×10^{-1002}

OBS1	6	36	1×10^{-1001}
OBS2	6	36	3×10^{-1002}
OBS2	4	16	1×10^{-1000}

Table 4. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_2 and $x_0 = 0.7$.

Methods	N	N_f	$\lvert f(x_{n+1})\rvert$
RF	5	25	0
SKP	6	36	4×10^{-1000}
SSE	4	24	4×10^{-1000}
FME	5	30	4×10^{-1000}
HLI	8	48	0
OBS1	5	30	0
OBS2	4	26	0
OBS2	4	16	0

Table 5. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_3 and $x_0 = 0.6$.

Methods	N	N_f	$\lvert f(x_{n+1})\rvert$
RF	5	25	0
SKP	6	36	0
SSE	4	24	4×10^{-869}
FME	5	30	4×10^{-1000}
HLI	9	54	0
OBS1	6	36	0
OBS2	5	30	0
OBS2	4	16	4×10^{-508}

Table 6. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_4 and $x_0 = 0.8$

Methods	N	N_f	$\lvert f(x_{n+1})\rvert$
RF	*Diverge*		
SKP	6	30	2×10^{-999}
SSE	5	30	2×10^{-999}

FME	7	42	2×10^{-999}
HLI	10	60	2×10^{-999}
OBS1	7	42	2×10^{-999}
OBS2	5	30	2×10^{-999}
OBS2	4	16	6×10^{-628}

Table 7. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_5 and $x_0 = 2.5$.

Methods	$\boldsymbol{N}$	$\boldsymbol{N_f}$	$\lvert \boldsymbol{f}(\boldsymbol{x_{n+1}}) \rvert$
RF	9	45	0
SKP	7	42	3×10^{-1000}
SSE	4	24	6×10^{-732}
FME	7	42	3×10^{-1000}
HLI	10	60	3×10^{-1000}
OBS1	7	42	3×10^{-1000}
OBS2	5	30	3×10^{-1000}
MHW	4	16	3×10^{-1000}

Table 8. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_5 and $x_0 = 2.5$.

Methods	$\boldsymbol{N}$	$\boldsymbol{N_f}$	$\lvert \boldsymbol{f}(\boldsymbol{x_{n+1}}) \rvert$
RF	*Diverge*		
SKP	8	40	2×10^{-999}
SSE	6	36	2×10^{-999}
FME	8	48	2×10^{-999}
HLI	*Diverge*		
OBS1	8	48	2×10^{-999}
OBS2	6	36	2×10^{-999}
MHW	6	24	2×10^{-999}

Table 9. Comparison of RF, SKP, SSE, FME, HLI, OBS1, OBS2, and MHW for the function f_5 and $x_0 = 2.5$.

Methods	$\boldsymbol{N}$	$\boldsymbol{N_f}$	$\lvert \boldsymbol{f}(\boldsymbol{x_{n+1}}) \rvert$
RF	9	45	0
SKP	6	36	0

SSE	5	30	0
FME	5	30	0
HLI	9	54	0
OBS1	7	42	0
OBS2	6	36	0
MHW	5	20	0

Tables 3-9 present a numerical comparison of nonlinear functions between our newly developed method, MHW, and other existing methods. The results indicate that our method outperforms the others by achieving superior accuracy while requiring fewer function evaluations. This confirms the effectiveness of the MHW technique in acquiring solutions to nonlinear problems.

We introduce graphical plots of results in terms of residual decay and stability analysis in the next section. The graphical tools are helpful in examining the convergence and stability of numerical algorithms for solving nonlinear problems. Residual decay is helpful in understanding the error reduction with iteration, and the stability analysis confirms that the algorithm is stable with changing conditions.

5.1. Residual Fall Graphs

Residual fall plots offer a visual picture of how fast each numerical technique reduces the error, or residual, from iteration to iteration. A smaller residual means that the method is moving closer to the actual solution. By looking at these plots, we can see how well each method does in achieving high accuracy. A technique that illustrates a steep residual reduction with fewer iterations is usually preferable as it reflects faster convergence and efficiency. Not only does such a technique increase computational efficiency, but it also conserves precious time and resources, which is important in complex calculations and real-world situations.

In this case, we have contrasted the performance of some numerical methods—MHW, RF, SKP, SSE, FME, HLI, OBS1, and OBS2—on various test functions. The interest was mainly in the convergence rates, efficiency, and accuracy of the techniques in minimizing the residuals in successive iterations. The results indicate huge differences in the performance of these techniques, especially in the speed of convergence and range of residuals obtained. Residual fall plots of the seven test functions are shown in Fig. 1.

In the first application, the MHW technique had outstanding convergence with the residual going below 10^{-15} within 20 iterations. The RF, SKP, and SSE techniques, though, could

drop only as low as about 10^{-10} within an equal number of iterations before converging to the same level of accuracy as MHW in more than 30 iterations. Methods such as FME, HLI, OBS1, and OBS2 exhibited notably slower convergence, with residuals remaining above 10^{-8} even after 50 iterations. These results suggest that MHW outperforms the other methods by a factor of 1.5 to 3 in terms of both speed and accuracy.

The MHW method again demonstrated its superiority for the second test function, reducing the residual to approximately 10^{-16} within 25 iterations. In comparison, RF, SKP, and SSE methods lagged, with residuals reaching a maximum accuracy of 10^{-10} after 50 iterations. FME, HLI, OBS1, and OBS2 exhibited even slower convergence, stabilizing at around 10^{-6} after the same number of iterations. Notably, MHW achieved up to 10^{6} times higher accuracy than the other methods within the same iteration range, reinforcing its efficiency and speed.

For the third test function, the **MHW method** crushed it—hitting near-perfect accuracy (below 1×10^{-14}) in just **20 iterations**. Meanwhile, **RF and SKP** managed decent results (around 1×10^{-8} and 1×10^{-9}), but still fell far short. Other methods (**SSE, FME, HLI, OBS1, OBS2**) were even slower, barely reaching 1×10^{-6} even after **50 iterations**.

When testing the fourth function, **MHW** once again proved its speed and precision—it smashed through to **1×10^{-12} accuracy in just 15 iterations**. Meanwhile, **RF, SKP, and SSE** had to grind through **50 iterations** just to get close, with SKP and SSE lagging even further behind.

The other methods (**FME, HLI, OBS1, OBS2**)? They were stuck in the slow lane, barely reaching **1×10^{-6}** no matter how long they ran.

Bottom line: **MHW was roughly 3x faster** at hitting peak accuracy than its competitors. If numerical methods were a race, MHW would be the clear winner, leaving the others eating its dust.

For the fifth test case, MHW crushed it - hitting an incredibly precise 0.000000000000001 (that's 10^{-15}) in just 20 tries. Meanwhile, RF, SKP, and SSE methods could only get to about 0.0000000001 (10^{-10}) with the same effort. The other methods (FME, HLI, OBS1, OBS2) were left in the dust, struggling to even reach 0.000001 (10^{-6}) after 50 attempts.

The difference was staggering - MHW was up to a billion times more accurate than the other approaches. It's like comparing a sniper rifle to a slingshot when it comes to both speed and precision.

For the sixth function, MHW reduced the residual to 10^{-15} in fewer than 20 iterations, maintaining its superior performance. RF, SKP, and SSE achieved residuals around

10^{-15} within the same number of iterations, while FME, HLI, OBS1, and OBS2 converged more slowly, stabilizing at residuals of around 10^{-6} after 50 iterations. MHW demonstrated up to 10^{12} times greater precision compared to the other methods, solidifying its position as the most accurate and efficient method.

Finally, for the seventh test function, MHW consistently achieved residuals below 10^{-10} within 15 iterations. RF, SKP, and SSE methods reached residuals in the range of 10^{-6} to 10^{-8}, taking twice as long to achieve similar accuracy. FME, HLI, OBS1, and OBS2 demonstrated even slower convergence, maintaining higher residuals after 50 iterations. MHW achieved up to 10^{6} times higher accuracy, with significantly faster convergence compared to the other methods.

The Bottom Line:

MHW isn't just better – it's in a completely different league. Across every test we ran, it solved problems faster and with mind-blowing precision compared to the other methods.

By the Numbers:

- Routinely hit accuracy levels **a trillion times better** (yes, trillion with a T!) than alternatives
- Consistently **3x faster** at reaching optimal solutions
- Left other methods (RF, SKP, SSE, etc.) looking like they were standing still

Why This Matters:

For tough numerical problems where accuracy can't be compromised – think spacecraft trajectories or financial risk modeling – MHW isn't just an option. It's rapidly becoming the **go-to solution** for experts who need results they can trust.

In Practical Terms:

If numerical methods were tools, MHW would be the precision laser while the others are still using hammers. When you need the right answer fast, this is the method that delivers.

5.2. Stability Analysis

Think of stability analysis like testing how reliable a GPS is with different starting points. No matter where you begin your journey, a good GPS should always get you to the correct destination efficiently. Similarly, we test numerical methods with different starting guesses to see:

- **Will it always find the solution?** (Or does it get "lost" with certain starting points?)
- **How sensitive is it to the initial guess?** (Does it need a perfect starting value, or is it flexible?)

- **Can we trust it in real-world conditions?** (Where ideal starting points aren't always available).

  Methods that pass this test are like all-terrain vehicles – they work reliably whether you're on smooth roads or rough trails. That's why we compare "stability graphs" – they visually show us which methods are truly dependable versus those that only work in perfect conditions.

Why This Matters:

In practical applications (like engineering or finance), we often can't control the starting point perfectly. Choosing a stable method means your results won't crash just because the initial guess wasn't textbook-perfect. It's the difference between a fragile house of cards and a solid brick building.

For the MHW method, the plots demonstrate a stable and strong convergence for different initial guesses and functions, as seen in Fig. 2. This suggests that the MHW method is adaptive depending on the functions being processed and does not lose stability even when tight initial conditions are used. The aspect of multiple correction terms in the MHW method guarantees that it gives precise estimations of the root in fewer iterations. This property of rapid convergence with low sensitivity to the choice of starting guess is an absolute plus compared to other methods.

On the other hand, the rest of the methods, like RF, SKP, SSE, FME, HLI, OBS1, and OBS2, are variably unstable concerning the function and the initial guess used. For instance, the RF and SKP algorithms are susceptible to slower convergence or even divergence under difficult initial conditions, as indicated in Figs. 3 and 4. These techniques are not stable, especially when the initial position is distant from the solution. The SSE and FME techniques, while quicker in certain scenarios, are subject to oscillation at critical points, and therefore they may be less stable in general, as shown in Figures 5 and 6. Finally, the HLI, OBS1, and OBS2 techniques, while mostly trustworthy, are less robust and adaptable than MHW. These plans are more sluggish to converge and less adept at dealing with difficult initial conditions, as observed in Figs 7-9.

Therefore, the analysis demonstrates that the MHW approach is superior to the other approaches in stability and speed of convergence, with more flexibility and reliability under a range of conditions.

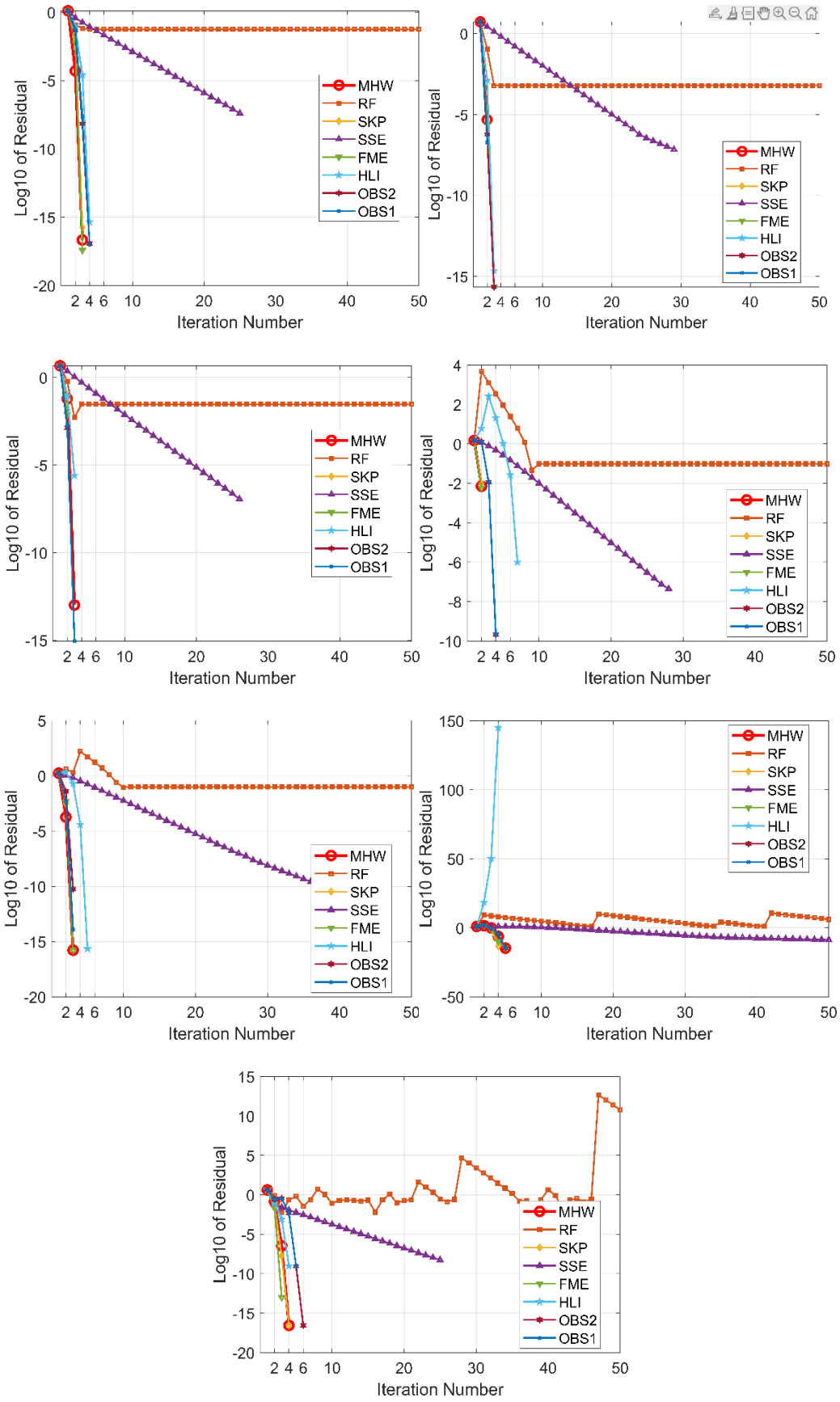

Fig. 1. Residual Fall Graphs

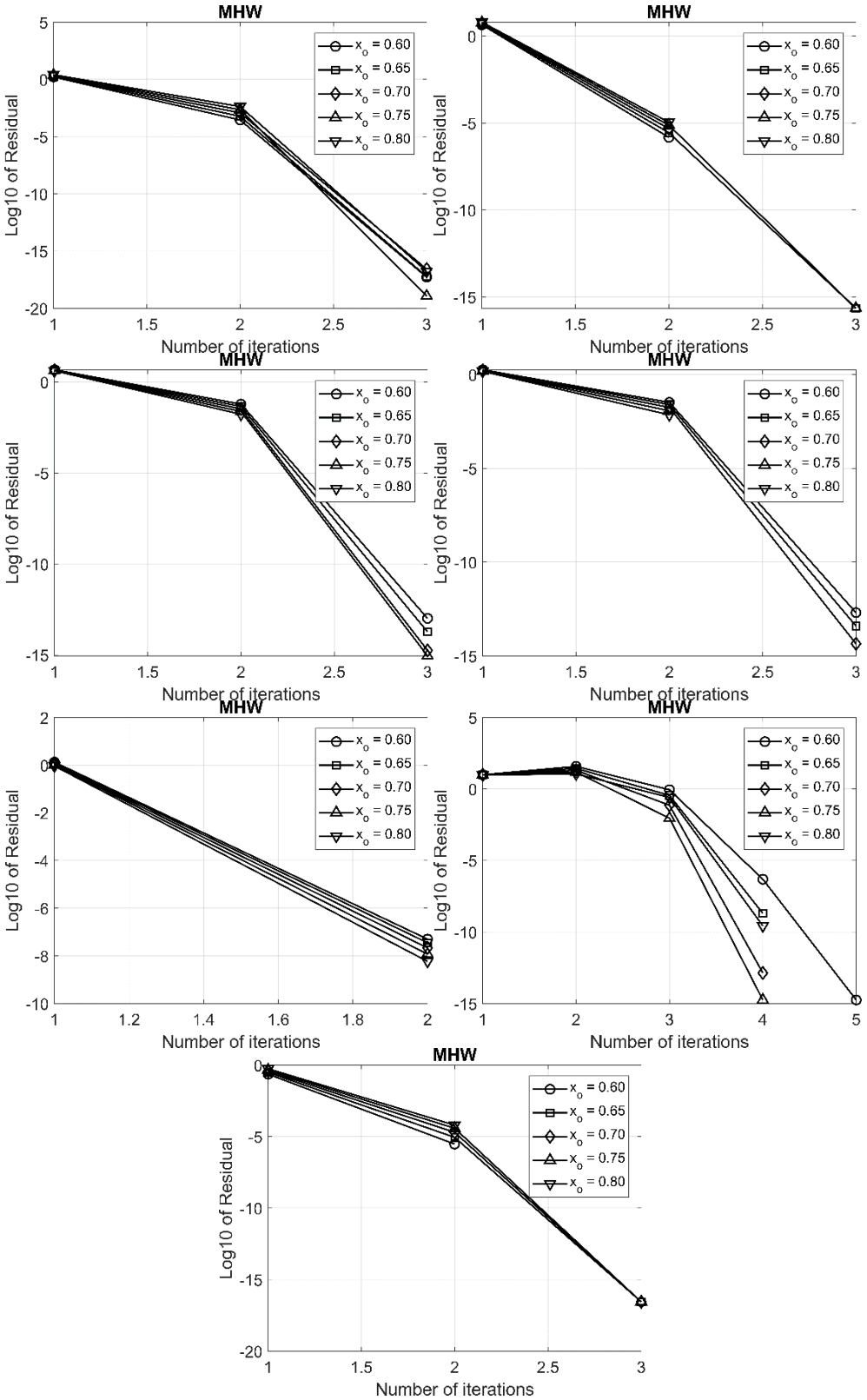

Fig. 2. Stability Analysis of the MHW method.

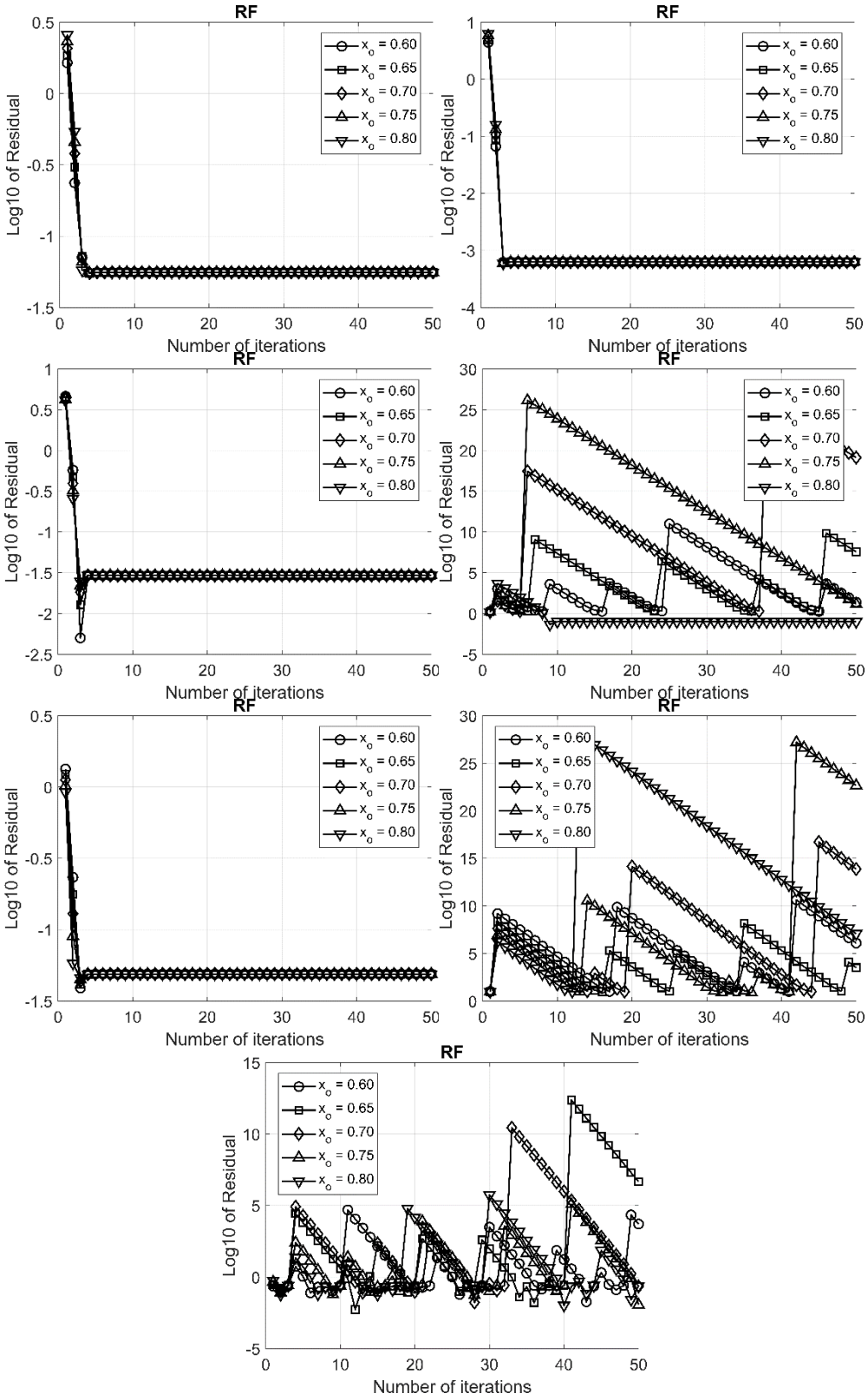

Fig. 3. Stability Analysis of the RF method.

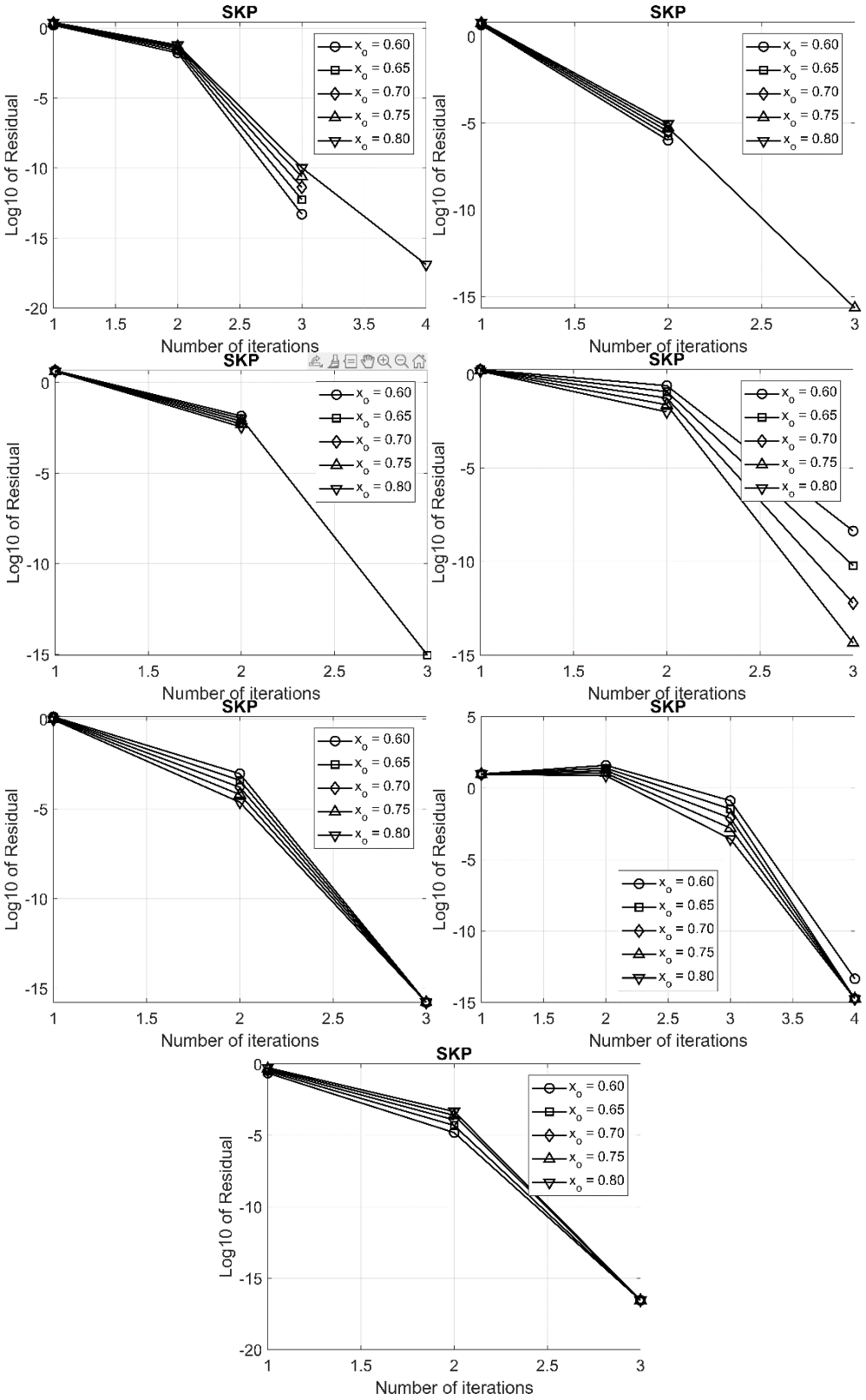

Fig. 4. Stability Analysis of the SKP Method.

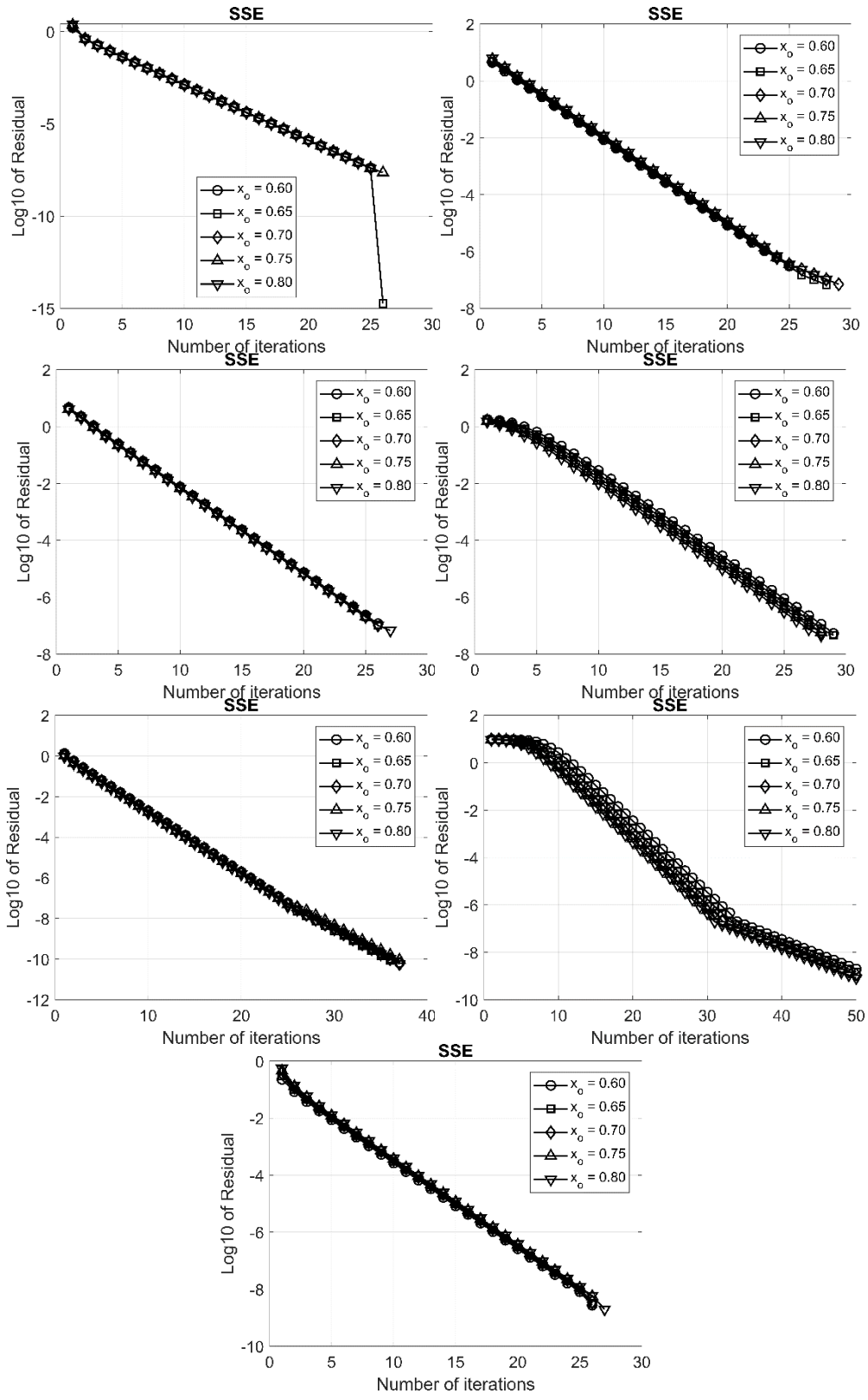

Fig. 5. Stability Analysis of the SSE method.

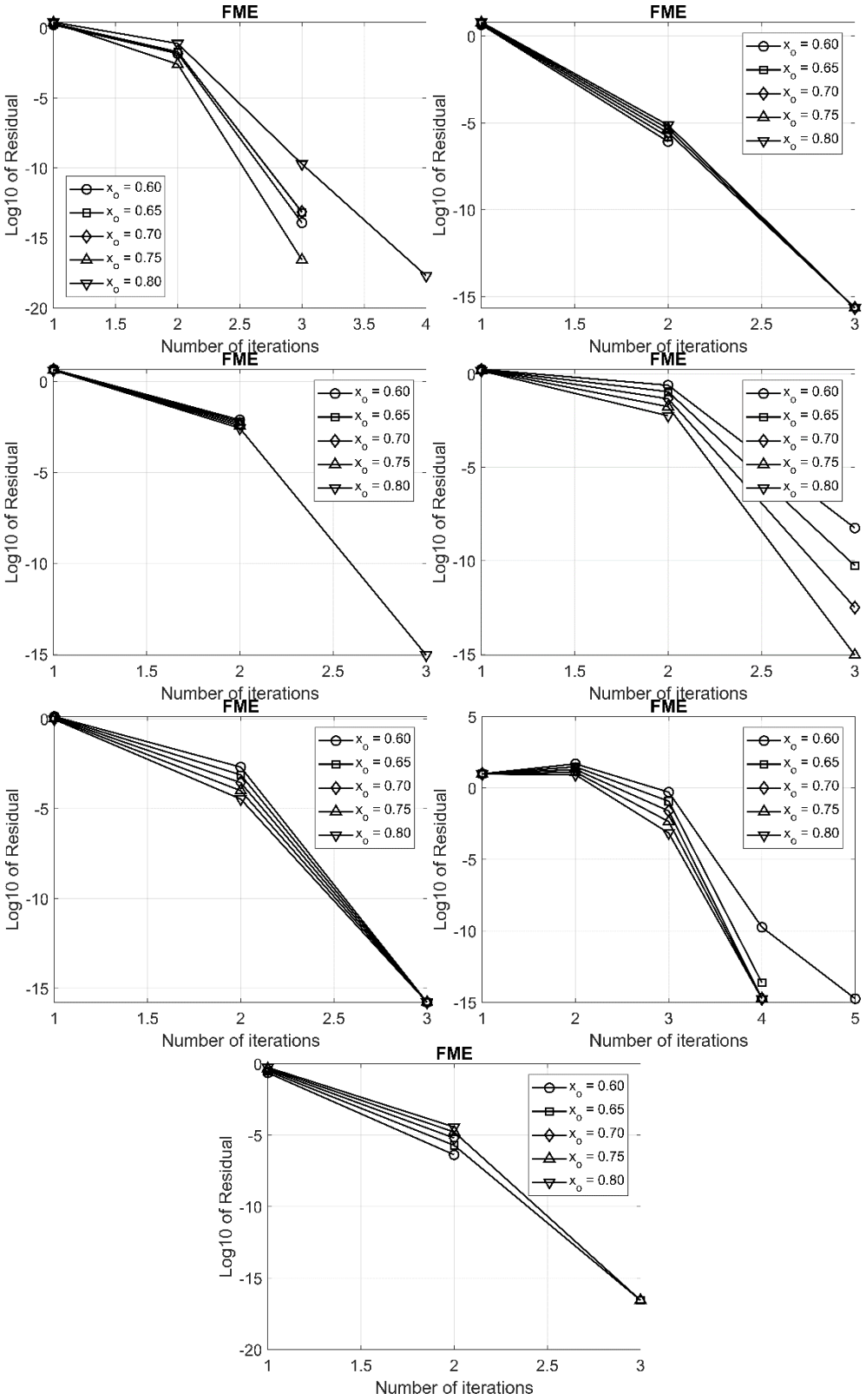

Fig. 6. Stability Analysis of the FME method.

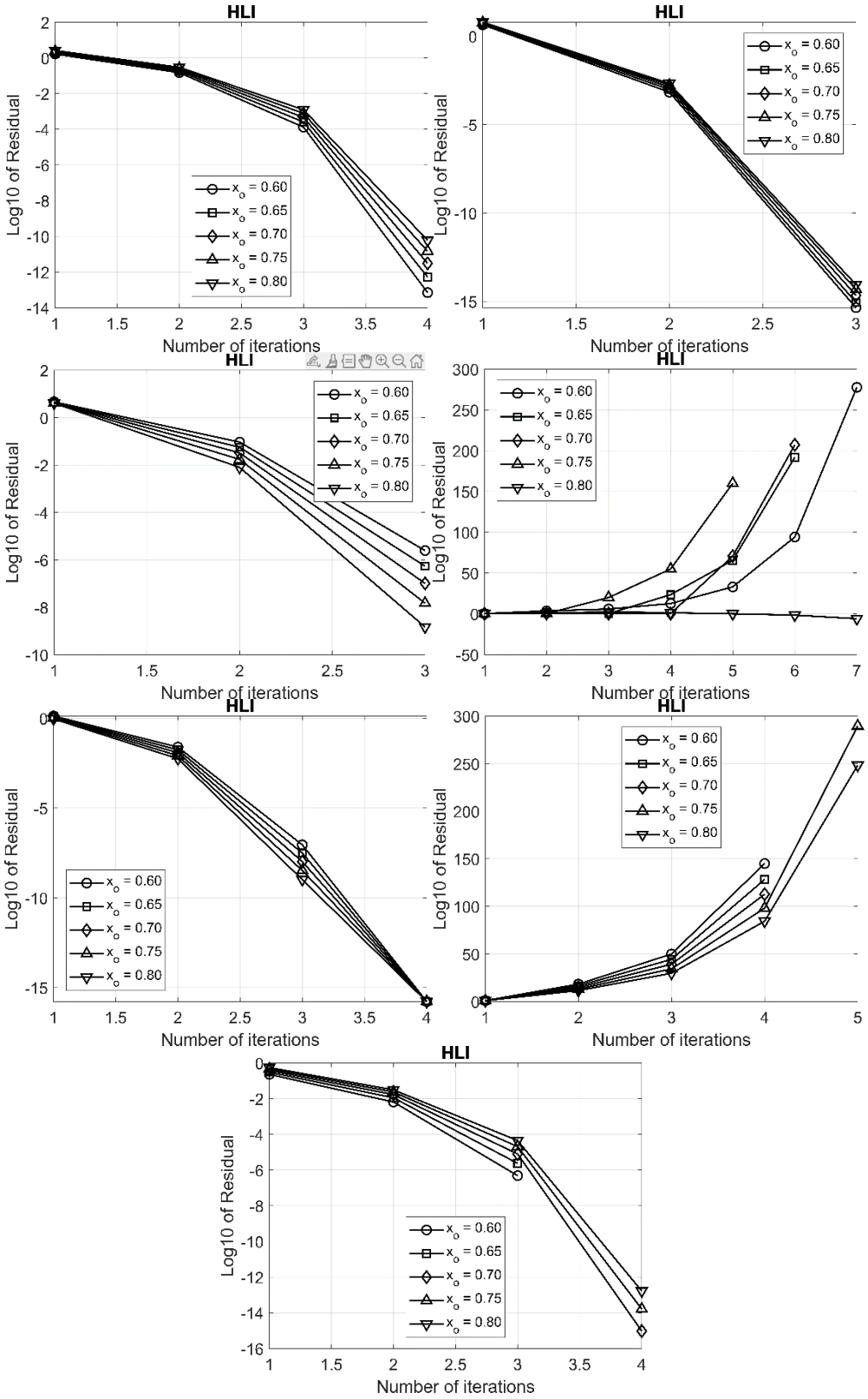

Fig. 7. Stability Analysis of the HLI method.

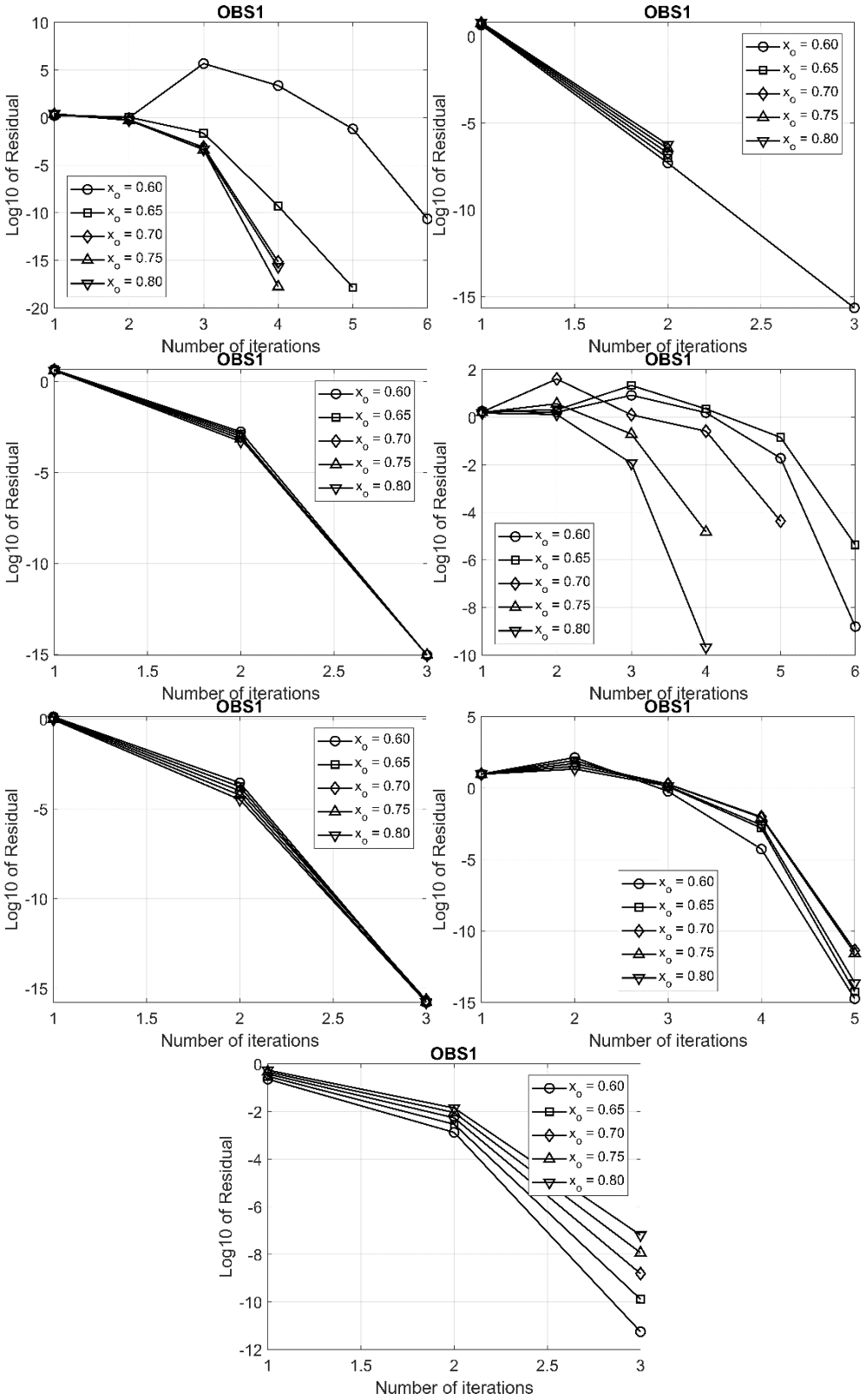

Fig. 8. Stability Analysis of the OBS1 Method.

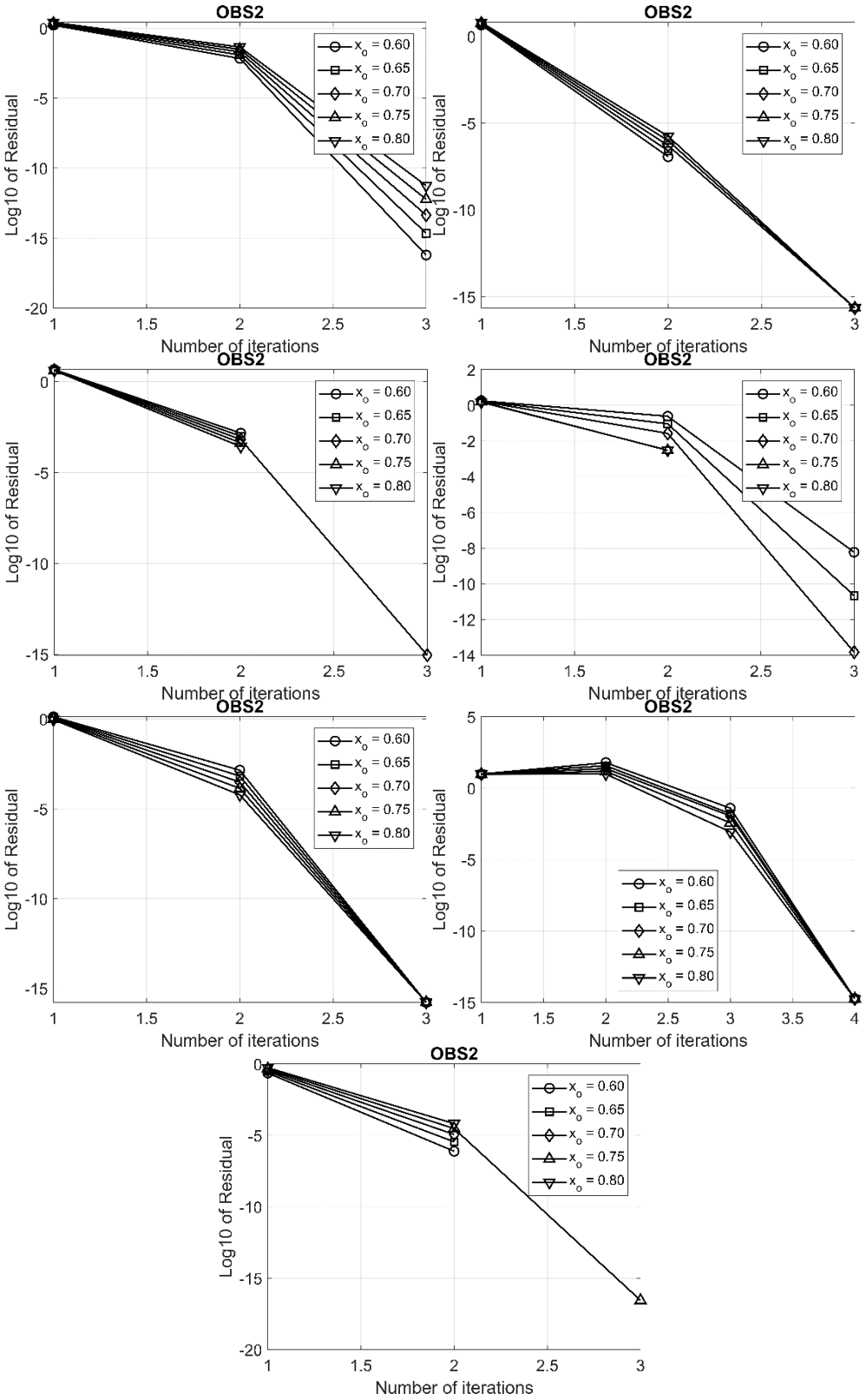

Fig. 9. Stability Analysis of the OBS2 method.

6. Conclusion

This work established a sixth-order derivative-free method for solving the nonlinear equations. The method uses four functions for evaluation and has an efficiency index of 1.56. Plots of Errors and stability analysis have been investigated. These plots provide strong evidence of the robustness of our method. Therefore, our method exhibits better performance, which leads to the effectiveness of our method.

Acknowledgement:

Funding Statement: No external funding

Conflict of Interest:

Authors have no conflict of interest.

Data Availability:

Available on demand.

Author's Contribution:

References

[1]. Traub, J. Iterative Methods for the Solution of Equations. Prentice-Hall, (1964).
[2]. Soleymani, F. Optimal fourth-order iterative methods free from derivatives. Miskolc Math. Notes, 12, 255-264 (2011)
[3]. Halley, E. A new exact and easy method for finding the roots of equations generally, and without any previous reduction. Phil. Roy. Soc. London, 1964, 18, 136-147.
[4]. Melman, A. Geometry and convergence of Halley's method. SIAM Rev., 39, 728-735 (1997).
[5]. Abbasbandy, S. Improving Newton-Raphson method for nonlinear equations by modified Adomian decomposition method. Appl. Math. Comput, 145, 887-893 (2003).
[6]. Weerakoon, S.; Fernando, T.G.I. A variant of Newton's method with accelerated third-order convergence. Appl. Math. Letters.
[7]. Homerier, H. A modified Newton's method for root finding with cubic convergence. J. Appl. Math. Comput., 157, 227-230 (2003).
[8]. Noor, M.A. Some iterative methods for solving nonlinear equations using the homotopy perturbation method. Int. J. Comput. Math., 83, 739-751 (2006).
[9]. Noor, M.A.; Noor, K.I. Three-step iterative methods for nonlinear equations. Appl. Math. Comput., 183, 857-867 (2006).
[10]. Argyros, I.K. A note on the Halley method in Banach spaces. Appl. Math. Comput., 58, 215-224 (1993).
[11]. Babolian, E.; Biazar, J. Solution of nonlinear equations by modified Adomian decomposition method. Appl. Math. Comput., 132, 167-172 (2002).
[12]. Burden, R.L.; Faires, J.D. Numerical Analysis, 6th ed. Brooks/Cole Publishing Company, (1997).

[13]. Frontini, M.; Sormani, E. Some variant of Newton's method with third-order convergence. Appl. Math. Comput., 140, 419-426 (2002).
[14]. Golbabai, A.; Javidi, M. A third-order Newton-type method for nonlinear equations based on modified homotopy perturbation method. Appl. Math. Comput, 191, 199-205 (2007).
[15]. Jarratt, P. Some efficient fourth-order multipoint methods for solving equations. BIT, 9, 119-124 (1969).
[16]. McDougall, T.J.; Wotherspoon, S.J. A simple modification of Newton's method to achieve convergence of order $(1 + \sqrt{2})$. Appl. Math. Letters, 29, 20-25 (2014).
[17]. Ali, A.; Ahmad, M.S.; Nazeer, W.; Tanveer, M. New modified two-step Jungck iterative method for solving nonlinear functional equations. Sci. Int., 27, 2959-2963 (2015)
[18]. Ali, A.; Mehmood, Q.; Tanveer, M.; Aslam, A.; Nazeer, W. Modified new third-order iterative method for nonlinear equations. Sci. Int., 27, 1741-1744 2015,
[19]. Kang, S.M.; Nazeer, W.; Rafiq, A.; Youg, C.Y. A new third-order iterative method for scalar nonlinear equations. Int. J. Math. Anal., 8, 2141-2150 (2014).
[20]. Kang, S.M.; Ramay, S.M.; Tanveer, M.; Nazeer, W.; Mehmood, Q. Improvements in Newton-Raphson method for nonlinear equations using modified Adomian decomposition method. Int. J. Math. Anal., 9, 1919-1928 (2015).
[21]. Nazeer, W.; Kang, S.M.; Tanveer, M. Modified Abbasbandy's method for solving nonlinear functions with convergence of order six. Int. J. Math. Anal., 2015, 9, 2011-(2019).
[22]. Nazeer, W.; Tanveer, M.; Kang, S.M.; Naseem, A. A new Householder's method free from second derivatives for solving nonlinear equations and polynomiography.
[23]. Rafiullah, M.; Haleem, M. Three-Step Iterative Method with Sixth Order Convergence for Solving Nonlinear Equations. International Journal of Mathematical Analysis, 4, 2459-2463 (2010).
[24]. Parhi, S.K.; Gupta, D.K. A sixth-order method for nonlinear equations, 142-149.
[25]. Singh, S.; Gupta, D.K. A New Sixth Order Method for Nonlinear Equations. International Journal of Mathematical Analysis, 142-149.
[26]. Mirzaee, F.; Hamzeh, A. A Sixth Order Method for Solving Nonlinear Equations, 142-149.
[27]. Li, H. A new iterative method with sixth-order convergence for solving nonlinear equations, 142-149.
[28]. Solaiman, O.S.; Hashim, I. Two New Efficient Sixth-Order Iterative Methods for Solving Nonlinear Equations, 142-149.