Mathematical Biosciences and Engineering

MBE, 19(8): 8215–8258.
DOI: 10.3934/mbe.2022383
Received: 14 May 2022
Revised: 25 May 2022
Accepted: 26 May 2022
Published: 06 June 2022

http://www.aimspress.com/journal/MBE

Research article

Review of chaotic mapping enabled nature-inspired algorithms

Zheng-Ming Gao[1,*], Juan Zhao[2] and Yu-Jun Zhang[2]

[1] School of computer engineering, Jingchu university of technology, Jingmen 448000, China
[2] School of electronics and information engineering, Jingchu university of technology, Jingmen 448000, China

* **Correspondence:** Email: gaozming@jcut.edu.cn; Tel: +867242355622; Fax: +867242355622.

Abstract: Chaotic maps were frequently introduced to generate random numbers and used to replace the pseudo-random numbers distributed in Gauss distribution in computer engineering. These improvements in optimization were called the chaotic improved optimization algorithm, most of them were reported better in literature. In this paper, we collected 19 classical maps which could all generate pseudo-random numbers in an interval between 0 and 1. Four types of chaotic improvement to original optimization algorithms were summarized and simulation experiments were carried out. The classical grey wolf optimization (GWO) and sine cosine (SC) algorithms were involved in these experiments. The final simulation results confirmed an uncertainty about the performance of improvements applied in different algorithms, different types of improvements, or benchmark functions. However, Results confirmed that Bernoulli map might be a better choice for most time. The code related to this paper is shared with https://gitee.com/lvqing323/chaotic-mapping.

Keywords: chaotic maps; nature-inspired algorithms; benchmark functions; simulation experiments; chaotic improvements

1. Introduction

Inspired by the living existence that evolved from their ancestors lived millions of years ago, scientists and engineers have proposed a lot of nature-inspired algorithms [1], however, most of the algorithms including their improvements were not able to find all the solutions of the existed problems due to the No Free Lunch (NFL) rule [2]. And to verify the capability of algorithms, we also formulate many functions to test them, most of them nowadays become classical in literature and so we called

them benchmark functions [3]. After thorough study of verification experiments, the algorithms were soon applied to solve the real-world engineering problems, for instance, prediction of routing scheme for softwarized vehicular networks [4], adaptive routing for edge software-defined vehicular networks [5], satellite-terrestrial networks storage strategy planning [6]. However, we soon confirmed that there still did not exist an efficient algorithm that could solve all of the problems we met. Therefore, new algorithms even their improvements were remaining under demand.

Literal reviews claimed that the nature-inspired algorithms could be classified into four types based on the source of the inspiration: evolutionary algorithms, swarm-based, human-based, and physics-based ones [7]. The exact number of the proposed nature-inspired algorithms might be not known causing its current prosperousness in artificial intelligence [8], however, the improvement of a given algorithm might be also difficult to tell in number. It could be relevant to all aspects of an existed algorithm regarding the improvements. For instance, the particle swarm optimization (PSO) algorithm, which was proposed in 1995 [9], became popular soon after its birth and various kinds of improvements were raised by scientists and engineers all over the world. In the traditional PSO algorithm, the individuals update their positions according to their velocities, random weighted distance between them and their best trajectories, and the global best candidates in each iteration.

In a detailed study of the improved PSO algorithms, first of all, the continuous domain definition was considered to be not able to meet the request of problems in the digital world. And consequently, the discrete binary version of the PSO algorithm was proposed [10]. The weight of the velocity was soon added and better performance was astonishingly achieved [11]. Inertia weights were then evaluated to remove the constant characteristics of this weight parameter [12]. Some improvements reconsider the involvement of the three parts and decreased them linearly to increase the convergence ratio, and therefore, the neighborhood operator improved PSO algorithm [13] was proposed.

Except for the parameters involved in the original algorithm, the participation of individuals in swarms might be another interesting hotspot. A fixed number of successful or failed operations in optimization might be introduced to guarantee convergence [14]. While in the fully informed PSO algorithm [15], all of the individuals were introduced to play roles in the updating of their positions, discarding the original concept of performance of the velocity, historical best, and global best candidate. Eliminating the historical trajectory with $c_1 = 0$ [16] or replacing the global best candidates with the worst ones in each iteration [17] might be other tries.

Furthermore, the velocities or positions of individuals in swarms might also be reinitialized during iterations [18]. The multiple updating disciplinary [19] might also be introduced. For instance, in the bare-boned PSO algorithm [20], half of the individuals in swarms were updated according to the global best candidates in iteration and another half of them with traditional way. The probability might be defined by the number of tunnels for individuals to follow, such as in the heterogeneous PSO [21].

Let alone the hybridization of famous algorithms [22], the randomness involved in the algorithms was another hot spot in the design of improvements. Traditional pseudo-random numbers are equally selected from the given domain, therefore, the distribution of weights was Gauss distribution, the random steps were equally chosen from the distribution. However, the population number in swarms was not expected to be large in number under the control of current computer hardware. And few samples could not follow the statistics of Gauss distribution. It might decrease the capability in exploration and exploitation [23]. Considering the Levy flight [24] could generate some long steps after several rounds of small steps, it could be used to increase the performance of algorithms in optimization. The overall better performance demonstrated in the literature proved that such kind of

improvement could indeed improve the capability of the original algorithm [25]. Another common improvement was to replace the randomness in Gauss distribution with chaos. Because of the pseudo randomness generated by Gauss distribution in computer engineering, the random numbers were believed not well with smaller population size. However, the population size could not be set with larger numbers under the constraints of our real-world computer hardware. Considering the chaos are derived from non-linear dynamic systems, they would perform more chaotically in generating numbers. In fact, almost all of the existed algorithms have been improved with chaos, as shown in Table 1.

Table 1. History of nature-inspired algorithms and their chaotic improvements.

Original [26]				Chaotic improvements		
Name	Authors	Year	Classification	Chaotic map	Authors	Year
GA	JH Holland	1992	Evolutionary	Logistic [27]	Debolina G	2021
HS	ZW Geem	2001	Evolutionary	Multiple [28]	B Alatas	2010
ABC	D Karaboga	2007	Swarm-based	Logistic [29]	L Ding	2015
FA	XS YANG	2008	Swarm-based	Multiple [30]	AH Gandomi	2013
GSA	E Rashedi	2009	Physics-based	Multiple [31]	JH Jiang	2020
BA	XS YANG	2010	Swarm-based	Multiple [32]	AH Gandomi	2014 [33]
SMO	JC Bansal	2014	Swarm-based			
GWO	S Mirjalili	2014	Swarm-based	Multiple [34]	A Farshin	2017
PPA	SL Tilahun	2015	Swarm-based	Logistic [35]	W Zhu	2014
WOA	S Mirjalili	2016	Swarm-based	Multiple [36]	G Kaur	2018
GSO	V Muthiah	2016	Swarm-based			
CSA	A Askarzadeh	2016	Swarm-based	Multiple [37]	Rizk M	2018
IPO	MH Mozaffari	2016	Physics-based			
ABO	X Qi	2017	Physics-based			
FFA	H Shayanfar	2018	Swarm-based	Multiple [38]	M Mitic	2015
QSA	J Zhang	2018	Human-based			
SSFO	HY Sang	2019	Swarm-based			
AOA	L Abualigah	2020	Physics-based			
JS	JS Chou	2021	Swarm-based			
ALSO	N Kumar	2021	Swarm-based			

Therefore, we focused our attention and collected most of the chaotic mappings in literature and tried to verify its better performance. The main contribution of this paper would be:

1) A collection of 19 popular chaotic mappings and all of them could generated random numbers within an interval of 0 and 1.

2) A brief review of the chaotic improvements and four types of improvements were summarized.

3) Applying in improving the well-known grey wolf optimization (GWO) and sine cosine (SC) algorithms, detailed comparison and simulation experiments were carried out.

The rest of this paper would be arranged as follows: in Section 2, all of the chaos we found which could generate chaos in [0, 1] would be listed. Literature review of the chaotic improvements would be reviewed in Section 3. Some chaotic improvements and the experiments would be shown in Section 4. Discussions would be made and conclusions would be drawn in Section 5.

2. The chaos

The chaos is a kind of deterministic random-like method, most of them are derived from non-linear dynamic systems [39]. Chaos is also a kind of pseudo randomness, however, since they are derived from the chaotic system, most of them would be bounded, irregular and sensitive to the initial conditions [40]. Meanwhile, the values would spread all around the domain with a given route from the initial values. The chaos could be used to eliminate the uncertainty of pseudo randomness and therefore, better stability could be wanted.

To replace the pseudo randomness in computer engineering, the chaos should be in an interval between 0 and 1. Consequently, some transformation would be wanted to change the original values if the original chaos were limited in other domains except [0, 1], the following chaotic maps could all generate chaos satisfying the requirements with or without some extra functions.

2.1. CM1: Bernoulli map

Bernoulli map is a famous map to generate chaos, its formulation might be shown as follows [41]:

$$x_{n+1} = \begin{cases} 1.75x_n - 0.5 & x_n > 0 \\ 1.75x_n + 0.5 & x_n \leq 0 \end{cases} \tag{1}$$

Where x_n represents the current chaos with $n-$th iteration and x_{n+1} represents the chaos at the next iteration.

To generate pseudo randomness in the interval between 0 and 1, chaos generated by the Bernoulli map might be transformed with double and absolute functions. A brief view of the values versus five hundred iterations would be shown in Figure 1.

Note that there are five hundred of iterations for the Bernoulli map. For convincement, all of the following chaotic maps would be also iterated five hundred times for comparison.

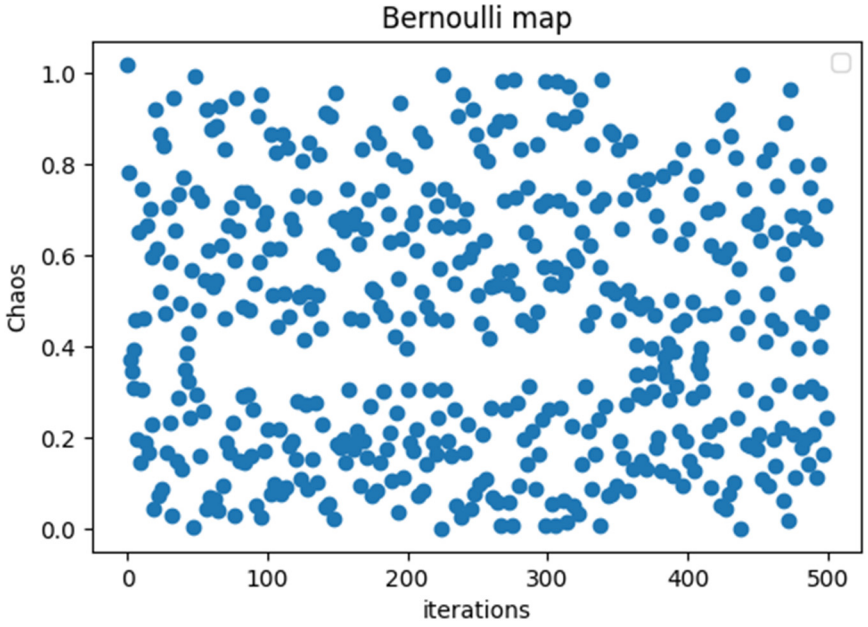

Figure 1. Chaos generated by Bernoulli map.

2.2. CM2: Chebyshev map

Chaos generated by Chebyshev map could be obtained from an equation formulated as follows [32]:

$$x_{n+1} = \cos\left(\frac{n}{cosx_n}\right) \tag{2}$$

Chaos generated by the Chebyshev map fluctuates in the interval between -1 and 1. Therefore, to replace the pseudo randomness in computer engineering, the final chaos should be transformed with absolute function, as shown in Figure 2.

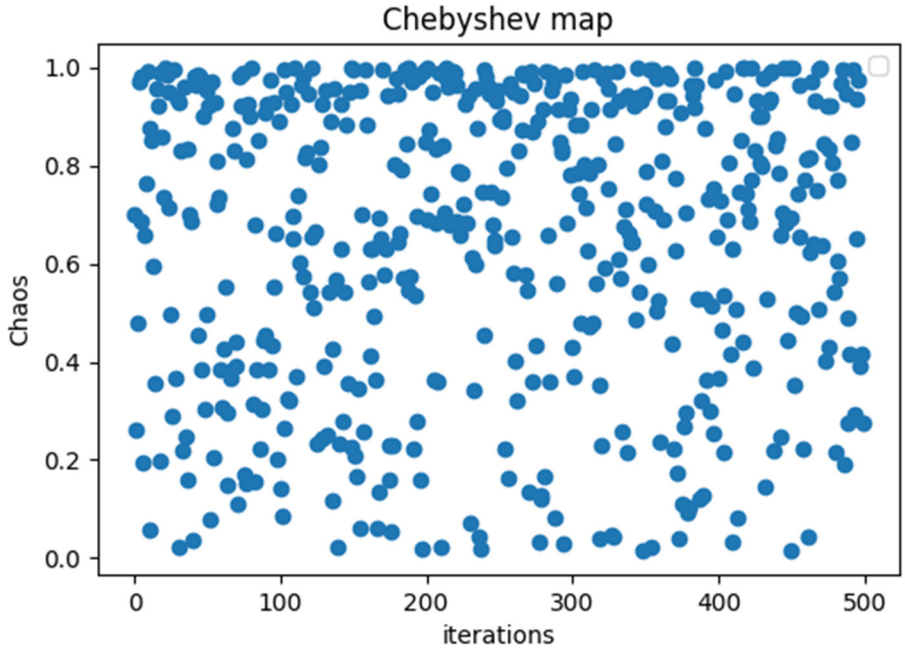

Figure 2. Chaos generated by Chebyshev map.

2.3. CM3: Circle map

Circle map [32] is formulated with the sine function:

$$x_{n+1} = \left[x_n + b - \frac{a}{2\pi}\sin(2\pi x_n)\right] mod(1) \tag{3}$$

Where mod (1) is the mathematical function to find the remainder number after dividing by 1. If a = 0.5 and b = 2, the Circle map would generate chaos in the interval of 0 and 1, as shown in Figure 3.

2.4. CM4: Gauss map

Gauss map is also called Mouse map [32]:

$$x_{n+1} = \begin{cases} 0 & x_n = 0 \\ \frac{1}{x_n} - \left\lfloor \frac{1}{x_n} \right\rfloor & otherwise \end{cases} \tag{4}$$

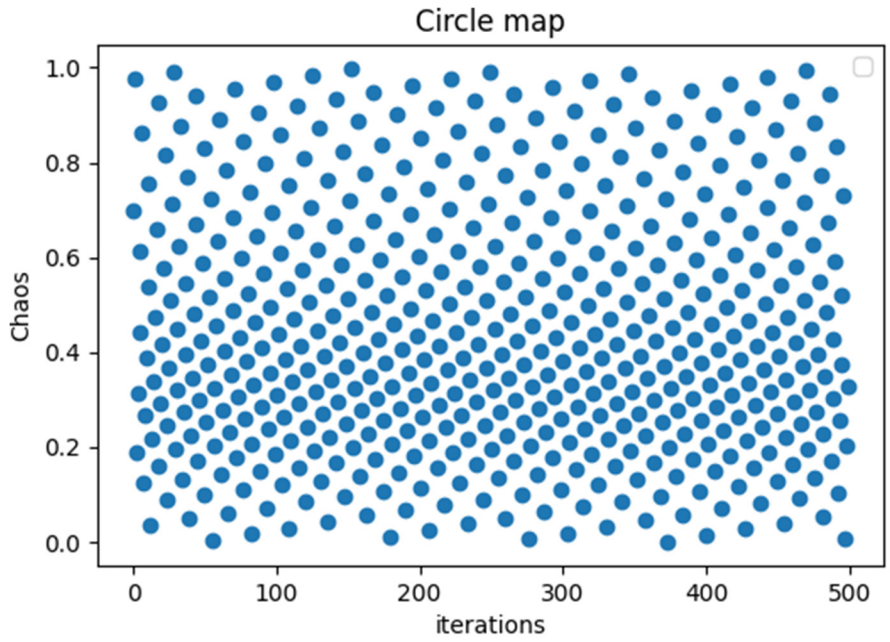

Figure 3. Chaos generated by Circle map.

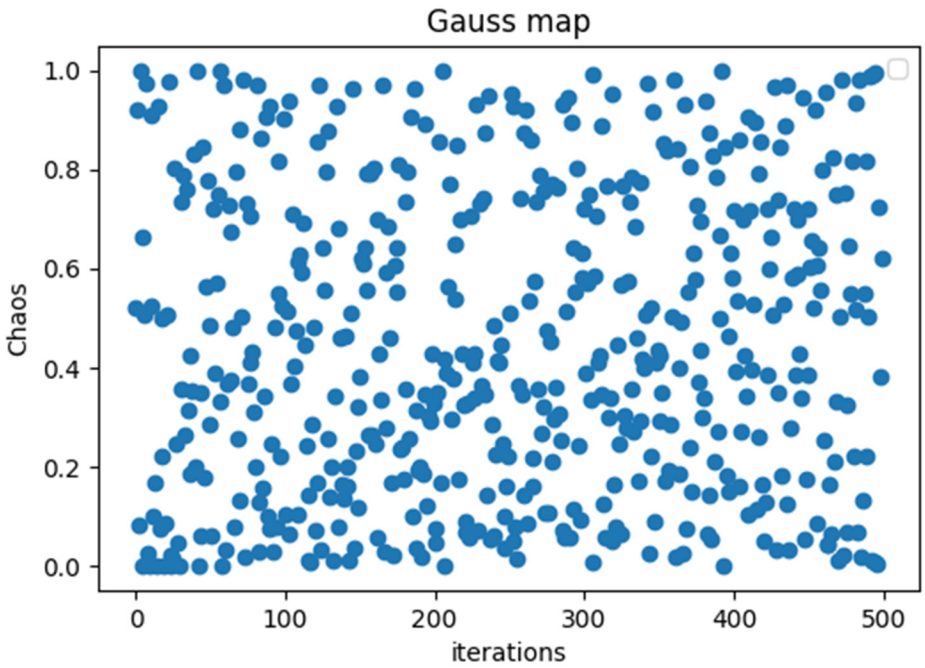

Figure 4. Chaos generated by Gauss map.

Where $\lfloor x \rfloor = x - x\ mod\ (1)$ represents the integer part of the float number. Gauss map directly generates chaos in [0, 1] domain, seen from Figure 4.

2.5. CM5: Hybrid map

The hybrid map is a new kind of chaos generator [42], the Hybrid system is formulated as follows:

$$x_{n+1} = \begin{cases} b(1 - \mu_1 x_n^2 & -1 < x_n \leq 0 \\ 1 - \mu_2 x_n & 0 < x_n < 1 \end{cases} \tag{5}$$

Where μ_1 = 1.8, μ_2 = 2.0, b = 0.85. The chaos generated by the Hybrid map falls within -1 and 1. Therefore, the absolute function should be introduced to constraint the chaos to fall in the [0, 1] domain, as shown in Figure 5.

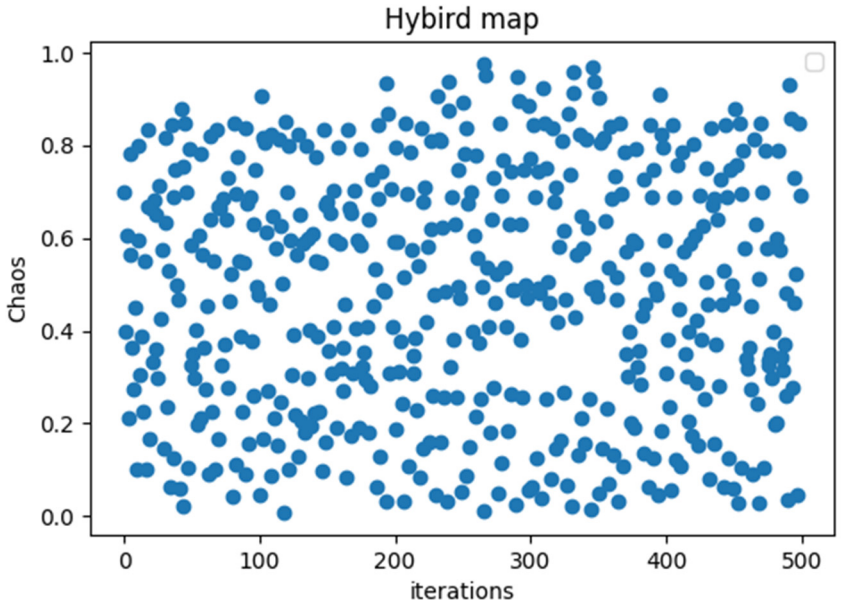

Figure 5. Chaos generated by Hybrid map.

2.6. CM6 ICMIC map

The iterative chaotic map with infinite collapses (ICMIC) map is a one-dimensional chaotic map [43], it is relevant to the sine function and formulated as follows:

$$x_{n+1} = \sin(\frac{a}{x_n}) \tag{6}$$

Due to the definition of the sine function, the ICMIC map also generates chaos fallen in [-1, 1], the absolute function is also introduced to transform the chaos to fall in [0, 1], seen from Figure 6.

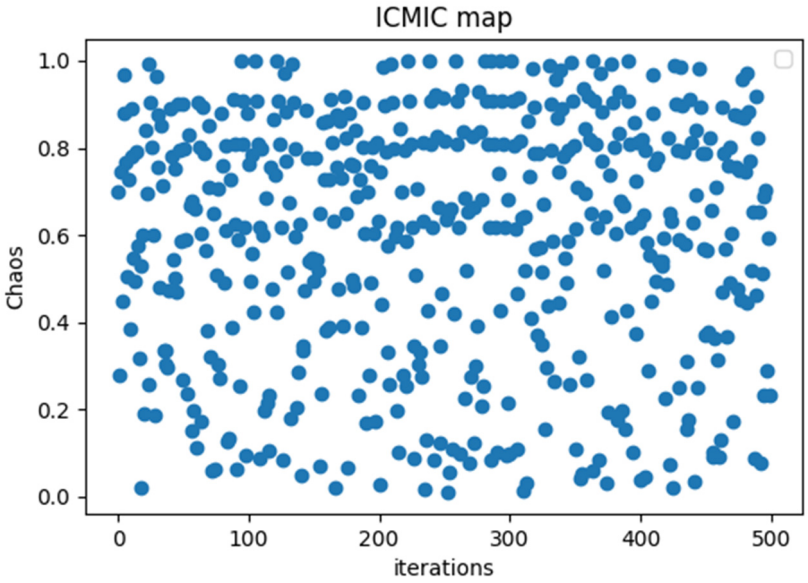

Figure 6. Chaos generated by ICMIC map.

2.7. CM7 Intermittency map

Intermittency map is an extension of the Bernoulli shift in which the piecewise linear map representing the passive interval is replaced by the following equation [32,39]:

$$x_{n+1} = \begin{cases} \varepsilon + x_n + cx_n^m & 0 < x_n \leq d \\ \frac{x_n - d}{1-d} & d < x_n < 1 \end{cases} \tag{7a}$$

Where ε is a small number and c is calculated as follows:

$$c = \frac{1-\varepsilon-d}{d^m} \tag{7b}$$

Intermittent map generates chaos whose values are in [0, 1], yet spread in a weird distribution, seen from Figure 7.

2.8. CM8 Iterative map

The iterative map is similar to the ICMIC map, as shown in the following equation:

$$x_{n+1} = \sin(\frac{a\pi}{x_n}) \tag{8}$$

The iterative map also generates chaos in [-1, 1], so it should be transformed by the absolute function. The distribution of chaos could be seen in Figure 8.

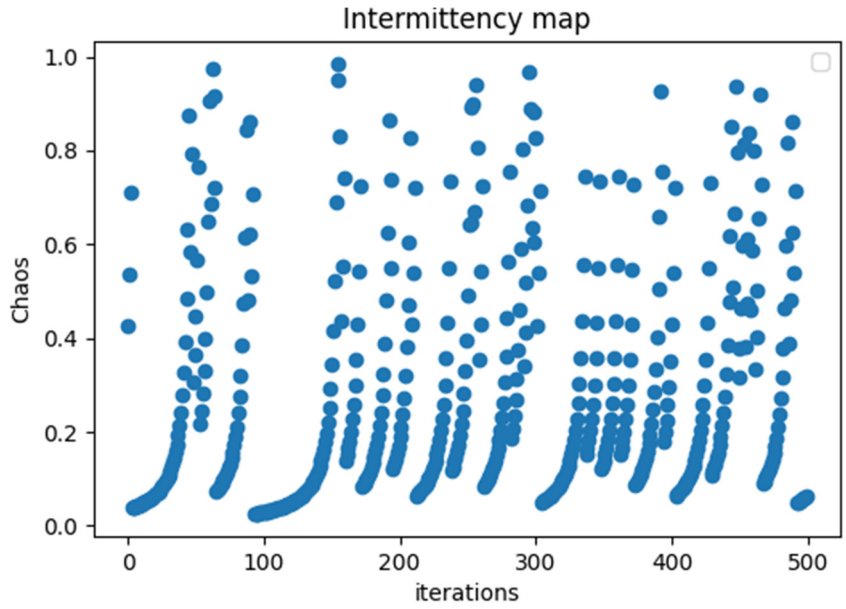

Figure 7. Chaos generated by Intermittency map.

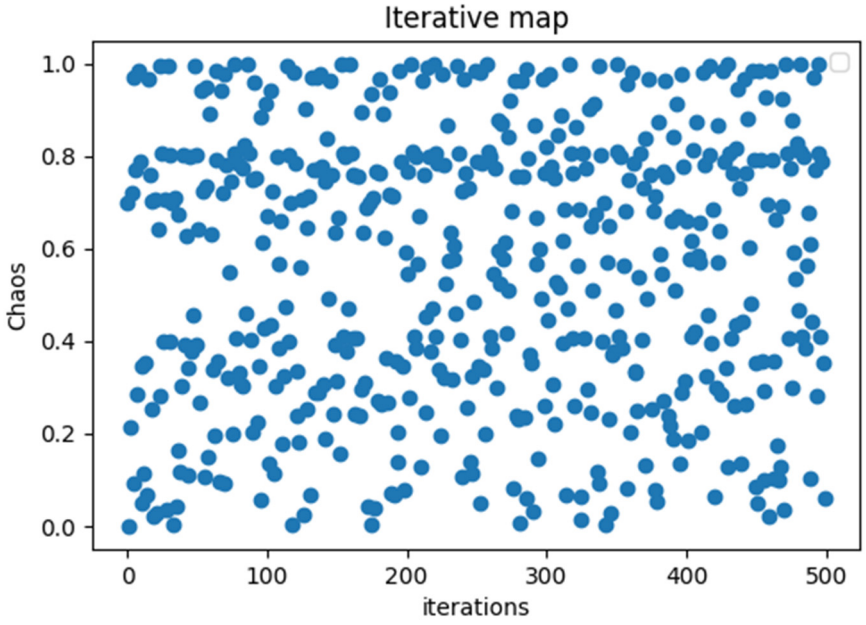

Figure 8. Chaos generated by Iterative map.

2.9. CM9 Liebovitch map

Liebovitch map was unique, it has two separators in the domain (0, 1) [32], as formulated as follows:

$$x_{n+1} = \begin{cases} \alpha x_n & 0 < x_n \le P_1 \\ \frac{P - x_n}{P_2 - P_1} & P1 < x_n \le P_2 \\ 1 - \beta(1 - x_n) & P_2 < x_n \le 1 \end{cases} \tag{9a}$$

Where, $\alpha < \beta$ and defined as follows:

$$\alpha = \frac{P_2}{P_1}[1 - (P_2 - P_1)] \tag{9b}$$

$$\beta = \frac{1}{P_2 - 1}[(P_2 - 1) - P_1(P_2 - P_1)] \tag{9c}$$

Liebovitch map would generate chaos mostly distributed in (0, 0.2) and (0.8, 1) domains, seen from Figure 9.

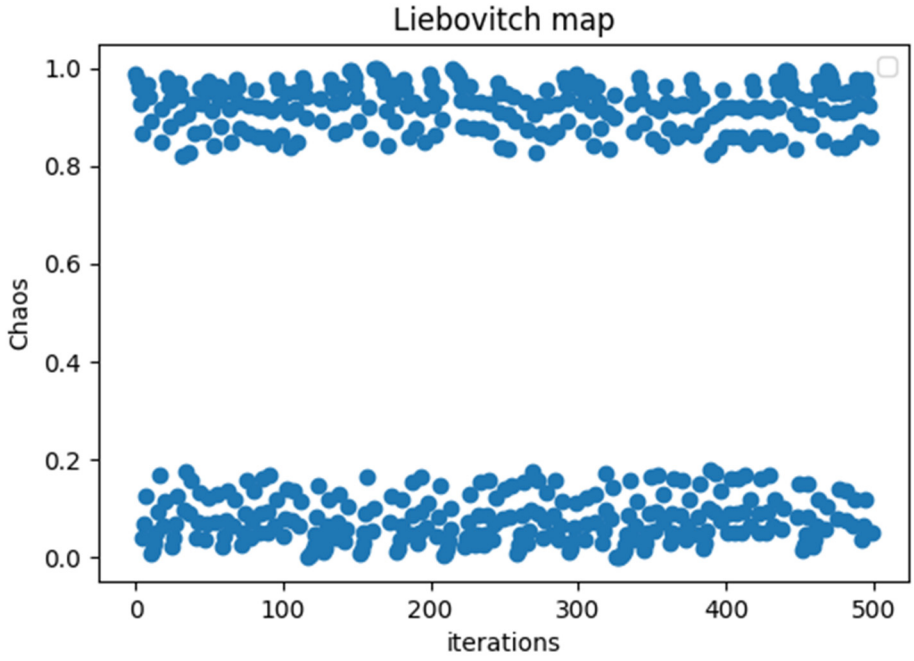

Figure 9. Chaos generated by Liebovitch map.

2.10. CM10 Logistic map

The logistic map is a classic chaotic map derived from nonlinear dynamics of biological population evidencing chaotic behavior [32]:

$$x_{n+1} = a\, x_n(1 - x_n) \tag{10}$$

Initially, $x_0 \neq \{0, 0.25, 0.5, 0.75, 1\}$. Logistic map directly generates chaos whose values are in [0, 1] domain, seen from Figure 10.

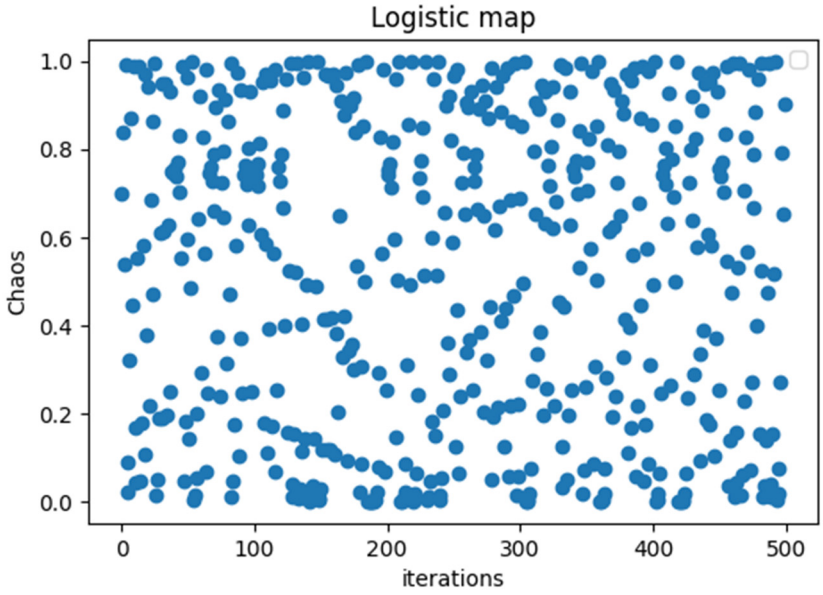

Figure 10. Chaos generated by the Logic map.

2.11. CM11 Kent map

Kent map is a famous map used to generate pseudo-random numbers [44], it is defined as follows:

$$x_{n+1} = \begin{cases} \frac{x_n}{m} & 0 < x_n \le m \\ \frac{1-x_n}{1-m} & m < x_n < 1 \end{cases} \tag{11}$$

Kent map generates chaos in (0, 1), seen from Figure 11.

2.12. CM12 Piecewise map

Piecewise map consists of four linear pieces and is formulated as follows:

$$x_{n+1} = \begin{cases} \frac{x_n}{d} & 0 \le x_n < d \\ \frac{x_n - d}{0.5 - d} & d \le x_n < \frac{1}{2} \\ \frac{1-d-x_n}{0.5-d} & \frac{1}{2} \le x_n < 1-d \\ \frac{1-x_n}{d} & 1-d \le x_n < 1 \end{cases} \tag{12}$$

Obviously, $0 < \mathrm{d} < 0.5$, and $x_n \in [0, 1)$, it could also be derived from Figure 12.

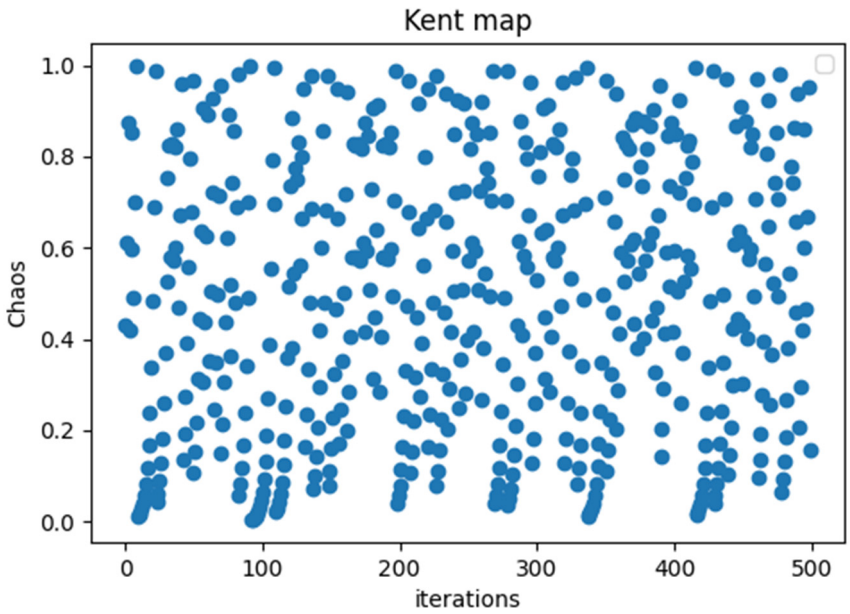

Figure 11. Chaos generated by Kent map.

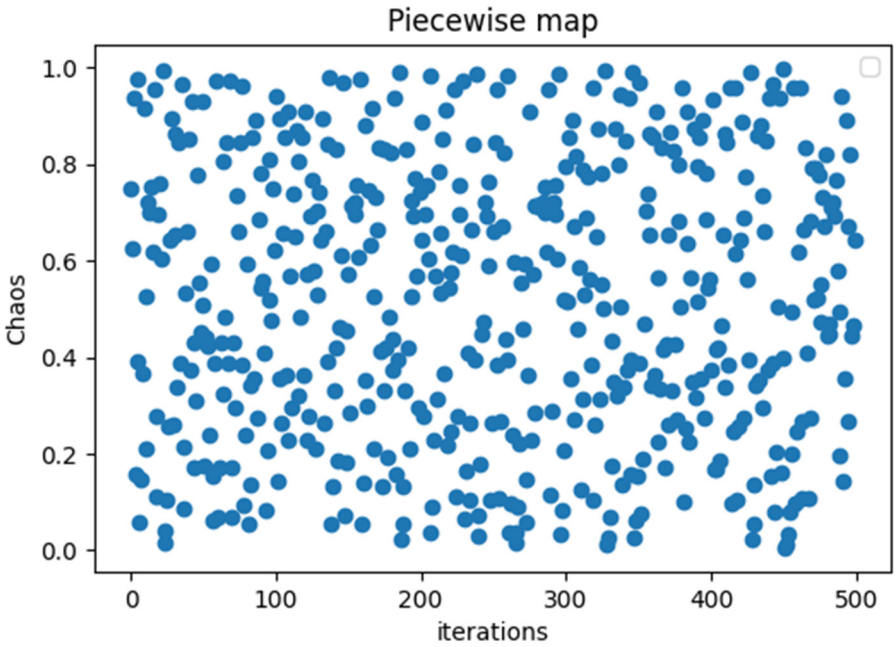

Figure 12. Chaos generated by Piecewise map.

2.13. CM13 Piecewise Linear map

Piecewise Linear map also consists of two piecewise linear segments [40], it is defined as follows:

$$x_{n+1} = \begin{cases} \frac{x_n}{1-\lambda} & 0 < x_n \leq 1-\lambda \\ \frac{x_n-(1-\lambda)}{\lambda} & (d \equiv 1-\lambda) < x_n < 1 \end{cases} \tag{13}$$

There are two linear segments in the Piecewise Linear map, which generate chaos in (0, 1), as shown in Figure 13.

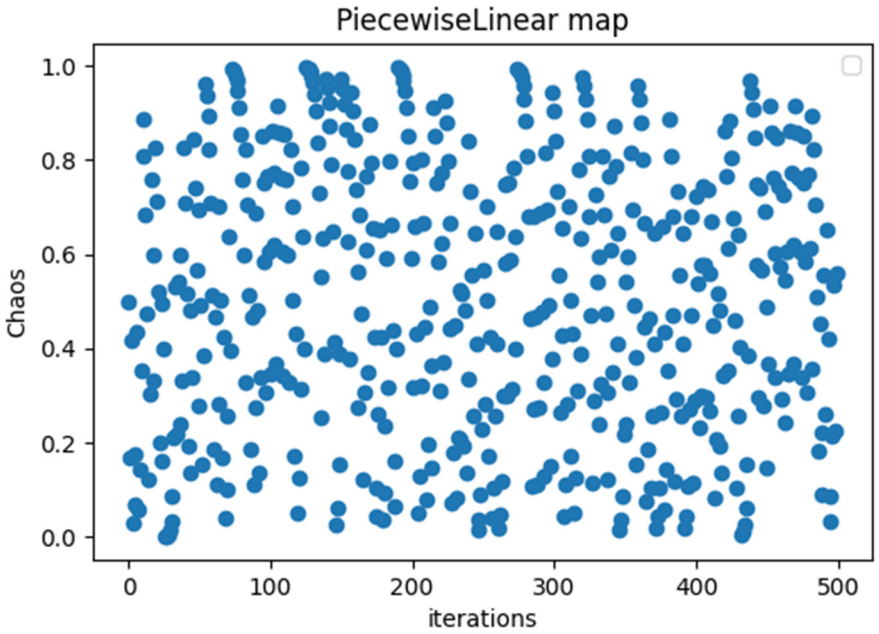

Figure 13. Chaos generated by Piecewise Linear map.

2.14. CM14 Quadratic map

The quadratic map is decoded from shared codes [41], it is a new kind of Mandelbrot map [45] and formulated as follows:

$$x_{n+1} = b - ax_n^2 \tag{14}$$

Where $a = 4,\ b = 0.5$. Quadratic map generates chaos fallen in [-0.5, 0.5]. To replace the pseudo-random numbers, double and absolute functions would be introduced, the final chaos values could be seen in Figure 14.

2.15. CM15 Sine map

Sine map is another iteration form with sine function [32], its formulation is:

$$x_{n+1} = a \cdot \sin(\pi x_n) \tag{15}$$

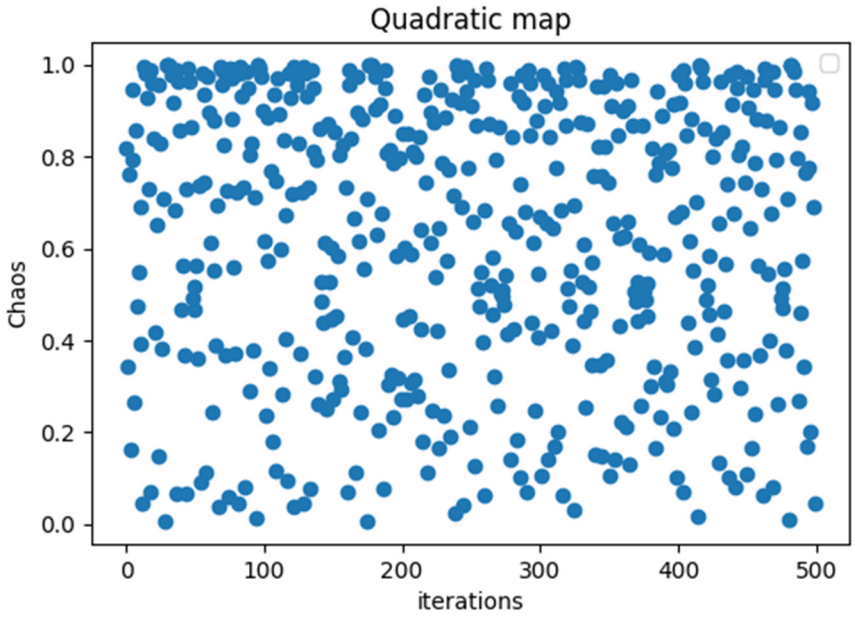

Figure 14. Chaos generated by Quadratic map.

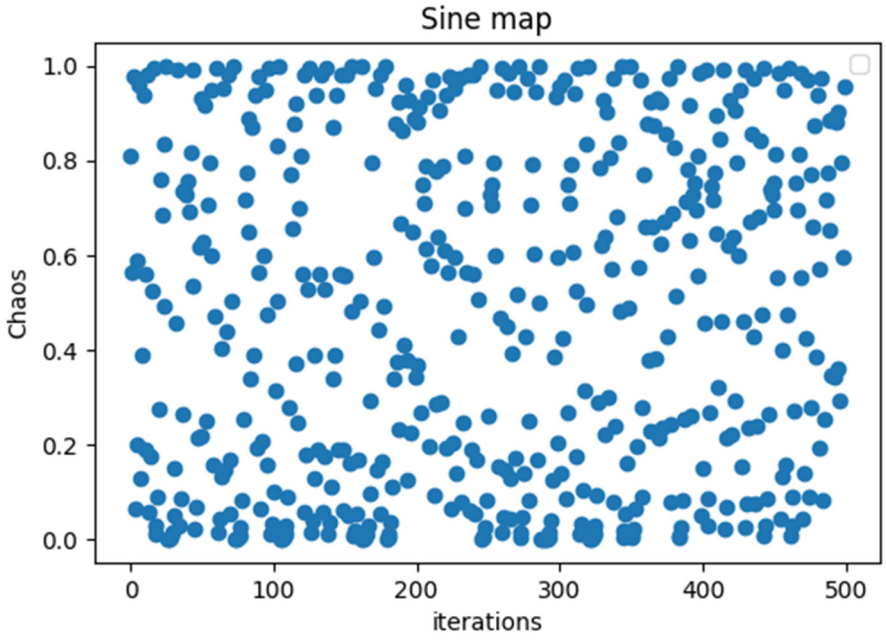

Figure 15. Chaos generated by Sine map.

To be simple, $a = 1$, and we can obtain the chaos as shown in Figure 15.

2.16. CM16 Singer map

Singer map is a one-dimensional system defined as follows [32]:

$$x_{n+1} = \mu(7.86x_n - 23.31x_n^2 + 28.75x_n^3 - 13.3x_n^4) \tag{16}$$

Where $0.9 < \mu < 1.08$. The Singer map generates chaos whose values could be directly treated as pseudo-random numbers, seen from Figure 16.

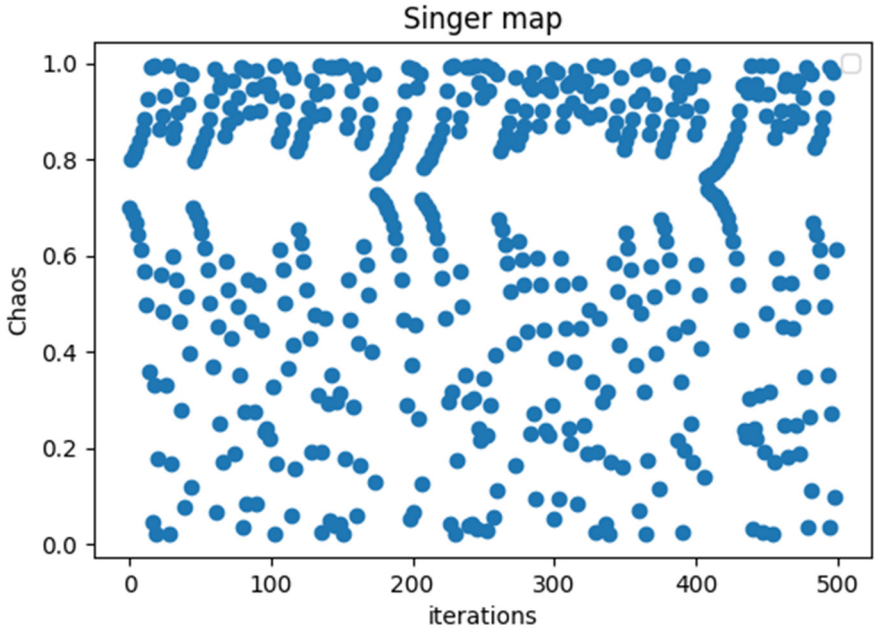

Figure 16. Chaos generated by Singer map.

2.17. CM17 Sinusoidal map

Sinusoidal map [32] is a combination of Sine map and Quadratic map, as shown in the follows equation:

$$x_{n+1} = a \cdot x_n^2 \cdot \sin(\pi x_n) \tag{17}$$

The sinusoidal map also directly generates pseudo-random numbers, as shown in Figure 17.

2.18. CM18-19 Tent map

There appeared two kinds of Tent maps in literature. One of them is similar to the well-known Logistic map and is defined by the following equation [32]:

$$x_{n+1} = \begin{cases} \frac{x_n}{0.7} & x_n < 0.7 \\ \frac{10}{3}(1 - x_n) & x_n \geq 0.7 \end{cases} \tag{18}$$

We called this Tent1 map, labeled CM18 for future applications, it could also directly be used to replace the pseudo-random numbers, as shown in Figure 18.

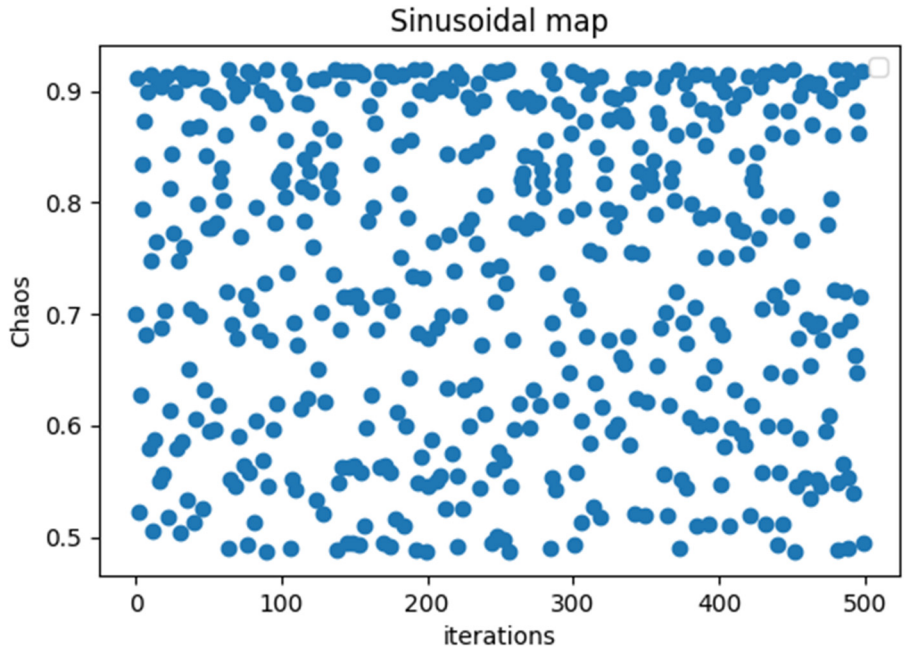

Figure 17. Chaos generated by Sinusoidal map.

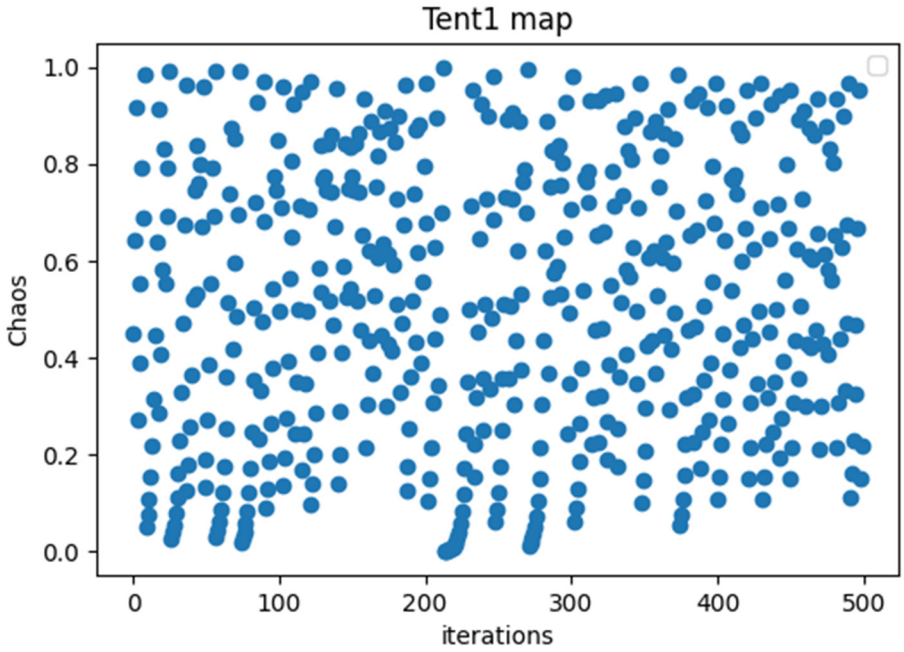

Figure 18. Chaos generated by Tent 1 map.

Another kind of Tent map is introduced by the following iterated function [44]:

$$x_{n+1} = \begin{cases} \mu x_n & x_n < \frac{1}{2} \\ \mu(1 - x_n) & x_n \geq \frac{1}{2} \end{cases} \tag{19}$$

When $\mu = 2 - \varepsilon$, this Tent map could generate chaos, we called this map Tent2 map, labeled

CM19 for future purpose, its values could be seen from Figure 19.

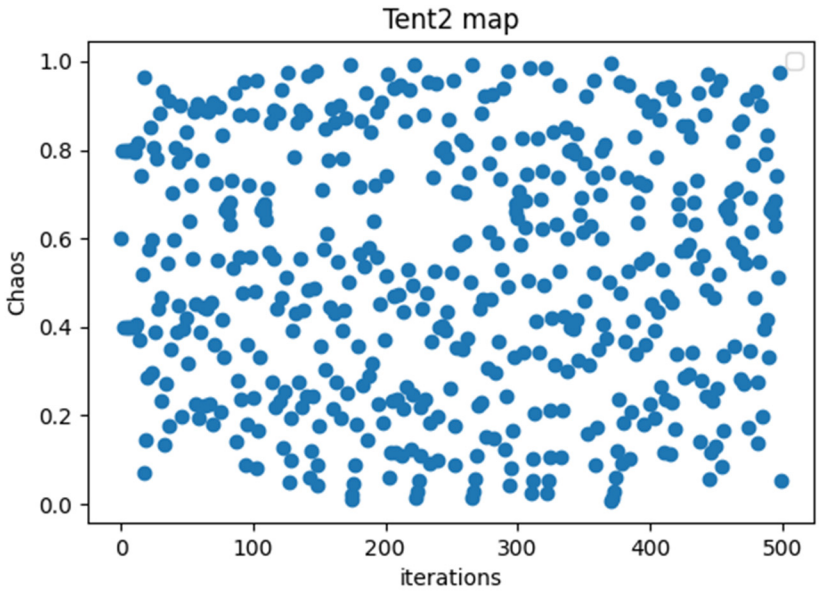

Figure 19. Chaos generated by Tent 2 map.

3. A brief review of chaotic improvements in literature

The chaotic map has been introduced to improve nature-inspired algorithms for many years. Scientists and engineers all around the world were still trying to improve the existed algorithms with chaos.

3.1. Ways of chaotic applications in improvements

The existed algorithms have been tried to improve their capability in every aspect we could find.

First of all, since all of the individuals in swarms must be initialized at the beginning, the positions of individuals should be spread all over the domain of the given problem, they should be distributed as equally as possible. Every optimum, either being global or local, should be gained access by some individuals to be found as soon as possible. Therefore, the traditional pseudo-random number might also be not the best choice. This kind of effort has been made, for example, in some kind of chaotic PSO algorithm [46]. However, our early simulation experiments on some benchmark functions optimized by the chaotic Slime mould (SM) algorithm [47] proved that the chaotic initialization might decrease the capability of algorithms in optimization sometimes.

The second kind of chaotic application in improving the existed algorithms is to replace the pseudo-randomness numbers reasonably. Most of the efforts were made on this kind of improvement. Every random number falling within the interval between 0 and 1 might be replaced by chaos. Considering the number of randomnesses, every randomness could be tried to replace by every kind of chaotic map one by one, as jobs executed in the chaotic bat algorithm [32], the chaotic whale optimization (WO) algorithm [36]. Results might also be fluctuated according to the characteristics of different chaotic maps, even the positions of randomness in mathematics.

The third kind of chaotic application to improve the capability of algorithms might be to define a

chaotic domain near the global best candidates in iterations. This is an efficient way to find whether the global best candidate being approaching the optimum or not. For every global best candidate in iterations, the current best position might be used to generate a random domain, several rounds of other chaotic approaches might be carried out to find whether there was another best position or not. This kind of operation increased the time complexity and meanwhile increased the convergence ratio dramatically. Most of our human attention was paid to this kind of improvement, such as the chaotic PSO algorithm [46], chaotic grey wolf optimization (GWO) algorithm [39].

A new kind of chaotic application was proposed most recently. Almost in all of the swarm-based algorithms, there always existed a controlling parameter to balance the ratio of exploration and exploitation for individuals during iterations. For example, chaotic parameter f in chimp optimization algorithm (ChOA) [48], or the control parameter a in the GWO algorithm [2]. All of these kinds of parameters were declined linearly from the maximum to minimum or zero. In 2019, a new kind of non-linear chaotic decreased way was proposed by Saxena et al. [49], the new control parameter a was defined as a combination of the traditional parameter and chaos following the β distribution. However, although simulation experiments verified its capability in optimization, the overall results were not promising as a whole.

3.2. Performance of chaotic applications in improvements

For the above four types of chaotic applications, the first kind of improvements had been already proved to be incapable frequently in literature. While the rest kinds of improvements were not deterministic, sometimes the chaotic enabled algorithms would perform better, sometimes they did not.

Due to the chaotic characteristics of chaos, chaotic mapping enabled improvement has been popular in literature for decades. Almost all of the algorithms have been tried to be improved with chaotic mapping, as shown in Table 2.

Table 2. Algorithms and their chaotic improvements.

Algorithms	Chaotic map	Performance
Particle swarm optimization	Piecewise Linear map [46]	Better
Ant Colony optimization	Logistic map [50]	Better
Artificial bee colony	Logistic map [51]	Better
Salp swarm	Logistic map [52]	Better
Sine cosine algorithm	Logistic map [53]	Better
Cuckoo search	Logistic map [54]	Better
Social spider optimization	Logistic map [55]	Better
Grey wolf optimization	Ten chaotic maps [39]	Some better some worse
Bat algorithm	More maps involved [32]	Some better some worse
Whale optimization	Ten chaotic maps [36]	Some better some worse
Harris hawks optimization	Nine maps [56]	Some better some worse

4. Applications of chaotic improvements

In Section 3, we found that there were four types of chaotic applications to improve the

performance of the existed algorithms. Some of them had already proved to be incapable for most times. Yet some of them were frequently reported to perform better or worse in literature. In Section 2, we collected 19 chaotic maps and all of them could generate chaos in the interval between 0 and 1. Some of them, for example, Logistic and Piecewise maps, were frequently used in applications. Some of them seldom appeared in literature regarding nature-inspired algorithms. In this section, we would carry out some simulation experiments to verify their suitability and capability. For simplicity, the GWO [57] and sine cosine (SC) [58] algorithms were introduced to be representatives of the swarm-based nature-inspired algorithms, and the convergence rate would be the only metrics to be balanced.

4.1. Experimental setup

Simulation experiments would be carried out to verify the capability of algorithms. There are 19 chaotic maps, four types of improvements, and two famous algorithms being involved in this paper.

Due to the randomness being involved in every nature-inspired algorithm, the results would be changed every time we got. To reduce the influence of randomness, Monte Carlo methods were always introduced to average the results and verified the capability of algorithms in optimization. In this experiment, 100 Monte Carlo simulation experiments would be carried out and the final results would be their average.

For simplicity, we improved the GWO and SC algorithms with different kinds of chaotic maps. And all of the benchmark functions in Table 3−6 were introduced to carry on simulation experiments.

Table 3. Unimodal benchmark functions.

No.	Name	Equations	Optimum	
			Values	Location
1)	Ackley 1	$f(x) = -20e^{-0.02\sqrt{\frac{\sum_{i=1}^{d} x_i^2}{d}}} - e^{\frac{\sum_{i=1}^{d}\cos(2\pi x_i)}{d}} + 20 + e$	0	$x_i = 0$
2)	Exponential	$f(x) = 1 - e^{-0.5\sum_{i=1}^{d} x_i^2}$	0	$x_i = 0$
3)	Sargan	$f(x) = \sum_{i=1}^{d} d\left(x_i^2 + 0.4\sum_{j=1, j\neq i}^{d} x_i x_j\right)$	0	$x_i = 0$
4)	Schwefel 2.20	$f(x) = \sum_{i=1}^{d} \lvert x_i \rvert$	0	$x_i = 0$

Table 4. Multimodal benchmark functions.

No.	Name	Equations	Optimum	
			Values	Location
5)	Alpine 1	$f(x) = \sum_{i=1}^{d} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	0	$x_i = 0$

Continued on next page

No.	Name	Equations	Optimum Values	Location
6)	Cosine Mixture	$f(x) = \frac{d}{10} + \sum_{i=1}^{d} x_i^2 - \frac{1}{10}\sum_{i=1}^{d} \cos(5\pi x_i)$	0	$x_i = 0$
7)	Griewank	$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - -\cos\prod \left(\frac{x_i}{\sqrt{i}}\right) + 1$	0	$x_i = 0$
8)	Rastrigin	$f(x) = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	0	$x_i = 0$

Table 5. Unimodal benchmark functions with planes.

No.	Name	Equations	Optimum Values	Location
9)	Chung Reynolds	$f(x) = \left(\sum_{i=1}^{d} x_i^2\right)^2$	0	$x_i = 0$
10)	Csendes	$f(x) = \sum_{i=1}^{d} x_i^6 \left(2 + \sin\frac{1}{x_i}\right)$	0	$x_i = 0$
11)	Holzman 2	$f(x) = \sum_{i=1}^{d} i x_i^4$	0	$x_i = 0$
12)	Schwefel	$f(x)\left(= \sum_{i=1}^{d} x_i^2\right)^{\alpha}$	0	$x_i = 0$

Table 6. Unimodal benchmark functions with Valleys.

No.	Name	Equations	Optimum Values	Location
13)	Bent Cigar	$f(x) = x_1^2 + 10^6 \sum_{i=2}^{d} x_i^2$	0	$x_i = 0$
14)	Quartic	$f(x) = \sum_{i=1}^{d} i x_i^4$	0	$x_i = 0$
15)	Schwefel 1.2	$f(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j\right)^2$	0	$x_i = 0$
16)	Zakharov	$f(x) = \sum_{i=1}^{d} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d} i x_i\right)^4$	0	$x_i = 0$

4.2. Type 1: Chaotic mapped initialization

For the first kind of improvements, results were shown from Figures 20 to 23.

We can find that although different chaotic maps might generate different kinds of chaos when they are introduced to improve the algorithms at the initialization stage, some of the chaos might perform better, some of them not. For Ackley 1 function, almost all of the chaos introduced to improve the algorithms at the initialization stage performed better than the original. For the Alpine 2 function, all of the chaotic mapped SCA performed better than before, however, only Chebyshev, Tent1, Piecewise, Intermittency, Sine, Gauss, Logistic, ICMIC maps could result in better performance than before.

Optimizing the Chung Reynolds function, only a few of the chaotic maps perform worse than before. However, in a different way for SC algorithm (Liebovitch map performed worse) and the GWO algorithm (Logistic, Chebyshev, Kent, and Liebovitch performed worse).

Similar conclusions could be made on the Bent Cigar function. All of the chaotic maps could yield better performance for the SC algorithm, while for the GWO algorithm, the Hybrid, Chebyshev, Tent2 map would perform worse in improvement.

Based on the results above, we cannot conclude that when the chaotic maps were introduced to improve the randomness at the initialization stage, they would perform better or worse. Different chaotic maps and benchmark functions need to be specially reviewed and verified. It was really hard to predict even for an experienced person. Nevertheless, the chaos introduced at the initialization stage would improve the capability of optimization with a large probability.

4.3. Type 2: Chaotic mapped randomness

The pseudo-random numbers are always replaced by chaos to reduce the influence of traditional pseudo-randomness.

However, since there always existed more than one random number involved in the algorithms, every randomness might be involved to be replaced like the work in chaotic bat algorithm [32]. Furthermore, more chaotic maps might be also introduced to replace each random number respectively.

In this paper, only the first random number was replaced by chaos, the rest of the random numbers remained unchanged, results were also averaged from 100 Monte Carlo simulation experiments, and were shown in Figures 24−27.

Convergence ratio curves of different kinds of equations showed that there might be no strict discipline that could be followed in this kind of experiment, the improvements were random for all of the chaos and benchmark functions.

4.4. Type 3: Chaotic mapped iterations

The most popular improvements introduced were to carry on several rounds of iterations with chaos. In this kind of experiment, we might choose extra rounds of iterations as chaoticMaxIter = 10.

During the main iteration carrying out by individuals, their positions were updated, and then the best candidate was found. Extra chaoticMaxIter iterations were carried out with chaos, when a better position was achieved, then this new best candidate would be treated as the global best candidate in the current iteration. Otherwise, if chaoticMaxIter iterations were carried to the end and no better position was found, the current best candidate still took the position and led the individuals to the next

iteration of swarms.

Due to the extra chaotic exploration and exploitation around the best candidate, who might be the global or local optimum, consequent better results were promised. However, the final results as shown in Figures 28−31 didn't fully support this conclusion.

For Sargan and Holzman's 2 functions, most of the chaotic maps would increase the capability of optimization, only a few of them failed to do so. However, 18/19 chaotic maps failed to increase the SC algorithm in optimizing the Griewank function, the dramatically complex landscape and optima might be the reason. And 13/19 chaotic maps failed to increase the SC algorithm in optimization of Schwefel 1.2 function, the valley shape might be a reason.

Statistically speaking, this kind of improvement could indeed increase the algorithms as a whole, but not all of the chaotic maps could increase or decrease the capability of algorithms in optimization. Confirmative conclusions could be drawn based on the first glance of the benchmark functions or chaotic maps, simulation experiments remain indeed to verify whether a given chaotic map could improve a selected algorithm or not.

4.5. Type 4: Chaotic mapped controlled parameters

The controlled parameter balancing the exploration and exploitation ratio was mostly declined from the maximum to minimum values linearly, for instance, the energy parameter E in the Harris hawk optimization (HHO) algorithm [59], a in the SC algorithm or GWO algorithm. Sometimes other distributions might be also introduced, such as the exponential function in the equilibrium optimization (EO) algorithm [60], arctanh [61], or cosine [62] functions in the slime mould (SM) algorithm. In the SCA and GWO algorithms, the controlled parameters were declined linearly from 2 to zero versus iterations with randomness. The randomness could also be replaced by chaos to improve capability, steadiness, and suitability.

This kind of improvement was just proposed recently with a specific seldom applied chaotic map in β distribution [49]. In this experiment, 19 classical chaotic maps were introduced to replace the β ddistributed chaos, results were shown from Figures 32 to 35.

Results showed that sometimes this kind of improvement could improve the capability of the optimization algorithm, but sometimes the chaotic maps would fail to do so. Regarding Schwefeil 1.2 function, all of the chaotic improved GWO algorithms would perform better than the original, while only six of them could improve the SC algorithm.

4.6. Discussion of the results

Literal researches had been made and four types of classical improvements including a newly proposed version were concluded. Detailed results of works in this paper might be summarized in Tables 7 and 8.

Table 7. Summarize simulation experiments on the GWO algorithm.

Chaos	Unimodal				Multimodal				Unimodal with basins				Unimodal with valleys				B
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	
CM1	W	B	B	W	W	W	W	B	B	B	B	B	B	B	B	B	11

Continued on next page

Chaos	Unimodal				Multimodal				Unimodal with basins				Unimodal with valleys				B
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	
CM2	W	W	B	W	W	W	W	W	W	B	B	W	B	W	B	W	5
CM3	W	B	B	W	B	W	W	B	W	B	B	B	B	B	B	W	10
CM4	W	B	B	W	B	B	B	B	W	B	B	B	B	B	B	W	12
CM5	W	W	B	W	B	B	B	B	W	B	B	B	B	B	B	W	11
CM6	W	B	B	B	W	W	W	W	W	B	B	B	B	W	B	W	8
CM7	W	B	W	W	W	W	W	W	W	W	B	W	B	B	B	W	5
CM8	W	B	B	W	B	B	B	B	B	B	B	B	B	B	B	W	13
CM9	W	B	B	W	B	B	B	B	B	W	B	B	B	B	B	B	13
CM10	W	B	B	W	B	B	W	B	W	W	B	B	B	B	B	W	10
CM11	W	B	B	B	W	B	N	B	B	B	B	W	B	B	B	W	11
CM12	W	B	B	W	W	B	B	W	W	B	B	B	B	B	B	W	10
CM13	W	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	15
CM14	W	B	B	W	W	W	W	W	B	W	B	W	B	W	B	W	6
CM15	W	B	B	W	W		B	B	W	W	B	B	B	B	B	B	10
CM16	W	B	B	W	B	W	W	W	W	B	B	B	B	B	B	W	9
CM17	W	W	B	W	W	W	W	W	W	W	W	N	B	W	B	W	3
CM18	W	B	B	W	B	B	B	B	W	B	B	B	B	B	B	B	13
CM19	W	B	B	W	W	B	B	B	W	B	B	W	B	B	B	W	10
B	0	16	18	3	9	10	9	12	6	13	18	13	19	15	19	5	

Note: In the above table, B means better, represents the chaotic improved algorithm would perform better than the original algorithm; W means worse and the opposite meaning of B; N represents a meanness incensement, that is to say, we cannot find whether the improvement performs better or worse.

Table 7. Summarize simulation experiments on the SC algorithm.

Chaos	Unimodal				Multimodal				Unimodal with basins				Unimodal with valleys				B
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	
CM1	W	B	B	W	W	B	W	B	B	W	B	B	B	W	B	B	10
CM2	W	W	W	W	B	W	W	W	B	W	B	W	W	W	B	B	5
CM3	W	B	W	W	B	B	W	W	B	W	B	W	W	W	W	W	5
CM4	W	B	W	W	B	W	W	W	B	W	B	W	W	B	W	B	6
CM5	B	B	W	W	B	B	W	W	B	W	B	W	W	W	B	W	7
CM6	B	W	W	W	B	B	W	W	B	W	B	W	W	W	W	B	6
CM7	W	W	W	W	B	W	W	W	B	W	B	W	W	B	W	W	4
CM8	W	W	W	W	B	W	W	W	B	W	B	W	W	B	W	B	5
CM9	W	W	W	W	W	W	W	W	B	W	B	W	W	W	W	W	2
CM10	W	W	W	W	B	W	W	W	B	W	B	W	W	W	W	W	3
CM11	W	B	W	B	B	B	W	W	B	W	B	W	B	W	W	B	8
CM12	W	B	W	W	B	B	W	W	B	W	B	W	W	W	W	B	6
CM13	W	W	B	B	B	W	W	W	B	W	B	B	W	W	W	W	6
CM14	W	B	W	B	B	W	W	B	B	W	B	B	B	B	B	B	11
CM15	W	W	W	W	B	W	W	W	B	W	B	W	B	W	B	W	5

Continued on next page

Chaos	Unimodal				Multimodal				Unimodal with basins				Unimodal with valleys				B
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	
CM16	W	W	W	W	B	W	B	W	B	W	B	W	N	W	W	W	4
CM17	W	W	W	W	B	W	W	W	B	W	B	W	W	W	W	W	3
CM18	W	W	B	B	B	B	W	W	B	W	W	B	W	B	B	B	9
CM19	W	B	W	B	B	B	W	W	B	W	B	W	W	B	W	B	8
B	2	8	3	5	17	8	1	2	19	0	18	4	4	6	6	10	

We can find that the chaotic maps could increase the capability of the GWO algorithm more than that of the SC algorithm. Most of the chaotic maps could result in a better performance when they are applied in improving the GWO algorithm, simulation experiments verified their capability. Only a few of them, precisely 6/19, such as the ICMIC, Intermittency, Quadratic, Singer, and Sinusoidal maps could only perform better in less than 10/16 benchmark functions.

On the opposite side, only Bernoulli and Kent map performed better in no less than 10 of 16 benchmark functions. Most of them could not yield better results.

Focusing on the characteristics of benchmark functions, four types including unimodal, multimodal, unimodal with basin-like or valley-like profiles in their three-dimensional profiles [3] of benchmark functions were introduced. However, we still cannot draw confirmative conclusions from the experiment results. For example, from the results of simulation experiments on unimodal benchmark functions with the GWO algorithm, we can find that none of the chaotic improved GWO algorithms could perform better in optimizing Ackley 1 function, meanwhile, 16 and 18 of the 19 chaotic maps could perform better in improvement on the Exponential or Sargan function. Regarding the SC algorithm, most of the chaotic maps failed to do so.

5. Discussion and conclusions

In this paper, we paid attention to the chaotic maps which could generate pseudo-random numbers in an interval between 0 and 1. Nineteen classical chaotic maps were collected and their improvement on the famous GWO and SC algorithms were researched.

Four types of benchmark functions, including two main classical characteristics: modality and uniqueness in profiles, were introduced to carry on the simulation experiments. For the obviousness and simplicity, only convergence ratio experiments were carried out and the convergent curves were shown in figures. We can conclude about the study of chaotic improved research that:

A. Although the chaotic maps could be used to replace the pseudo-random number in computer engineering, they could not increase the performance of algorithms in optimization for sure.

B. Among all of the four types of chaotic improvements, sometimes they could perform better, yet none of them could perform confirmative better for a given problem.

C. Modality and the unique landscape could influence the performance of optimization algorithms, however, they are all not the only reason. A detailed study in mathematics might be needed to find the reason why some of the benchmark functions could be easily optimized, while some of them not.

D. There might be doubtful to apply the chaotic maps in improving optimization algorithms in engineering. It might be OK for a given problem and algorithm, which might occur in some subjects. Nevertheless, due to the uncertainty of a specific chaotic map on all improved ways and functions, it would be unnecessary to establish chaotic improved platforms in computer engineering.

E. It might be a little meaningless to investigate the performance of chaos in optimization henceforth.

Acknowledgments

The authors would like to thank the supports of the following projects: the scientific research team project of Jingchu University of technology with grant number TD202001, and the key research and development project of Jingmen with grant numbers 2019YFZD009.

Conflict of interest

There is no conflict of interest.

References

1. X. S. Yang, Nature-inspired optimization algorithms: Challenges and open problems, *J. Comput. Sci.*, **46** (2020), 101104. https://doi.org/10.1016/j.jocs.2020.101104
2. Z. M. Gao, J. Zhao, An Improved Grey Wolf Optimization Algorithm with Variable Weights, *Comput. Intell. Neurosci.*, **2019** (2019), 2981282. https://doi.org/10.1155/2019/2981282
3. Z. M. Gao, J. Zhao, Benchmark functions with Python, Riga, Latvia: Golden Light Academic Publishing, (2020), 3−5.
4. L. Zhao, T. Zheng, M. Lin, A. Hawbani, J. Shang, C. Fan, SPIDER: A social computing inspired predictive routing scheme for softwarized vehicular networks, *IEEE Trans. Intell. Transp. Syst.*, (2021), 1−12. https://doi.org/10.1109/TITS.2021.3122438
5. L. Zhao, W. Zhao, A. Hawbani, A. Y. Al-Dubai, G. Min, A. Y. Zomaya, et al., Novel online sequential learning-based adaptive routing for edge software-defined vehicular networks, *IEEE Trans. Wireless Commun.*, **20** (2021), 2991−3004. https://doi.org/10.1109/TWC.2020.3046275
6. L. Zhao, C. Wang, K. Zhao, D. Tarchi, S. Wan, N. Kumar, INTERLINK: A digital twin-assisted storage strategy for satellite-terrestrial networks, *IEEE Trans. Aerosp. Electron. Syst.*, (2022). https://doi.org/10.1109/TAES.2022.3169130
7. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software*, **95** (2016), 51−67. https://doi.org/10.1016/j.advengsoft.2016.01.008
8. Y. Chen, G. De Luca, Technologies supporting artificial intelligence and robotics application development, *J. Artif. Intell. Technol.*, **1** (2021), 1−8. https://doi.org/10.37965/jait.2020.0065
9. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95 – International Conference on Neural Networks*, **4** (1995), 1942−1948. https://doi.org/10.1109/ICNN.1995.488968
10. J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, **5** (1997), 4104−4108. https://doi.org/10.1109/ICSMC.1997.637339
11. Y. Shi, R. Eberhart, A modified particle swarm optimizer, in *1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, (1998), 69−73. https://doi.org/ 10.1109/ICEC.1998.699146
12. Y. Shi, R. C. Eberhart, Parameter selection in particle swarm optimization, Springer, Berlin, Heidelberg, (1998), 591−600. https://doi.org/10.1007/BFb0040810

13. P. N. Suganthan, Particle swarm optimiser with neighbourhood operator, in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, **3** (1999), 1958−1962. https://doi.org/10.1109/CEC.1999.785514
14. R. C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, **1** (2000), 84−88. https://doi.org/10.1109/CEC.2000.870279
15. R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, *IEEE Trans. Evol. Comput.*, **8** (2004), 204−210. https://doi.org/10.1109/TEVC.2004.826074
16. M. E. H. Pedersen, A. J. Chipperfield, Simplifying particle swarm optimization, *Appl. Soft Comput.*, **10** (2010), 618−628. https://doi.org/10.1016/j.asoc.2009.08.029
17. Y. Chunming, D. Simon, A new particle swarm optimization technique, *in 18th International Conference on Systems Engineering (ICSEng'05)*, (2005), 164−169. https://doi.org/10.1109/ICSENG.2005.9
18. X. X. Feng, Z. W. Jun, Y. Z. Lian, Dissipative particle swarm optimization, in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, **2** (2002), 1456−1461 https://doi.org/10.1109/CEC.2002.1004457
19. Z. M. Gao, J. Zhao, X. R. Li, Y. R. Hu, An improved sine cosine algorithm with multiple updating ways for individuals, *J. Phys.: Conf. Ser.*, **1678** (2020), 012079. https://doi.org/10.1088/1742-6596/1678/1/012079
20. J. Kennedy, Bare bones particle swarms, in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706)*, (2003), 80−87. https://doi.org/10.1109/SIS.2003.1202251
21. A. P. Engelbrecht, Heterogeneous particle swarm optimization, in *Swarm Intelligence*, Springer, Berlin, Heidelberg, **6234** (2010). https://doi.org/10.1007/978-3-642-15461-4_17
22. P. J. Angeline, Using selection to improve particle swarm optimization, in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, (1998), 84−89. https://doi.org/10.1109/ICEC.1998.699327
23. H. Ullah, B. Ahmad, I. Sana, A. Sattar, A. Khan, S. Akbar, et al. Comparative study for machine learning classifier recommendation to predict political affiliation based on online reviews, *CAAI Trans. Intell. Technol.*, **6** (2021), 251−264. https://doi.org/10.1049/cit2.12046
24. H. Haklı, H. Uğuz, A novel particle swarm optimization algorithm with Levy flight, *Appl. Soft Comput.*, **23** (2014), 333−345. https://doi.org/10.1016/j.asoc.2014.06.034
25. E. Emary, H. M. Zawbaa, M. Sharawi, Impact of Lèvy flight on modern meta-heuristic optimizers, *Appl. Soft Comput.*, **75** (2019), 775−789. https://doi.org/10.1016/j.asoc.2018.11.033
26. B. Abdollahzadeh, F. S. Gharehchopogh, S. Mirjalili, African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems, *Comput. Ind. Eng.*, **158** (2021), 107408. https://doi.org/10.1016/j.cie.2021.107408
27. D. Ghosh, J. Singh, Spectrum-based multi-fault localization using Chaotic Genetic Algorithm, *Inf. Software Technol.*, **133** (2021), 106512. https://doi.org/10.1016/j.infsof.2021.106512
28. B. Alatas, Chaotic harmony search algorithms, *Appl. Math. Comput.*, **216** (2010), 2687−2699. https://doi.org/10.1016/j.amc.2010.03.114
29. L. Ding, H. Wu, Y. Yao, Chaotic artificial bee colony algorithm for system identification of a small-scale unmanned helicopter, *Int. J. Aerosp. Eng.*, **2015** (2015), 801874. https://doi.org/10.1155/2015/801874

30. A. H. Gandomi, X. S. Yang, S. Talatahari, A. H. Alavi, Firefly algorithm with chaos, *Commun. Nonlinear Sci. Numer. Simul.*, **18** (2013), 89−98. https://doi.org/10.1016/j.cnsns.2012.06.009
31. J. Jiang, X. Yang, X. Meng, K. Li, Enhance chaotic gravitational search algorithm (CGSA) by balance adjustment mechanism and sine randomness function for continuous optimization problems, *Phys. A*, **537** (2020), 122621. https://doi.org/10.1016/j.physa.2019.122621
32. A. H. Gandomi, X. S. Yang, Chaotic bat algorithm, *J. Comput. Sci.*, **5** (2014), 224−232. https://doi.org/10.1016/j.jocs.2013.10.002
33. X. S. Yang, A new metaheuristic bat-inspired algorithm, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Springer, Berlin, Heidelberg, **284** (2010), 65−74. https://doi.org/10.1007/978-3-642-12538-6_6
34. A. Farshin, S. Sharifian, A chaotic grey wolf controller allocator for Software Defined Mobile Network (SDMN) for 5th generation of cloud-based cellular systems (5G), *Comput. Commun.*, **108** (2017), 94−109. https://doi.org/10.1016/j.comcom.2017.05.003
35. W. Zhu, H. Duan, Chaotic predator–prey biogeography-based optimization approach for U CAV path planning, *Aerosp. Sci. Technol.*, **32** (2014), 153−161. https://doi.org/10.1016/j.as t.2013.11.003
36. G. Kaur, S. Arora, Chaotic whale optimization algorithm, *J. Comput. Des. Eng.*, **5** (2018), 275−284. https://doi.org/10.1016/j.jcde.2017.12.006
37. R. M. Rizk-Allah, A. E. Hassanien, S. Bhattacharyya, Chaotic crow search algorithm for fractional optimization problems, *Appl. Soft Comput.*, **71** (2018), 1161−1175. https://doi.org /10.1016/j.asoc.2018.03.019
38. M. Mitić, N. Vuković, M. Petrović, Z. Miljković, Chaotic fruit fly optimization algorithm, *Knowledge-Based Syst.*, **89** (2015), 446−458. https://doi.org/10.1016/j.knosys.2015.08.010
39. M. Kohli, S. Arora, Chaotic grey wolf optimization algorithm for constrained optimization problems, *J. Comput. Des. Eng.*, **5** (2018), 458−472. https://doi.org/10.1016/j.jcde.2017.02.005
40. A. Erramilli, R. P. Singh, P. Pruthi, Chaotic maps as models of packet traffic, *Teletraffic Sci. Eng.*, **1** (1994), 329−338. https://doi.org/10.1016/B978-0-444-82031-0.50040-8
41. Chaotic Maps, 2021. Available from: https://www.mathworks.com/matlabcentral/fileexchang e/7370-chaotic-maps.
42. Y. Hu, J. Gong, Y. Jiang, L. Liu, G. Xiong, H. Chen, Hybrid map-based navigation method for unmanned ground vehicle in urban scenario, *Remote Sens.*, **5** (2013), 3662−3680. https://doi.org/10.3390/rs5083662
43. D. He, C. He, L. G. Jiang, H. W. Zhu, G. R. Hu, Chaotic characteristics of a one-dimensional iterative map with infinite collapses, *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.*, **48** (2001), 900−906. https://doi.org/10.1109/81.933333
44. I. Fister, M. Perc, S. M. Kamal, I. Fister, A review of chaos-based firefly algorithms: Perspectives and research challenges, *Appl. Math. Comput.*, **252** (2015), 155−165. https://doi.org/10.1016/j.amc.2014.12.006
45. G. Pastor, M. Romera, F. Montoya, A revision of the Lyapunov exponent in 1D quadratic maps, *Physica D*, **107** (1997), 17−22. https://doi.org/10.1016/S0167-2789(97)00057-2
46. T. Xiang, X. Liao, K. Wong, An improved particle swarm optimization algorithm combined with piecewise linear chaotic map, *Appl. Math. Comput.*, **190** (2007), 1637−1645. https://doi.org/10.1016/j.amc.2007.02.103

47. J. Zhao, Z. M. Gao, B. L. Jia, The improved slime mould algorithm with piecewice map, in *2020 International Symposium on Computer Engineering and Intelligent Communications (ISCEIC)*, (2020), 25−29. https://doi.org/10.1109/ISCEIC51027.2020.00013
48. M. Khishe, M. R. Mosavi, Chimp optimization algorithm, *Expert Syst. Appl.*, **149** (2020), 113338. https://doi.org/10.1016/j.eswa.2020.113338
49. A. Saxena, R. Kumar, S. Das, β-chaotic map enabled grey wolf optimizer, *Appl. Soft Comput.*, **75** (2019), 84−105. https://doi.org/10.1016/j.asoc.2018.10.044
50. H. Li, S. Wang, M. Ji, An improved chaotic ant colony algorithm, in *Proceedings of the 9th international conference on Advances in Neural Networks*, Springer, Berlin, Heidelberg, (2012), 633−640. https://doi.org/10.1007/978-3-642-31346-2_71
51. B. Wu, S. Fan, Improved artificial bee colony algorithm with chaos, in *Computer Science for Environmental Engineering and EcoInformatics*, Springer, Berlin, Heidelberg, **158** (2011), 51−56. https://doi.org/10.1007/978-3-642-22694-6_8
52. A. A. Ateya, A. Muthanna, A. Vybornova, A. D. Algarni, A. Abuarqoub, Y. Koucheryavy, et al., Chaotic salp swarm algorithm for SDN multi-controller networks, *Eng. Sci. Technol. Int. J.*, **22** (2019), 1001−1012. https://doi.org/10.1016/j.jestch.2018.12.015
53. X. Liang, Z. Cai, M. Wang, X. Zhao, H. Chen, C. Li, Chaotic oppositional sine–cosine method for solving global optimization problems, *Eng. Comput.*, **38** (2022), 1223–1239. https://doi.org/10.1007/s00366-020-01083-y
54. S. I. Boushaki, N. Kamel, O. Bendjeghaba, A new quantum chaotic cuckoo search algorithm for data clustering, *Expert Syst. Appl.*, **96** (2018), 358−372. https://doi.org/10.1016/j.eswa.2017.12.001
55. S. Aggarwal, P. Chatterjee, R. P. Bhagat, K. K. Purbey, S. J. Nanda, A social spider optimization algorithm with chaotic initialization for robust clustering, *Procedia Comput. Sci.*, **143** (2018), 450−457. https://doi.org/10.1016/j.procs.2018.10.417
56. A. A. Dehkordi, A. S. Sadiq, S. Mirjalili, K. Z. Ghafoor, Nonlinear-based chaotic harris hawks optimizer: Algorithm and internet of vehicles application, *Appl. Soft Comput.*, (2021), 107574. https://doi.org/10.1016/j.asoc.2021.107574
57. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software*, **69** (2014), 46−61. https://doi.org/10.1016/j.advengsoft.2013.12.007
58. S. Mirjalili, SCA: A Sine Cosine Algorithm for solving optimization problems, *Knowledge Based Syst.*, **96** (2016), 120−133. https://doi.org/10.1016/j.knosys.2015.12.022
59. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849−872. https://doi.org/10.1016/j.future.2019.02.028
60. A. Faramarzi, M. Heidarinejad, B. Stephens, S. Mirjalili, Equilibrium optimizer: A novel optimization algorithm, *Knowledge Based Syst.*, **191** (2019), 105190. https://doi.org/10.1016/j.knosys.2019.105190
61. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, *Future Gener. Comput. Syst.*, **111** (2020), 300−323. https://doi.org/10.1016/j.future.2020.03.055
62. Z. M. Gao, J. Zhao, S. R. Li, The improved slime mould algorithm with cosine controlling parameters, *J. Phys.: Conf. Ser.*, **1631** (2020), 012083. https://doi.org/10.1088/1742-6596/1631/1/012083

Appendix

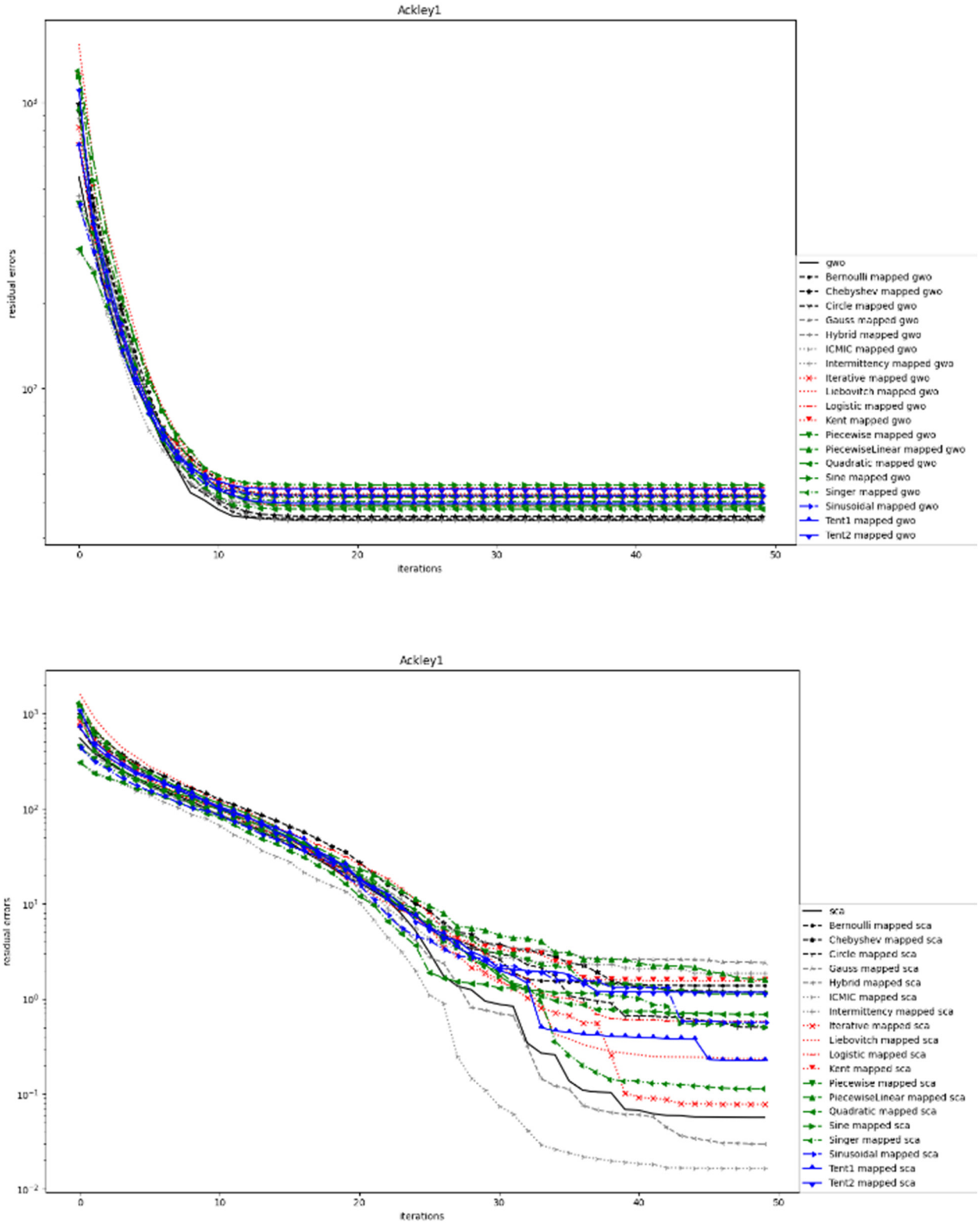

Figure 20. Convergence with iterations for Ackley 1 function.

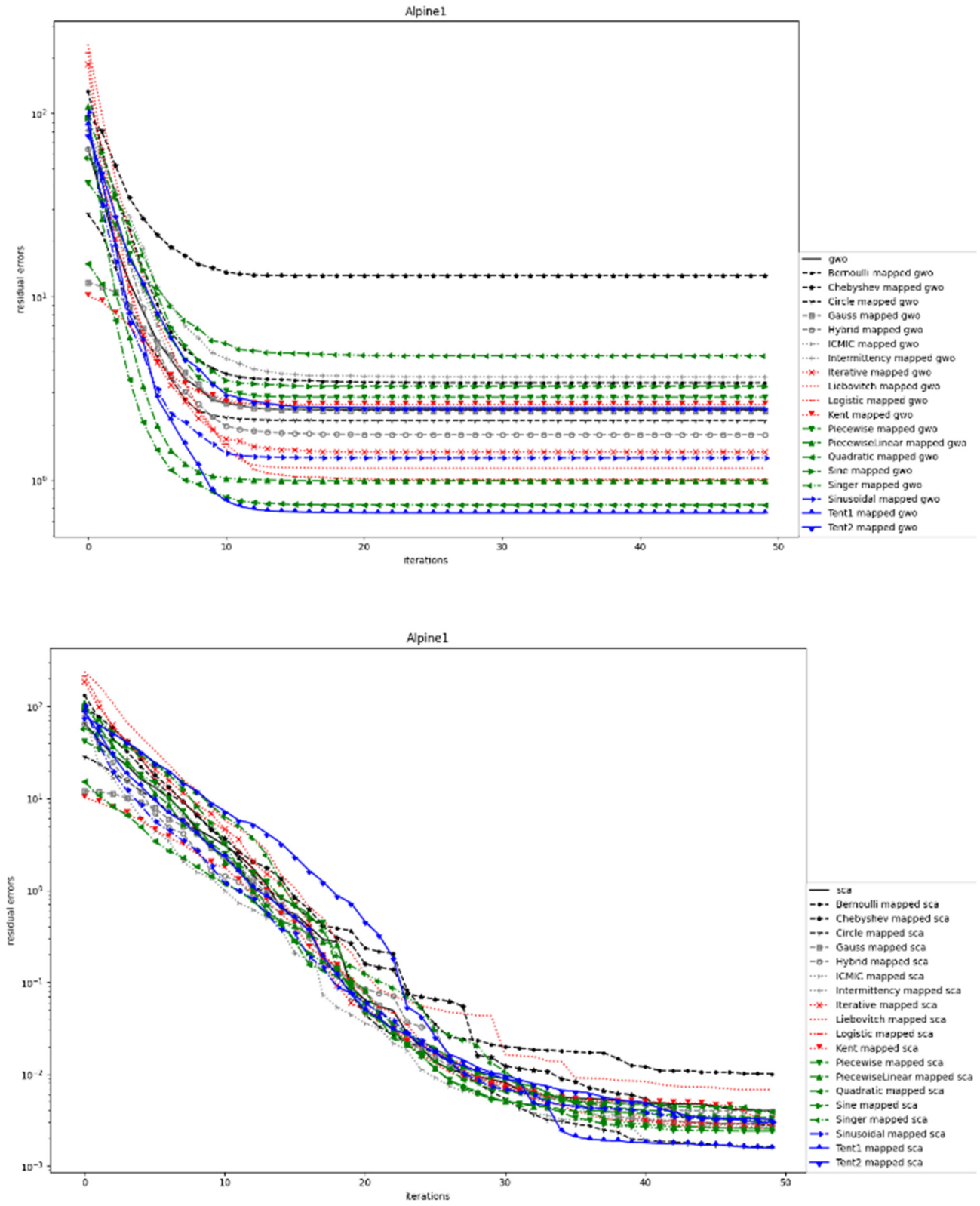

Figure 21. Convergence with iterations for Alpine 1 function.

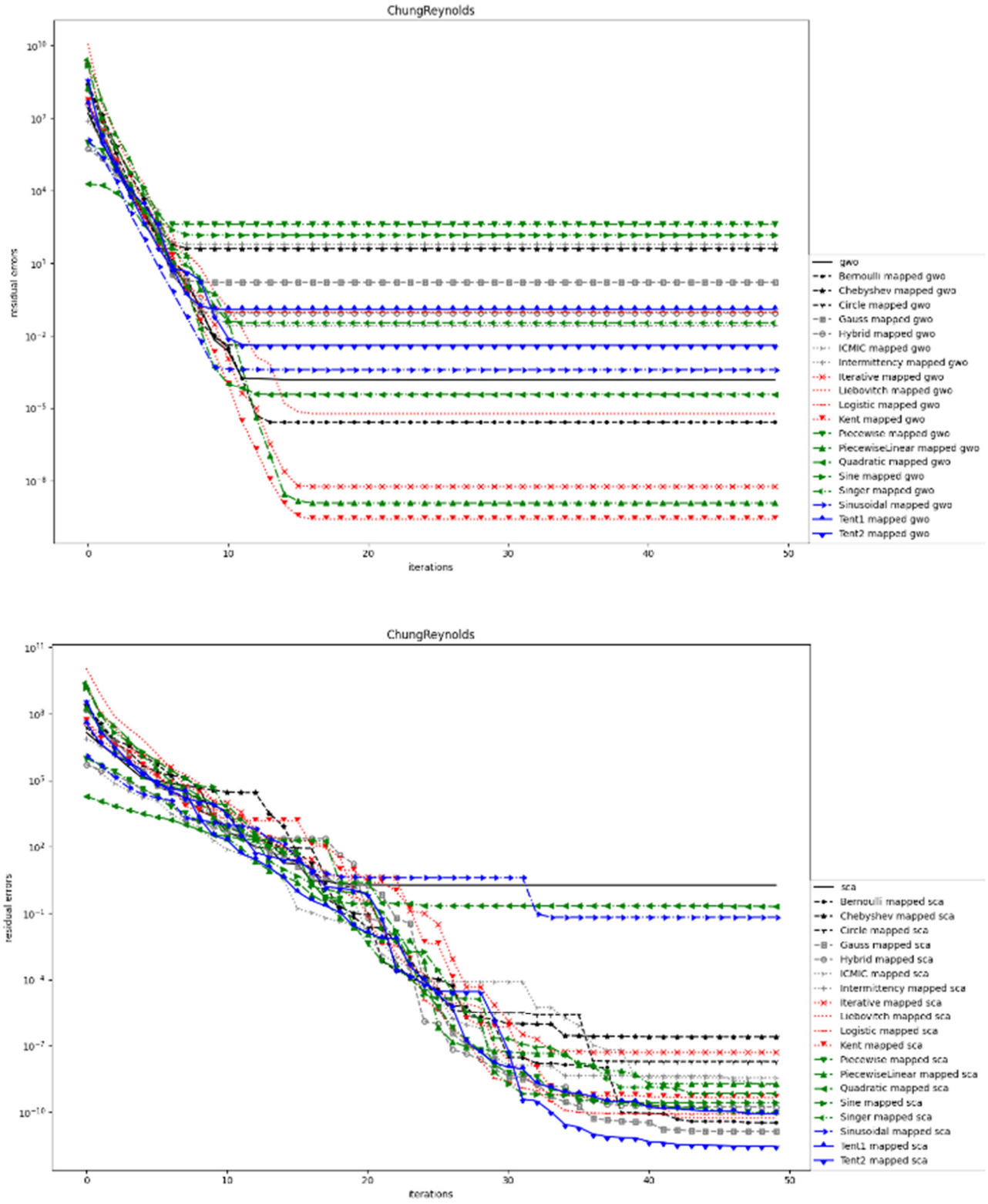

Figure 22. Convergence with iterations for Chung Reynolds function.

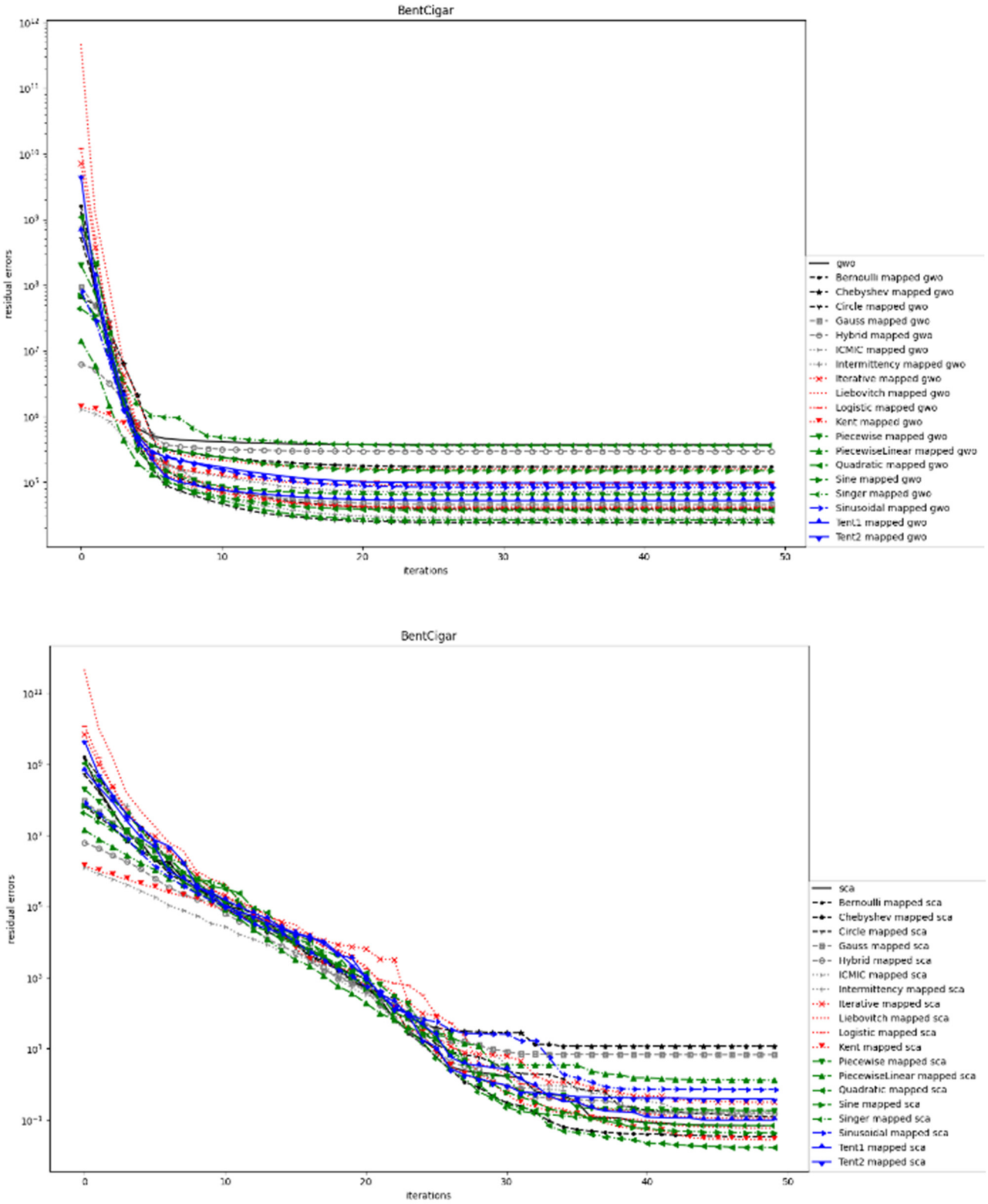

Figure 23. Convergence with iterations for Bent Cigar function.

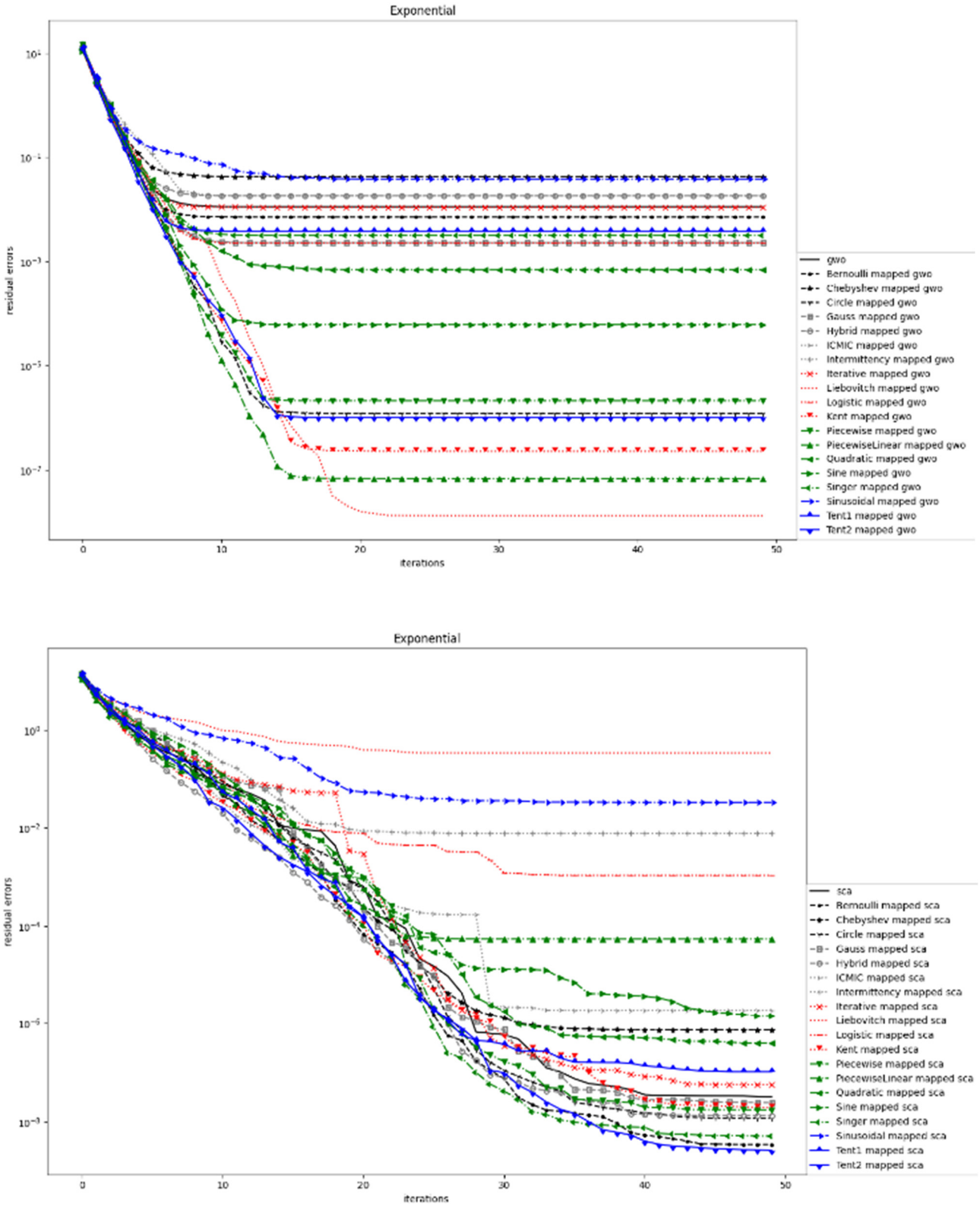

Figure 24. Convergence with iterations for Exponential function.

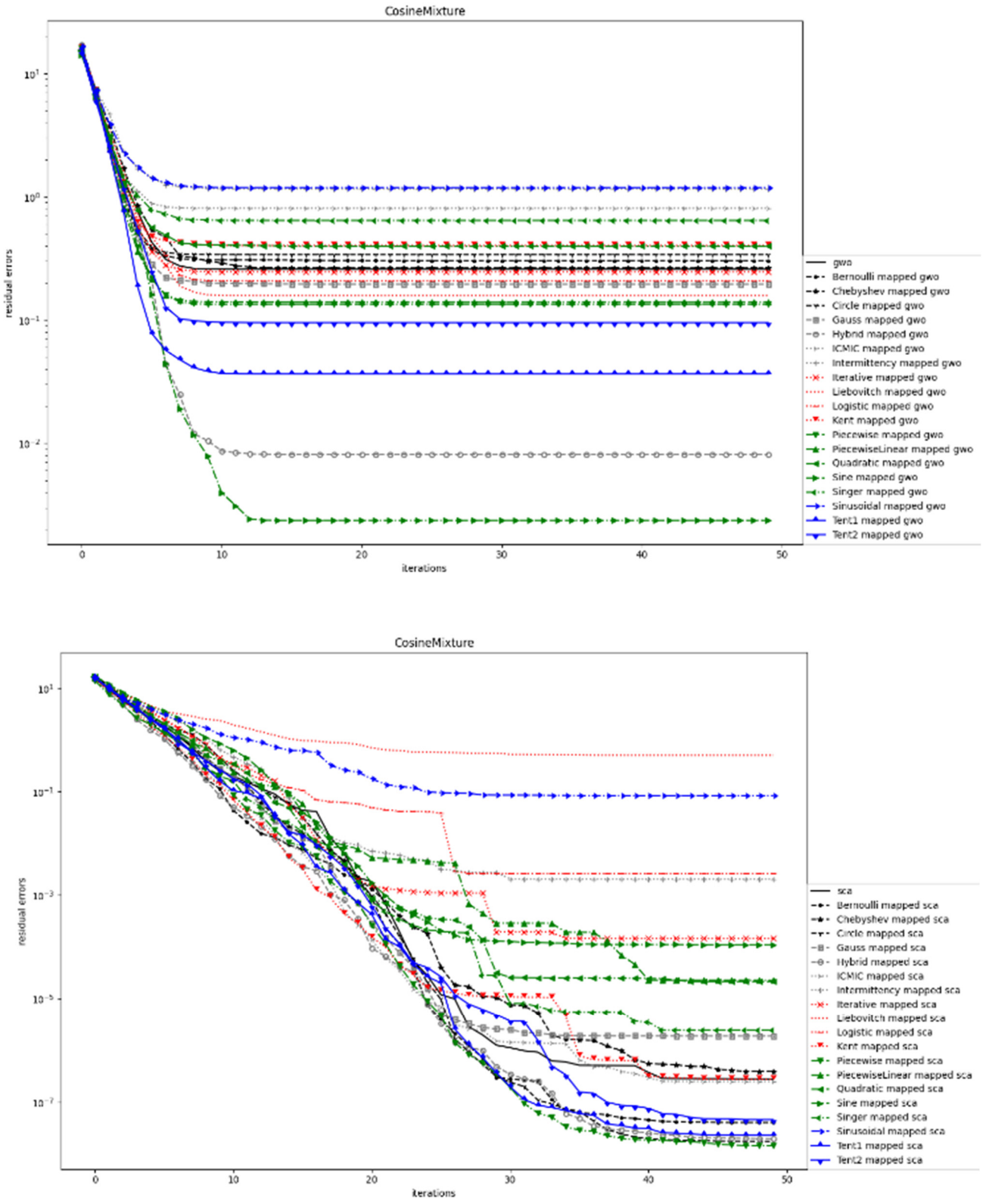

Figure 25. Convergence with iterations for Cosine Mixture function.

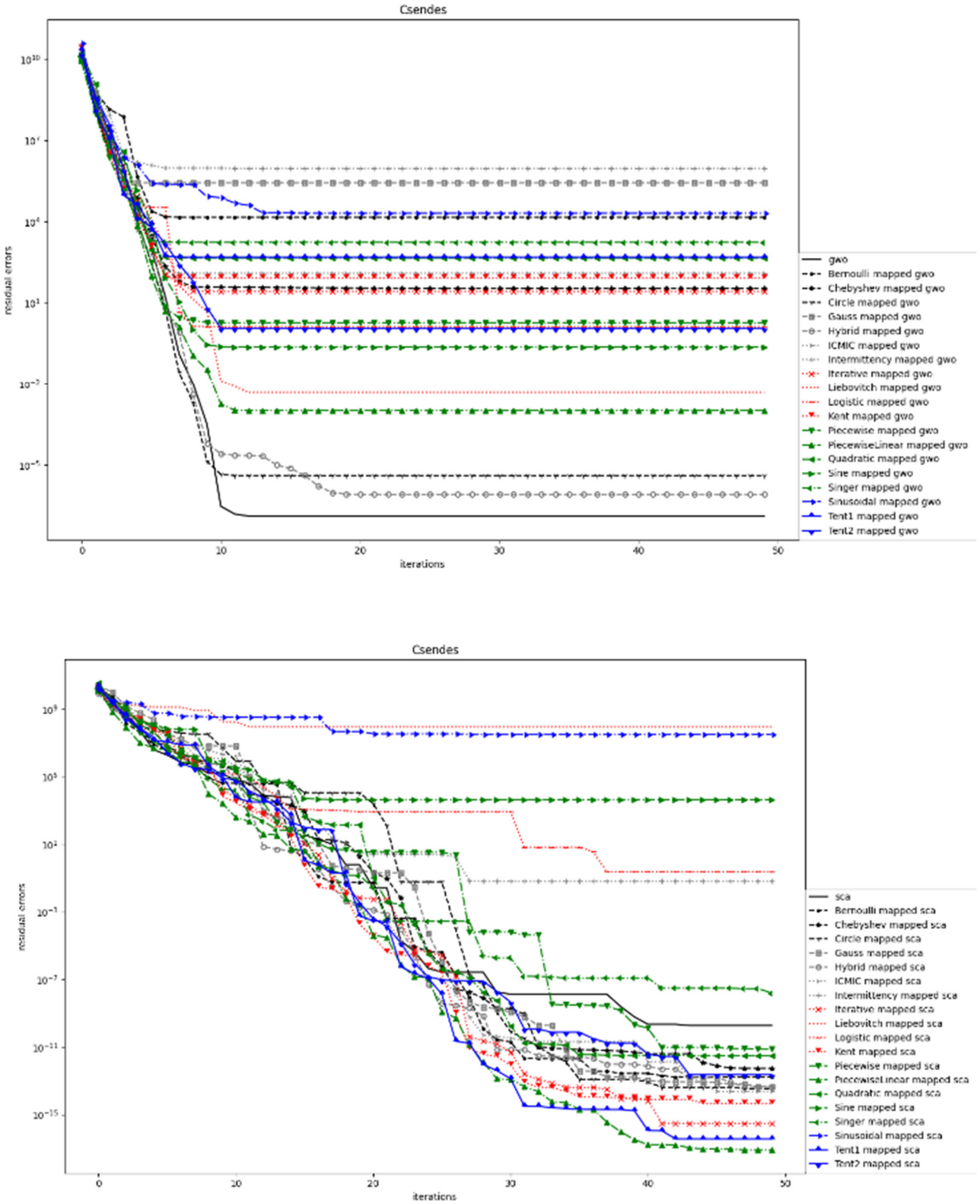

Figure 26. Convergence with iterations for Csendes function.

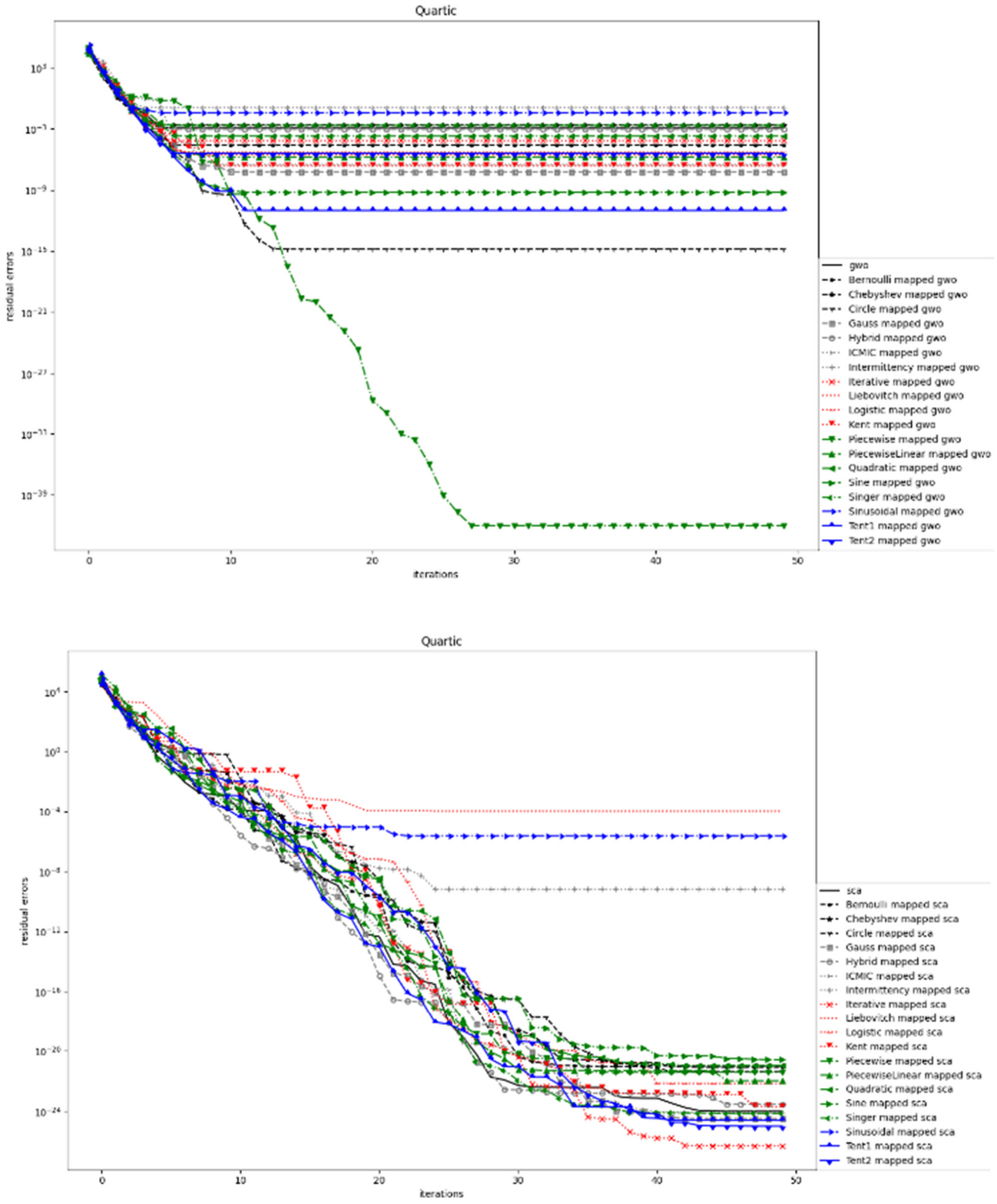

Figure 27. Convergence with iterations for Quartic function.

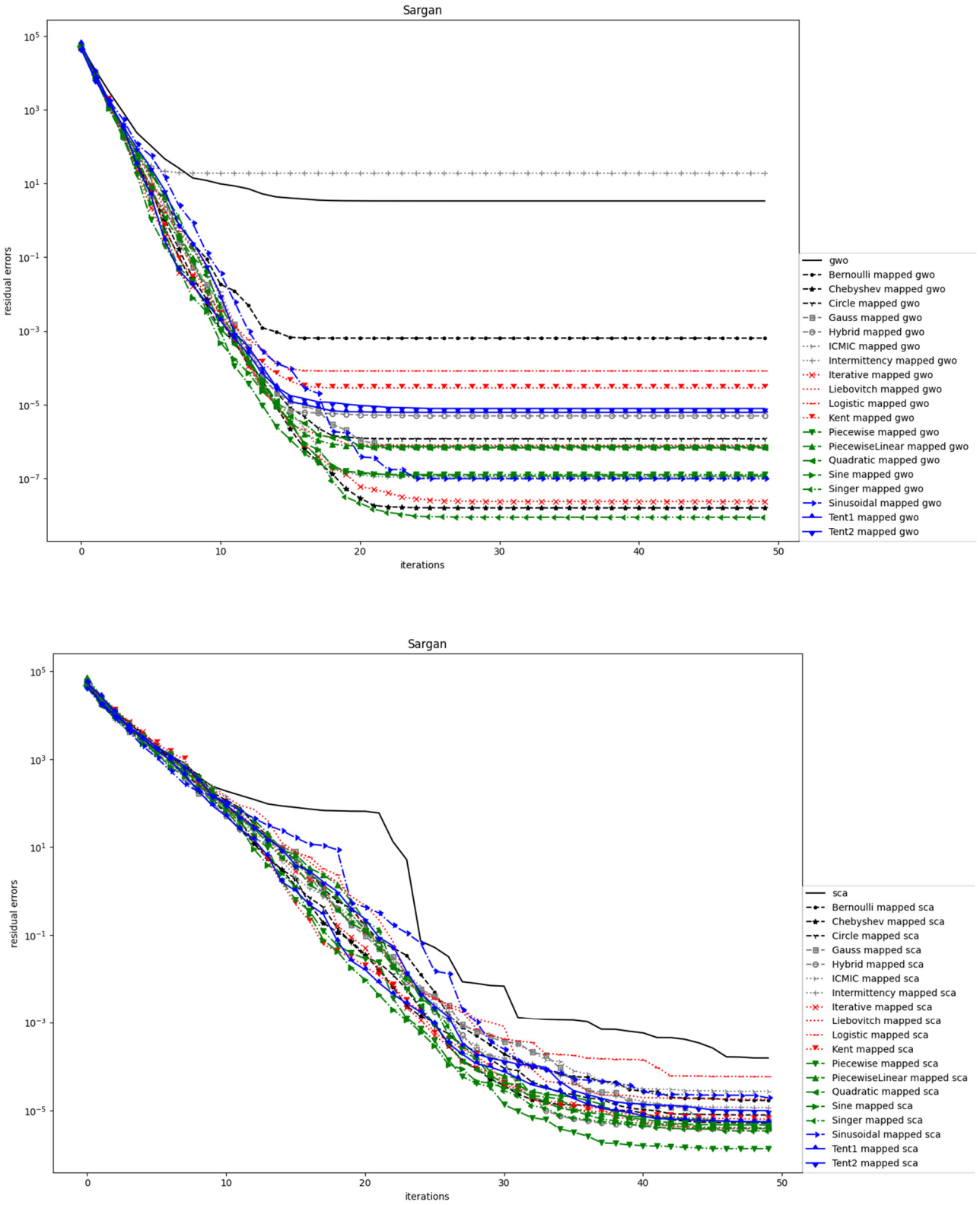

Figure 28. Convergence with iterations for Sargan function.

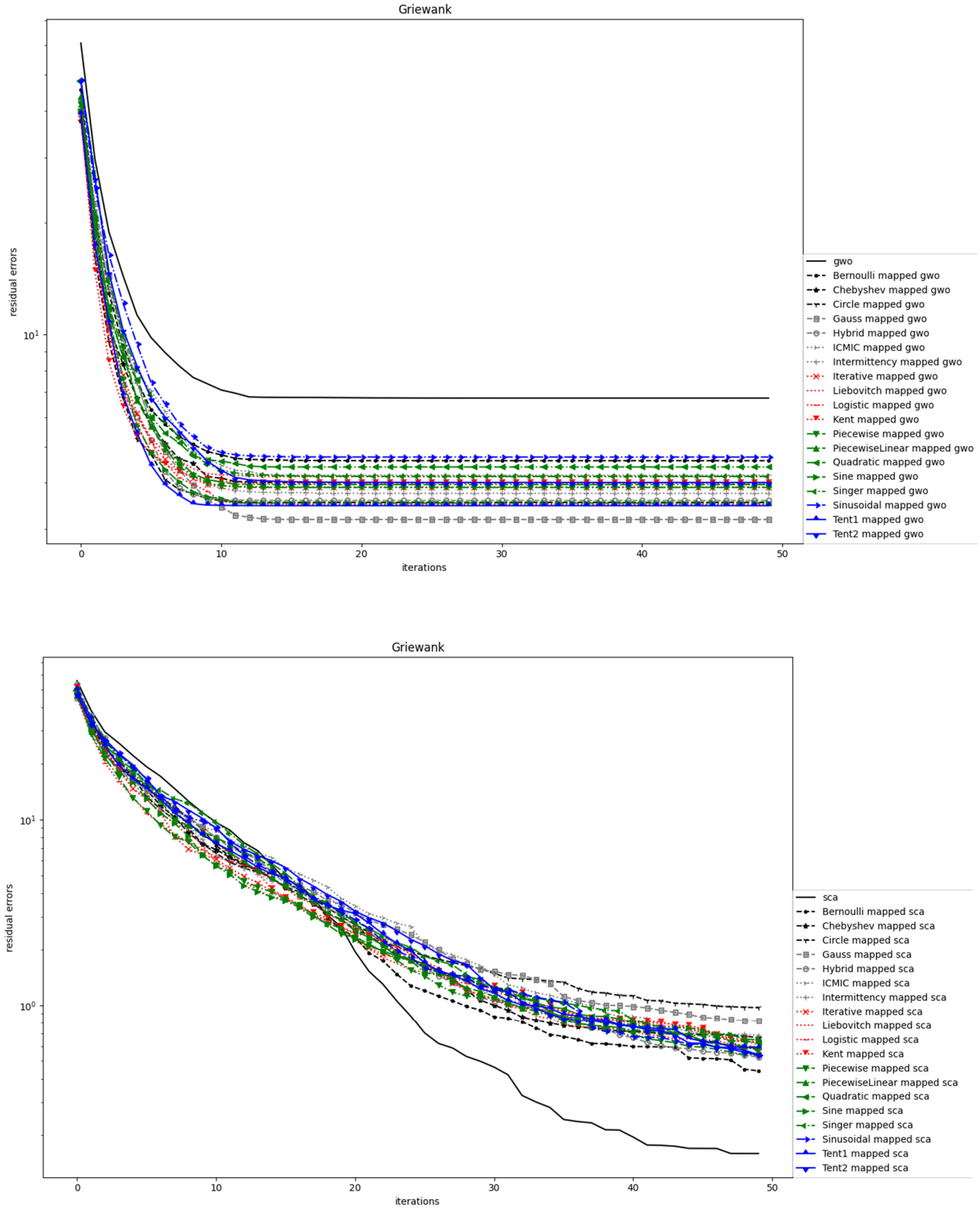

Figure 29. Convergence with iterations for Griewank function.

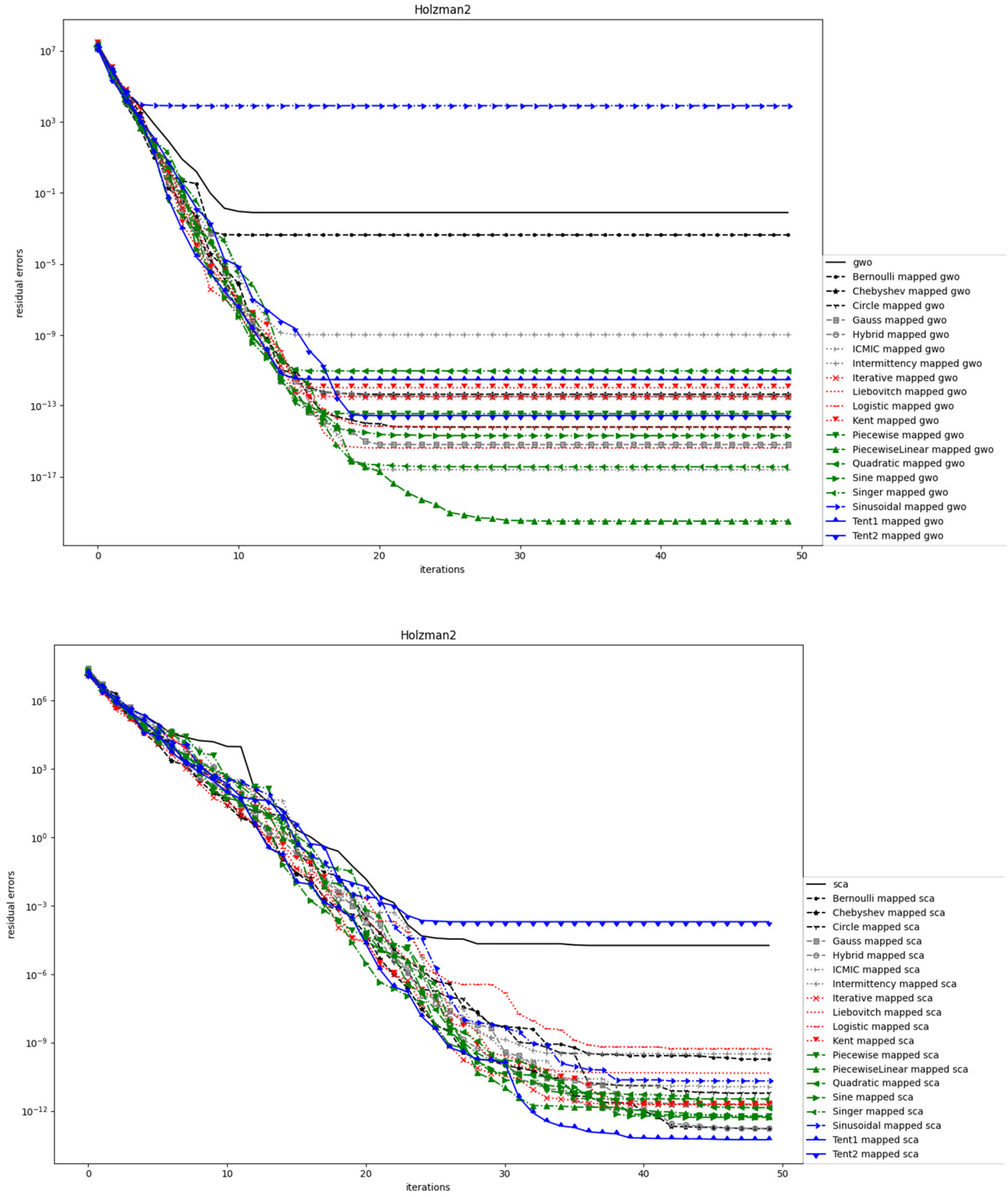

Figure 30. Convergence with iterations for Holzman 2 function.

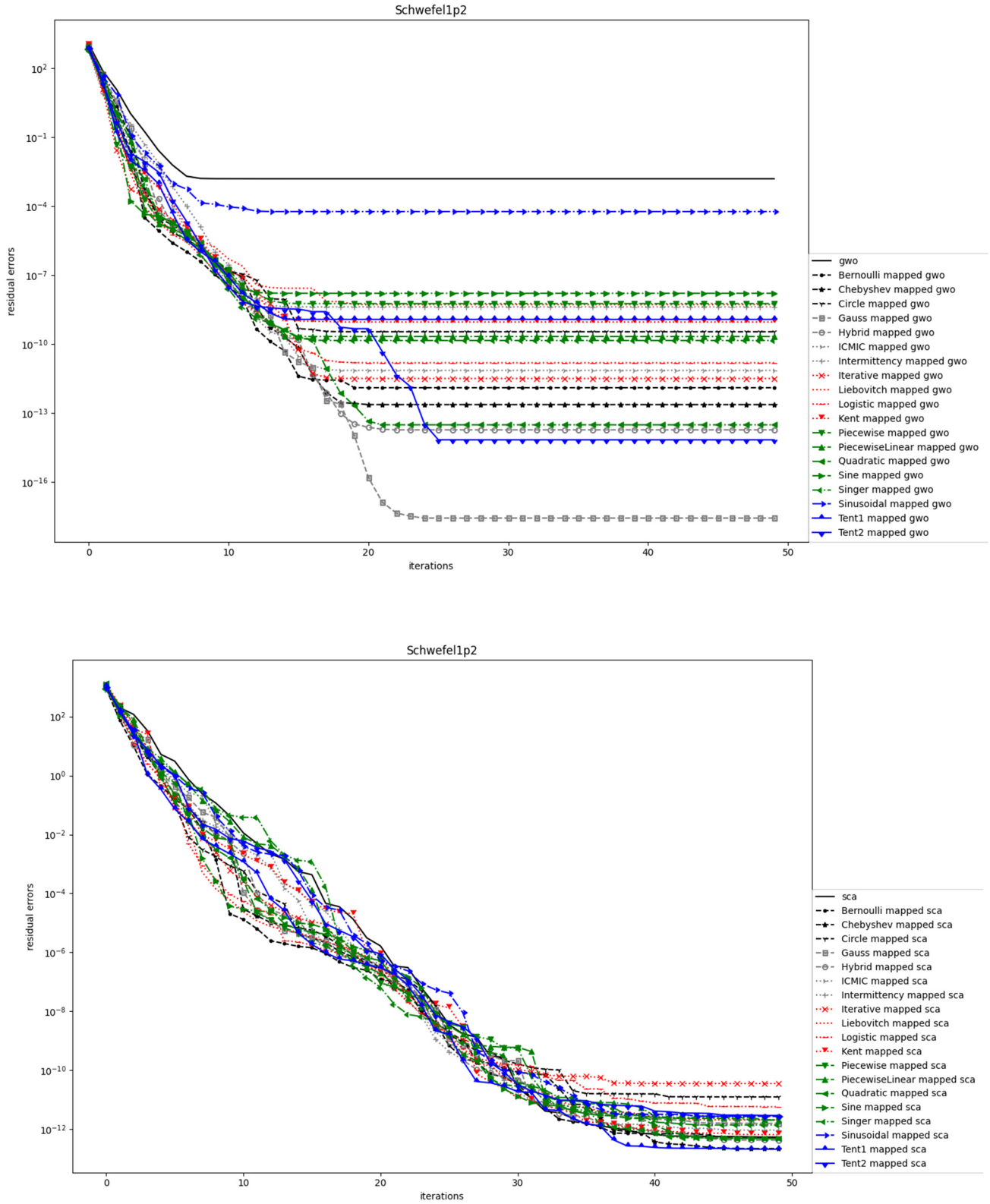

Figure 31. Convergence with iterations for Schwefel 1.2 function.

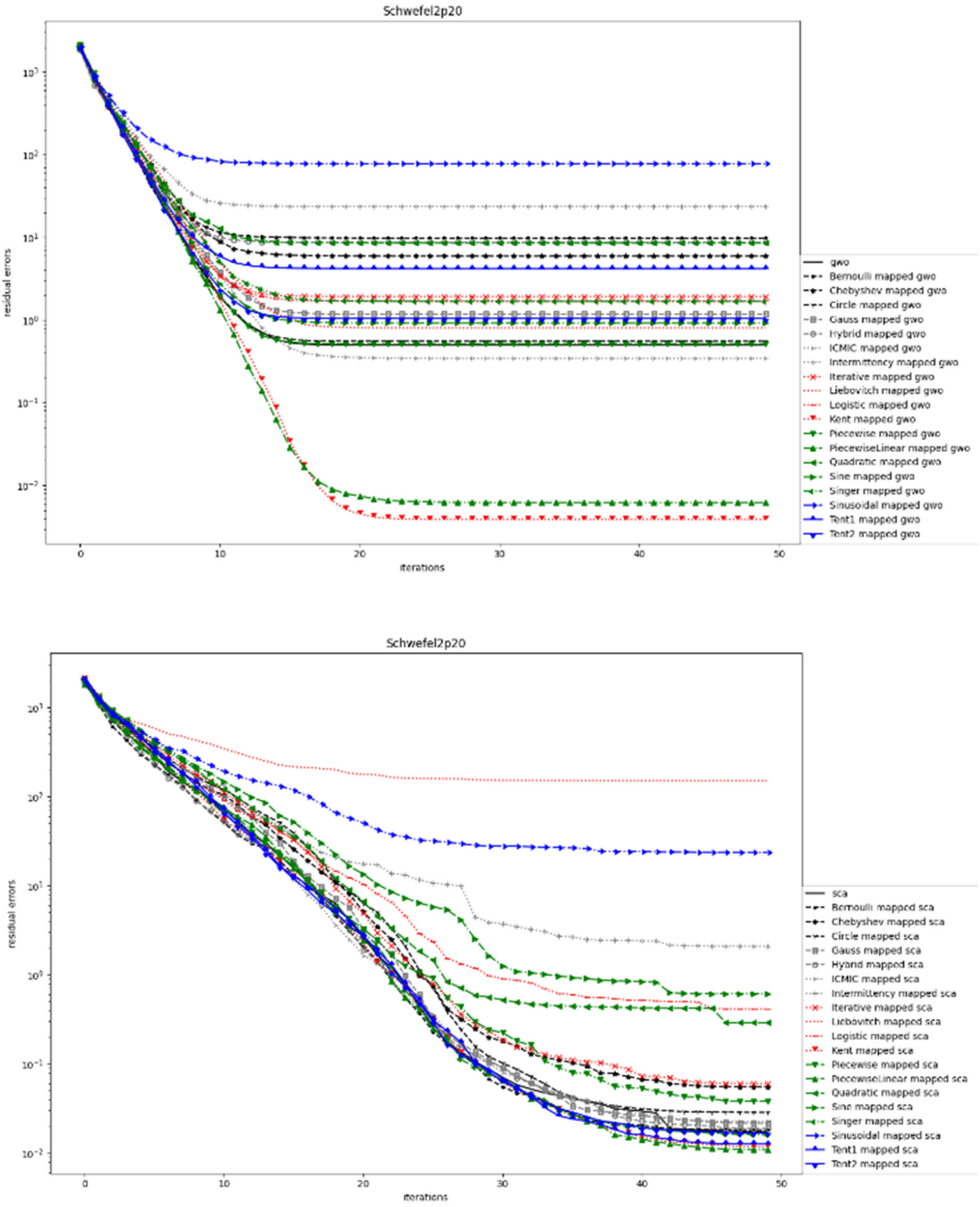

Figure 32. Convergence with iterations for Schwefel 2.20.

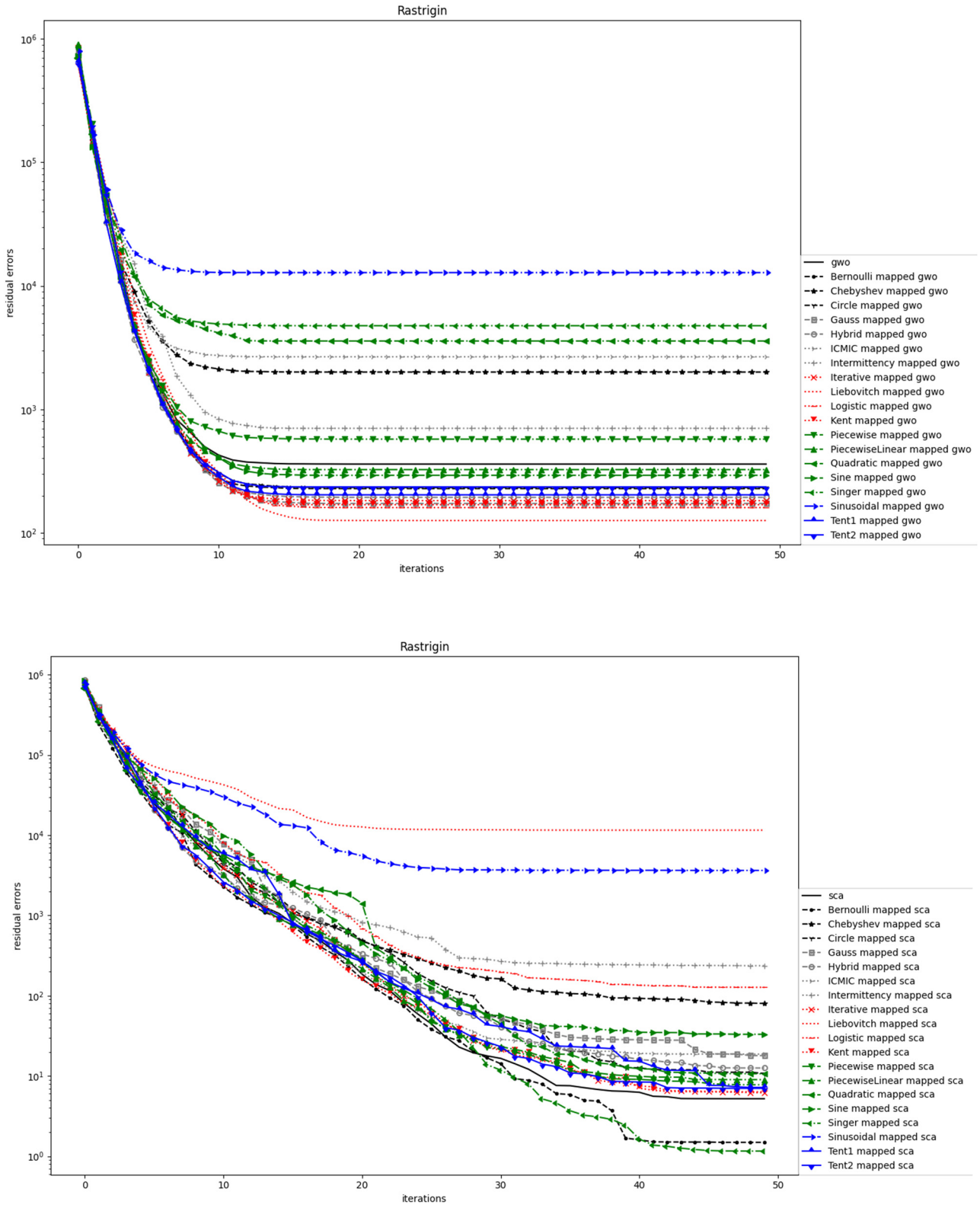

Figure 33. Convergence with iterations for Rastrigin function.

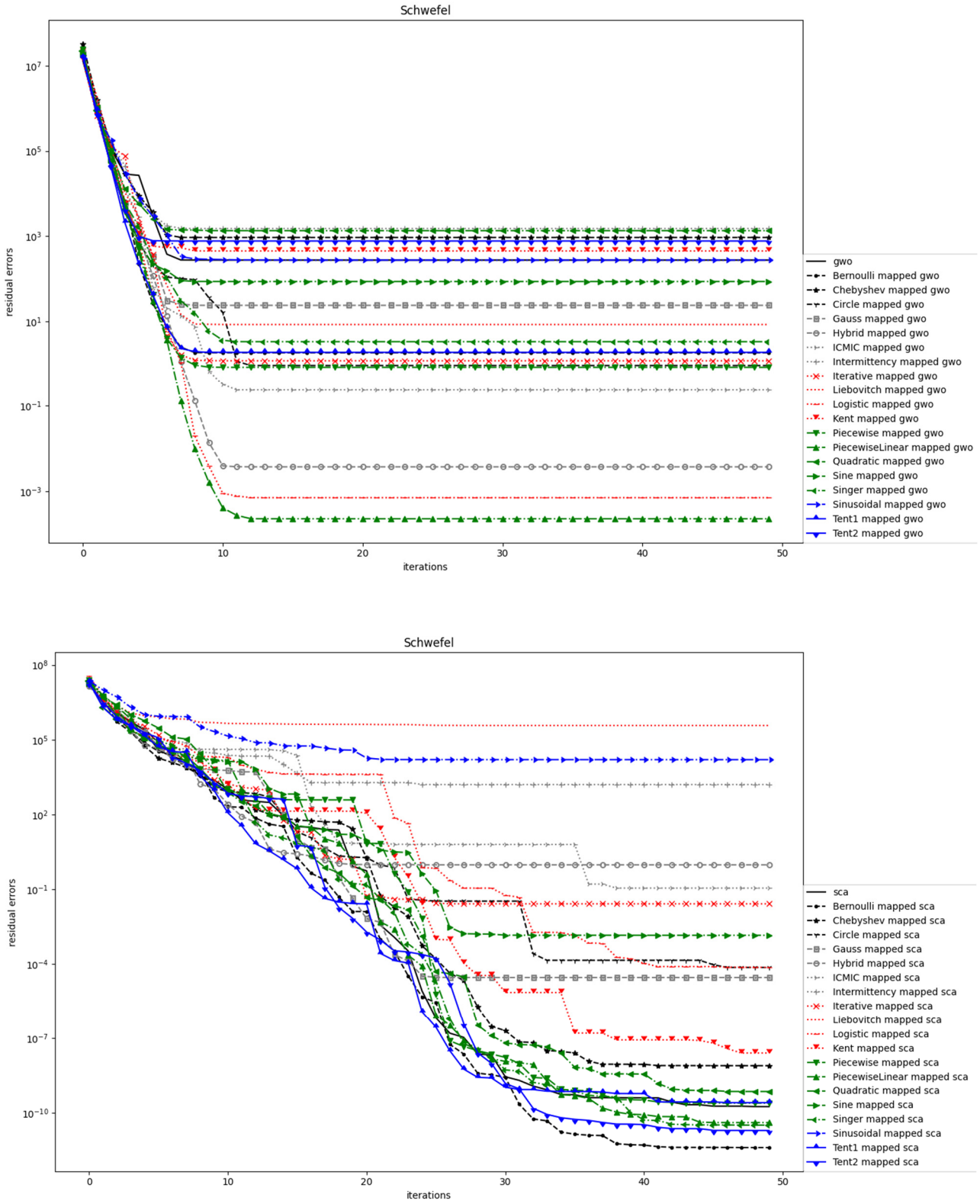

Figure 34. Convergence with iterations for Schwefel function.

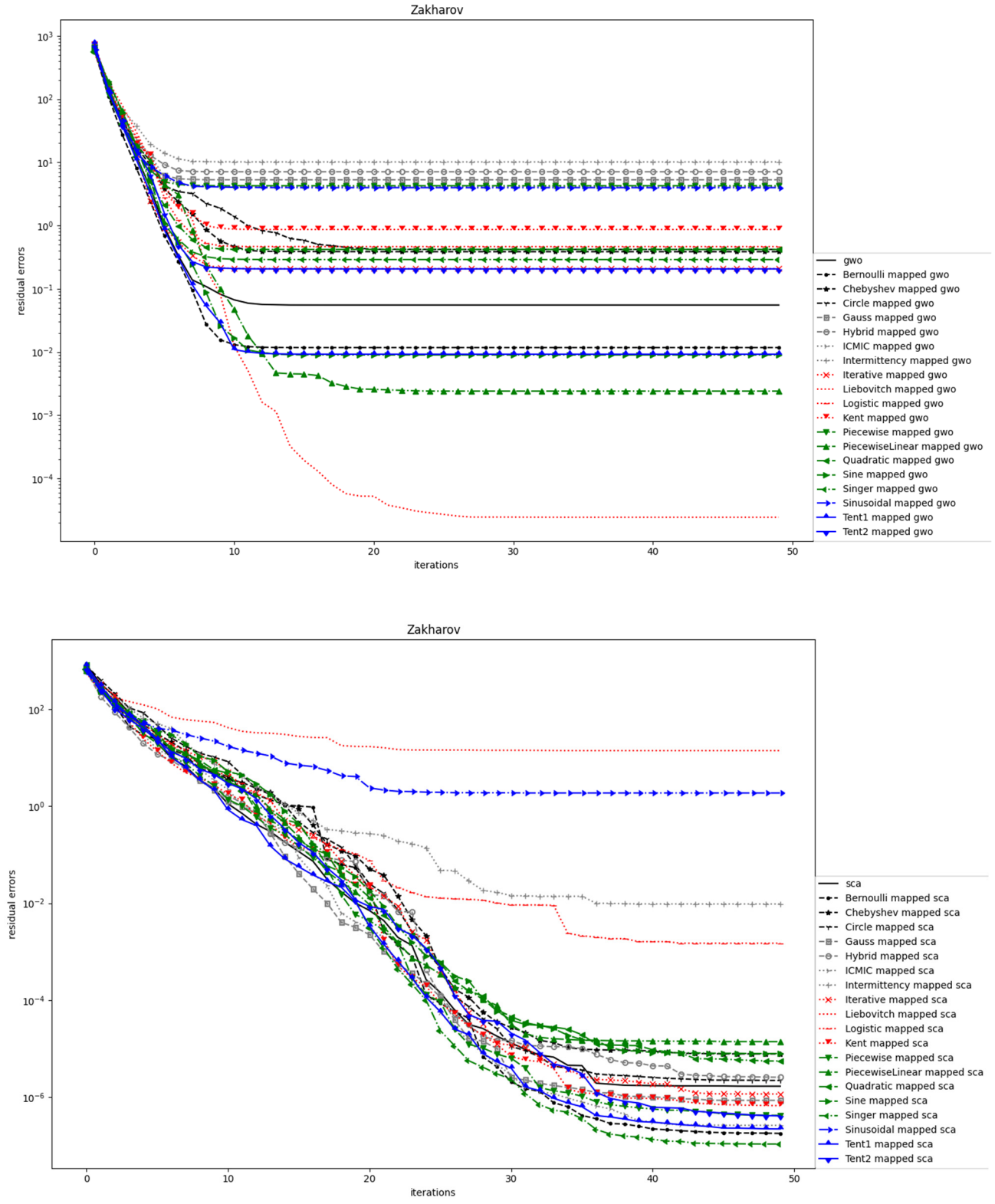

Figure 35. Convergence with iterations for Zakharov function function.