On a New Generalised Iteration Method in the PSO-based Newton-like Method

Ireneusz Gościniak[1(✉)][0000−0003−1315−6082] and Krzysztof Gdawiec[1][0000−0001−9434−9307]

Institute of Computer Science, University of Silesia, Będzińska 39, 41–200 Sosnowiec, Poland
{ireneusz.gosciniak,krzysztof.gdawiec}@us.edu.pl

Abstract. The root-finding problem is very important in many applications and has become an extensive research field. One of the directions in this field is the use of various iteration schemes. In this paper, we propose a new generalised iteration scheme. The schemes like Mann, Ishikawa, Das–Debata schemes are special cases of the proposed iteration. Moreover, we use the proposed iteration with the PSO-based Newton-like method in two tasks. In the first task, we search for the roots, whereas in the second one for patterns with aesthetic features. The obtained results show that the proposed iteration can decrease the average number of iterations needed to find the roots and that we can generate patterns with potential artistic applications.

Keywords: root-finding · dynamics · iterations · visualisation.

1 Introduction

Let $f_1, f_2, \ldots, f_D : \mathbb{R}^D \to \mathbb{R}$ and let

$$\mathbf{F}(z^1, z^2, \ldots, z^D) = \begin{bmatrix} f_1(z^1, z^2, \ldots, z^D) \\ f_2(z^1, z^2, \ldots, z^D) \\ \vdots \\ f_D(z^1, z^2, \ldots, z^D) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}. \tag{1}$$

Moreover, let us assume that $\mathbf{F} : \mathbb{R}^D \to \mathbb{R}^D$ is a continuous function and has continuous partial derivatives to appropriate order. To find the roots of $\mathbf{F}$, i.e., solve $\mathbf{F}(\mathbf{z}) = \mathbf{0}$, where $\mathbf{z} = [z^1, z^2, \ldots, z^D]$, we can use the Newton's method [3].

The root-finding problem (1) is very important in many applications [3,9]. Therefore, in the literature, we can find many root-finding methods. One of the methods is the mentioned earlier Newton's method which was the base for many other methods. For instance, in [6] the authors combined Newton's method with the Particle Swarm Optimisation (PSO) approach obtaining a method in which we can control the dynamics of the method very easily. Many of the root-finding methods are based on the finding of fixed points, so the use of the iteration

methods for approximate finding such point was proposed in recent years [2,5], and became a very popular study direction.

In this paper, we propose a new iteration scheme allowing an extension of the number of mappings. Moreover, we use the proposed scheme in the root-finding problem using the PSO-based Newton-like methods proposed in [6], and to generate patterns with aesthetic features.

The paper is organised as follows. In Sec. 2 we introduce the PSO-based Newton-like root-finding method. Then, in Sec. 3, we review some of the iteration schemes known in the literature. The new generalised iteration scheme is proposed in Sec. 4. In Sec. 5, we present the experimental setup, and in Sec. 6, we discuss the obtained results. Finally, in Sec. 7, we give the concluding remarks.

2 PSO-based Newton-like Method

In [6], the authors introduced a PSO-based Newton-like method to solve (1). The method combines the well-known Newton's method with the idea of PSO, namely it implements the best position of particle similarly to the PSO.

The method is defined using the following formula:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{v}_{n+1}, \tag{2}$$

where $\mathbf{z}_0 \in \mathbb{R}^D$ is the starting point, $\mathbf{v}_0 = [0, 0, \ldots, 0]$ is the starting velocity, $\mathbf{v}_{n+1}$ is the current velocity ($\mathbf{v}_{n+1} = [v^1_{n+1}, v^2_{n+1}, \ldots, v^D_{n+1}]$), $\mathbf{z}_n$ is the previous position point ($\mathbf{z}_n = [z^1_n, z^2_n, \ldots, z^D_n]$). The algorithm sums the position of the particle $\mathbf{z}_n$ with its current velocity $\mathbf{v}_{n+1}$, which is given by the formula:

$$\mathbf{v}_{n+1} = \omega\mathbf{v}_n + \eta\mathbf{N}(\mathbf{z}_n), \tag{3}$$

where $\mathbf{v}_n$ – the previous velocity of the particle, $\omega \in [0, 1)$ – inertia weight, $\eta \in (0, 1]$ – acceleration constant. Moreover, $\mathbf{N}$ represents the Newton's method part, and it is defined by

$$\mathbf{N}(\mathbf{z}) = -\mathbf{J}^{-1}(\mathbf{z})\mathbf{F}(\mathbf{z}) \tag{4}$$

where $\mathbf{J}^{-1}$ is the inverse of the Jacobian matrix of $\mathbf{F}$.

Method (2) reduces to the classical Newton's method if $\omega = 0$ and $\eta = 1$, and to the relaxed Newton's method if $\omega = 0$ and $\eta \neq 1$. For $\omega < 1$ the algorithm has good convergence and can reach the solution. The ω and η should be selected by tuning—it is a kind of art [6].

3 Iteration Processes

If we take $\mathbf{T}(\mathbf{z}_n) = \mathbf{z}_n + \mathbf{v}_{n+1}$, then (2) takes the following form:

$$\mathbf{z}_{n+1} = \mathbf{T}(\mathbf{z}_n). \tag{5}$$

This type of iteration is known as the Picard iteration [13], and is widely used among others in fixed point theory.

In fixed point theory, we can find many other iteration processes. For example,

1. The Mann iteration [11]:
$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \tag{6}$$
where $\alpha_n \in (0, 1]$ for all $n \in \mathbb{N}$, and for $\alpha_n = 1$ it reduces to the Picard iteration.
2. The Ishikawa iteration [8]:
$$\begin{aligned} \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}(\mathbf{z}_n), \\ \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}(\mathbf{u}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{7}$$
where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$, and we obtain the Mann iteration when $\beta_n = 0$, and the Picard iteration when $\alpha_n = 1$ and $\beta_n = 0$.
3. The Agarwal iteration [1] (or S-iteration):
$$\begin{aligned} \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}(\mathbf{z}_n), \\ \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}(\mathbf{z}_n) + \alpha_n \mathbf{T}(\mathbf{u}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{8}$$
where $\alpha_n \in [0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$, Moreover, when $\alpha_n = 0$, or $\alpha_n = 1$ and $\beta_n = 0$ the S-iteration reduces to the Picard iteration.

For an overview of some other iteration processes and their dependencies, see [5], where 17 different iterations are reviewed.

All the iterations shown so far use one mapping, but in fixed point theory, there are iterations that use several mappings. Examples of this type of iterations are the following:

1. The Das–Debata iteration [4]:
$$\begin{aligned} \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n), \\ \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}_2(\mathbf{u}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{9}$$
where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$. For $\mathbf{T}_1 = \mathbf{T}_2$ the Das–Debata iteration reduces to the Ishikawa iteration.
2. The Khan–Cho–Abbas iteration [10]:
$$\begin{aligned} \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n), \\ \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}_1(\mathbf{z}_n) + \alpha_n \mathbf{T}_2(\mathbf{u}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{10}$$
where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$, when $\mathbf{T}_1 = \mathbf{T}_2$ the equation reduces to the Agarwal iteration.
3. The generalised Agarwal's iteration [10]:
$$\begin{aligned} \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n), \\ \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}_3(\mathbf{z}_n) + \alpha_n \mathbf{T}_2(\mathbf{u}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{11}$$
where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$, moreover when $\mathbf{T}_1 = \mathbf{T}_3$ the equation reduces to the Khan–Cho–Abbas iteration, and the Agarwal iteration is obtained when $\mathbf{T}_1 = \mathbf{T}_2 = \mathbf{T}_3$.

4 The Generalised Iteration

Let us consider the following general iteration scheme:

$$\begin{aligned}
\mathbf{z}_{0,n+1} &= p_{0,0}\mathbf{z}_{i,n} + p_{0,1}\mathbf{T}_{\mathbf{0},\mathbf{0}}(\mathbf{z}_{i,n}),\\
\mathbf{z}_{1,n+1} &= p_{1,0}\mathbf{z}_{i,n} + p_{1,1}\mathbf{T}_{\mathbf{1},\mathbf{0}}(\mathbf{z}_{i,n}) + p_{1,2}\mathbf{z}_{0,n+1} + p_{1,3}\mathbf{T}_{\mathbf{1},\mathbf{1}}(\mathbf{z}_{0,n+1}),\\
\mathbf{z}_{2,n+1} &= p_{2,0}\mathbf{z}_{i,n} + p_{2,1}\mathbf{T}_{\mathbf{2},\mathbf{0}}(\mathbf{z}_{i,n}) + p_{2,2}\mathbf{z}_{0,n+1} + p_{2,3}\mathbf{T}_{\mathbf{2},\mathbf{1}}(\mathbf{z}_{0,n+1}) +\\
&\quad + p_{2,4}\mathbf{z}_{1,n+1} + p_{2,5}\mathbf{T}_{\mathbf{2},\mathbf{2}}(\mathbf{z}_{1,n+1}),\\
&\vdots\\
\mathbf{z}_{i,n+1} &= p_{i,0}\mathbf{z}_{i,n} + p_{i,1}\mathbf{T}_{\mathbf{i},\mathbf{0}}(\mathbf{z}_{i,n}) + p_{i,2}\mathbf{z}_{0,n+1} + ... +\\
&\quad + p_{i,2i}\mathbf{z}_{i-1,n+1} + p_{i,2i+1}\mathbf{T}_{\mathbf{i},\mathbf{i}}(\mathbf{z}_{i-1,n+1}),\ n = 0,1,2,\ldots,
\end{aligned} \tag{12}$$

where $p_{i,k} \in [0,1]$ for $k \in \{0,1,\ldots,2i+1\}$ for all $i \in \mathbb{N}$ and $\sum_{k=0}^{2i+1} p_{i,k} = 1$ for the given $i \in \mathbb{N}$. The $\mathbf{z}_{i,n}$ is the previous position of the particle and the $\mathbf{z}_{i,n+1}$ is the next position of the particle. The $\{\mathbf{z}_{0,n+1}, \mathbf{z}_{1,n+1}, \ldots, \mathbf{z}_{i-1,n+1}\}$ create a set of reference points – it plays a role similar to a swarm in PSO.

Iteration (12) is a general form of the iterations presented in Sec. 3, when the sequences of the parameters are constant:

1. The Mann iteration is obtained for $i = 0$ and $p_{0,0} + p_{0,1} = 1$:

$$\mathbf{z}_{0,n+1} = (1 - p_{0,1})\mathbf{z}_{0,n} + p_{0,1}\mathbf{T}_{\mathbf{0},\mathbf{0}}(\mathbf{z}_{0,n}),\ n = 0,1,2,\ldots, \tag{13}$$

   where $p_{0,1} \in (0,1]$.
2. The Ishikawa iteration is obtained for $i = 1$ and $p_{0,0}+p_{0,1} = 1$, $p_{1,0}+p_{1,3} = 1$, $p_{1,1} = p_{1,2} = 0$, and $\mathbf{T}_{\mathbf{0},\mathbf{0}} = \mathbf{T}_{\mathbf{1},\mathbf{1}}$:

$$\begin{aligned}
\mathbf{z}_{0,n+1} &= (1 - p_{0,1})\mathbf{z}_{1,n} + p_{0,1}\mathbf{T}_{\mathbf{0},\mathbf{0}}(\mathbf{z}_{1,n}),\\
\mathbf{z}_{1,n+1} &= (1 - p_{1,3})\mathbf{z}_{1,n} + p_{1,3}\mathbf{T}_{\mathbf{1},\mathbf{1}}(\mathbf{z}_{0,n+1}),\ n = 0,1,2,\ldots,
\end{aligned} \tag{14}$$

   where $p_{0,1} \in [0,1]$ and $p_{1,3} \in (0,1]$.
3. The Agarwal iteration is obtained for $i = 1$ and $p_{0,0}+p_{0,1} = 1$, $p_{1,1}+p_{1,3} = 1$, $p_{1,0} = p_{1,2} = 0$, and $\mathbf{T}_{\mathbf{0},\mathbf{0}} = \mathbf{T}_{\mathbf{1},\mathbf{0}} = \mathbf{T}_{\mathbf{1},\mathbf{1}}$:

$$\begin{aligned}
\mathbf{z}_{0,n+1} &= (1 - p_{0,1})\mathbf{z}_{i,n} + p_{0,1}\mathbf{T}_{\mathbf{0},\mathbf{0}}(\mathbf{z}_{1,n}),\\
\mathbf{z}_{1,n+1} &= (1 - p_{1,3})\mathbf{T}_{\mathbf{1},\mathbf{0}}(\mathbf{z}_{1,n}) + p_{1,3}\mathbf{T}_{\mathbf{1},\mathbf{1}}(\mathbf{z}_{0,n+1}),\ n = 0,1,2,\ldots,
\end{aligned} \tag{15}$$

   where $p_{0,1} \in [0,1]$ and $p_{1,3} \in (0,1]$.
4. The Das–Debata iteration is obtained for $i = 1$ and $p_{0,0} + p_{0,1} = 1$, $p_{1,0} + p_{1,3} = 1$, $p_{1,1} = p_{1,2} = 0$, and $\mathbf{T}_{\mathbf{0},\mathbf{0}} \neq \mathbf{T}_{\mathbf{1},\mathbf{1}}$:

$$\begin{aligned}
\mathbf{z}_{0,n+1} &= (1 - p_{0,1})\mathbf{z}_{1,n} + p_{0,1}\mathbf{T}_{\mathbf{0},\mathbf{0}}(\mathbf{z}_{1,n}),\\
\mathbf{z}_{1,n+1} &= (1 - p_{1,3})\mathbf{z}_{1,n} + p_{1,3}\mathbf{T}_{\mathbf{1},\mathbf{1}}(\mathbf{z}_{0,n+1}),\ n = 0,1,2,\ldots,
\end{aligned} \tag{16}$$

   where $p_{0,1} \in [0,1]$ and $p_{1,3} \in (0,1]$.

5. For $i = 1$ and $p_{0,0} + p_{0,1} = 1$, $p_{1,1} + p_{1,3} = 1$, $p_{1,0} = p_{1,2} = 0$, and $\mathbf{T_{0,0}} = \mathbf{T_{1,0}} \neq \mathbf{T_{1,1}}$ we obtain the Khan–Cho–Abbas iteration:

$$\begin{aligned} \mathbf{z}_{0,n+1} &= (1 - p_{0,1})\mathbf{z}_{i,n} + p_{0,1}\mathbf{T_{0,0}}(\mathbf{z}_{1,n}), \\ \mathbf{z}_{1,n+1} &= (1 - p_{1,3})\mathbf{T_{1,0}}(\mathbf{z}_{1,n}) + p_{1,3}\mathbf{T_{1,1}}(\mathbf{z}_{0,n+1}),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{17}$$

   where $p_{0,1} \in [0, 1]$ and $p_{1,3} \in (0, 1]$.
6. The generalised Agarwal's iteration is obtained for $i = 1$ and $p_{0,0} + p_{0,1} = 1$, $p_{1,1} + p_{1,3} = 1$, $p_{1,0} = p_{1,2} = 0$, and $\mathbf{T_{0,0}} \neq \mathbf{T_{1,0}} \neq \mathbf{T_{1,1}}$:

$$\begin{aligned} \mathbf{z}_{0,n+1} &= (1 - p_{0,1})\mathbf{z}_{i,n} + p_{0,1}\mathbf{T_{0,0}}(\mathbf{z}_{1,n}), \\ \mathbf{z}_{1,n+1} &= (1 - p_{1,3})\mathbf{T_{1,0}}(\mathbf{z}_{1,n}) + p_{1,3}\mathbf{T_{1,1}}(\mathbf{z}_{0,n+1}),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{18}$$

   where $p_{0,1} \in [0, 1]$ and $p_{1,3} \in (0, 1]$.

Iteration (12) not only reduces to the existing iteration methods, but it can be used to obtain completely new ones. For instance, we can consider the following iteration scheme:

$$\begin{aligned} \mathbf{z}_{0,n+1} &= (1 - p_{0,1})\mathbf{z}_{i,n} + p_{0,1}\mathbf{T_{0,0}}(\mathbf{z}_{2,n}), \\ \mathbf{z}_{1,n+1} &= (1 - p_{1,3})\mathbf{T_{1,0}}(\mathbf{z}_{2,n}) + p_{1,3}\mathbf{T_{1,1}}(\mathbf{z}_{0,n+1}), \\ \mathbf{z}_{2,n+1} &= (1 - p_{2,5})\mathbf{T_{2,1}}(\mathbf{z}_{0,n+1}) + p_{2,5}\mathbf{T_{2,2}}(\mathbf{z}_{1,n+1}),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{19}$$

Similar to the Agarwal's iterations, we can use one or several mappings in this iteration. For one mapping, i.e., $\mathbf{T_{0,0}} = \mathbf{T_{1,0}} = \mathbf{T_{1,1}} = \mathbf{T_{2,1}} = \mathbf{T_{2,2}}$ we will name this scheme as New', and for several mappings, i.e., $\mathbf{T_{0,0}} = \mathbf{T_{1,0}} \neq \mathbf{T_{1,1}} = \mathbf{T_{2,1}} \neq \mathbf{T_{2,2}}$, as New". The computational cost of iterations can be calculated as the number of different mappings of a point in the scheme. The Mann iteration has the same computational cost as Picard's iteration – only one mapping. The Ishikawa, Agarwal, Das–Debata, Khan–Cho–Abbas iterations have two different mappings. The Ishikawa, Agarwal, Das–Debata, Khan–Cho–Abbas have mappings for two different points, so we can consider that the computational cost is the same. The generalized Agarwal iteration and the proposed New 'and New" schemes have three different mappings, so they have the same computational cost.

5 Experimental Setup

In the experiments, we study the case in which $D = 2$, and the PSO-based Newton-like method from Sec. 2 is taken as the operators $T_{i,j}$ in the iteration process from Sec. 4. The experiments relied on the optimisation of the iteration's parameters for two tasks. In the first task, we minimise the number of iterations needed to find the root, whereas, in the second task, we are interested in obtaining graphical patterns with aesthetic features.

Let $\mathbb{C}$ be the field of complex numbers with a complex number $c = x + iy$ where $i = \sqrt{-1}$ and $x, y \in \mathbb{R}$. In the experiments, we want to solve the following non-linear equation

$$p(c) = 0 \tag{20}$$

where $p(c) = c^3 - 1$. This equation can be written in the following form:

$$0 = c^3 - 1 = (x + iy)^3 - 1 = x^3 - 3xy^2 - 1 + (3x^2y - y^3)i. \tag{21}$$

Expression (21) can be transformed into a system of two equations with two variables:

$$\mathbf{F}(x, y) = \begin{bmatrix} f_1(x, y) \\ f_2(x, y) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{0}, \tag{22}$$

where $f_1(x, y) = x^3 - 3xy^2 - 1$, $f_2(x, y) = 3x^2y - y^3$. The set of solutions of this system is the following: $[1, 0]$, $[-0.5, -0.866025]$, $[-0.5, 0.866025]$.

In a similar way, we can transform other commonly used, in the literature, complex polynomial equations into systems of non-linear equations:

$$0 = c^4 - 10c^2 + 9 = x^4 - 6x^2y^2 + y^4 - 10x^2 + 10y^2 + 9 + (4x^3y - 4xy^3 - 20xy)i, \tag{23}$$

where $f_1(x, y) = x^4 - 6x^2y^2 + y^4 - 10x^2 + 10y^2 + 9$, $f_2(x, y) = 4x^3y - 4xy^3 - 20xy$ and the set of solutions of this system is the following: $[-3.0, 0.0]$, $[-1.0, 0.0]$, $[1.0, 0.0]$, $[3.0, 0.0]$;

$$0 = c^5 - c = x^5 - 10x^3y^2 + 5xy^4 - x + (5x^4y - 10x^2y^3 + y^5 - y)i, \tag{24}$$

where $f_1(x, y) = x^5 - 10x^3y^2 + 5xy^4 - x$, $f_2(x, y) = 5x^4y - 10x^2y^3 + y^5 - y$ and the set of solutions of this system is the following: $[-1.0, 0.0]$, $[0.0, -1.0]$, $[0.0, 0.0]$, $[0.0, 1.0]$, $[1.0, 0.0]$;

$$\begin{aligned} 0 = c^6 + 10c^3 - 8 = x^6 - 15x^4y^2 + 15x^2y^4 - y^6 + 10x^3 - 30xy^2 - 8 + \\ + (6x^5y - 20x^3y^3 + 6xy^5 + 30x^2y - 10y^3)i, \end{aligned} \tag{25}$$

where $f_1(x, y) = x^6 - 15x^4y^2 + 15x^2y^4 - y^6 + 10x^3 - 30xy^2 - 8$, $f_2(x, y) = 6x^5y - 20x^3y^3 + 6xy^5 + 30x^2y - 10y^3$ and the set of solutions of this system is the following (approximately):$[-2.207, 0]$, $[-0.453, -0.785]$, $[-0.453, 0.785]$, $[0.906, 0]$, $[1.103, -1.911]$, $[1.103, 1.911]$.

Nowadays, visual analysis is an essential part of modern analysis of the quality of the root-finding methods [12]. To visualise the dynamics of the algorithm's operations, we use Algorithm 1, which is a standard algorithm used in polynomiography [9].

Algorithm 1: Visualisation of algorithm's operations dynamics.

Input: $\mathbf{F}$ – function; $\mathbf{A} \subset \mathbb{R}^D$ – solution space; m – the maximum number of iterations; I_q – iteration (12) with the parameters q; C – colouring function; ε – accuracy

Output: Visualisation of the dynamics

```
foreach z_0 ∈ A do
    i = 0
    v_0 = [0, 0, ..., 0]
    while i ≤ m do
        z_{n+1} = I_q(z_n)
        if ||z_{n+1} − z_n|| < ε then
            break
        i = i + 1
    colour z_0 with C(i)
```

The algorithms used in the experiments were implemented in the C++ programming language. The experiments were conducted on a computer with the Intel Core i7-8750H CPU 2.20GHz processor, 16 GB RAM, NVIDIA GeForce GTX 1060 Mobile and Linux Ubuntu 20.04 LTS. The iterations minimisation is implemented using the simple genetic algorithm from GAlib [14]. Each of the optimised parameters has a 63 bit representation mapped to the decimal phenotype with a given range: $[0, 1]$ or $[0, 1.5]$. The simple genetic algorithm operation parameters are: the probability of the mutation is 1.5%, the cross-over probability is 80%, the population size is 50, and the number of generations is 400. The fitness function is based on Algorithm 1 implemented using OpenCL 2.0 [7]. Implementing the genetic algorithm does not require any further discussion because the GAlib library is elementary to use. Optimisation of the algorithm's operation to obtain aesthetic patterns is based on expert knowledge. The selection of the iteration parameter values is based on the authors' experience, and assessing the aesthetics of the pattern is subjective. The influence of parameters on the algorithm's operation will be discussed in the next section.

To generate the images using Algorithm 1, we used a colour map with $m = 256$ levels (Fig. 1), $\varepsilon = 1.0e{-}2$, image resolution is 800×800 pixels, and the area $\mathbf{A}$ for the considered systems were the following: $[-2, 2]^2$, $[-4, 4] \times [-2, 2]$, $[-2, 2]^2$, $[-2.3, 1.7] \times [-2, 2]$, respectively.

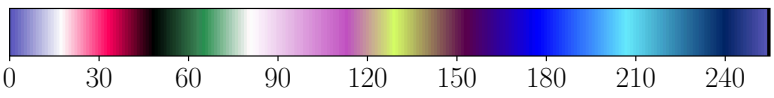


Fig. 1. Colour map used in the experiments.

6 Discussion on the Obtained Results

As we mentioned in the previous section, the selection of iteration parameters is analysed for two purposes: to minimise the number of iterations and to obtain patterns with aesthetic features.

In the tables with the results we use the following abbreviations: A – equation (21), B – equation (23), C – equation (24), D – equation (25), and the letter I denotes the simulation without inertia weight (which means that $\omega = 0$) and the simulation in which it is used by II.

6.1 Minimising the Number of Iterations

The simple genetic algorithm minimises the value of the average number of iterations by selecting appropriate coefficients. Table 1 contains the average iteration values obtained for the analysed algorithms without inertia weight (I) and with their use (II). The values of the optimised coefficients for the algorithms without inertia weight are presented in Tab. 2, and taking into account the inertia weight in Tab. 3. Unambiguous conclusions can be drawn from the data analysis. The inertia weight is of marginal importance. Only the transformation of the particle and the reference point significantly influence the operation of the algorithm. Algorithms are transformed into a sequence of such transformations. The iterations proposed in the article (New', New") increase the number of such transformations to obtain the best results – the lowest average iteration value.

Table 1. The average numbers of iterations of the analysed algorithms for the considered approach minimizing the number of iterations.

Test / Iteration	A		B		C		D	
	I	II	I	II	I	II	I	II
Picard	5.822 7		5.654 8		5.986 3		8.168 1	
Mann	5.820 6	5.792 9	5.654 8	5.617 0	5.985 8	5.676 0	8.115 9	8.116 7
Ishikawa	3.668 6	3.657 2	3.719 2	3.704 9	3.747 8	3.600 4	4.800 4	4.801 5
Agarwal	3.668 7	3.654 5	3.719 3	3.705 6	3.748 2	3.524 3	4.799 8	4.804 8
Das–Debata	3.668 5	3.656 3	3.719 2	3.705 2	3.748 3	3.524 7	4.800 5	4.803 2
Khan–Cho–Abbas	3.669 0	3.654 9	3.719 3	3.705 6	3.748 3	3.523 2	4.800 2	4.804 7
generalised Agarwal	3.668 9	3.655 2	3.719 3	3.707 0	3.748 3	3.522 9	4.799 3	4.817 2
New'	2.914 0	2.869 6	3.108 3	3.061 8	3.012 2	2.795 1	3.703 4	3.704 9
New"	2.914 3	2.850 9	3.108 8	3.013 4	3.012 3	2.799 3	3.704 0	3.727 2

Moreover, taking into account the values of the coefficients selected in the optimization process it can be concluded that the Mann iteration transforms into Picard's iteration. The Ishikawa, Agarwal, Das–Debata, Khan–Cho–Abbas and generalised Agarwal iterations are also transformed into the same form. And, the New' iteration is transformed to the New". These observations confirm the polynomiographs presented in Fig. 2 for iteration with transformations without

Table 2. Values of the optimised coefficients for algorithms without inertia weight for the considered approach minimizing the number of iterations.

Test environment			A		B		C		D	
Iteration	Coefficients		Coefficients		Coefficients		Coefficients		Coefficients	
Picard	$p_{0,1}$	$\eta_{0,0}$	1.000 0	1.000 0	1.000 0	1.000 0	1.000 0	1.000 0	1.000 0	1.000 0
Mann	$p_{0,1}$	$\eta_{0,0}$	0.999 9	0.999 7	1.000 0	1.000 0	1.000 0	1.000 0	0.998 7	0.968 0
Ishikawa	$p_{0,1}$	$\eta_{0,0}$	0.999 8	0.999 8	0.996 3	1.000 0	1.000 0	1.000 0	0.960 6	0.984 8
	$p_{1,3}$		1.000 0		1.000 0		1.000 0		1.000 0	
Agarwal	$p_{0,1}$	$\eta_{0,0}$	0.998 5	0.999 7	0.998 9	1.000 0	0.999 6	0.999 7	0.955 5	0.989 7
	$p_{1,3}$		0.999 7		1.000 0		0.999 5		0.995 0	
Das–Debata	$p_{0,1}$	$\eta_{0,0}$	0.999 4	0.998 3	0.997 5	1.000 0	0.999 6	0.999 5	0.998 7	0.991 4
	$p_{1,3}$	$\eta_{1,1}$	0.999 8	0.999 7	1.000 0	0.998 7	1.000 0	0.999 6	0.999 9	0.935 3
Khan–Cho–	$p_{0,1}$	$\eta_{0,0}$	0.998 3	1.000 0	0.999 4	0.999 9	0.999 6	0.999 7	0.964 4	0.991 3
–Abbas	$p_{1,3}$	$\eta_{1,1}$	0.999 1	0.999 9	1.000 0	0.996 1	0.999 8	0.999 5	0.993 4	0.962 6
generalised	$p_{0,1}$	$\eta_{0,0}$	0.999 7	0.999 0	0.990 1	1.000 0	0.999 5	0.999 6	0.984 3	0.990 4
Agarwal	$p_{1,3}$	$\eta_{1,1}$	0.999 7	0.999 2	1.000 0	0.998 7	0.999 9	0.999 6	0.997 0	0.941 2
		$\eta_{2,2}$		0.583 7		0.559 6		0.705 2		0.890 4
New'	$p_{0,1}$	$\eta_{0,0}$	0.999 9	1.000 0	0.999 9	1.000 0	0.999 7	0.999 9	0.951 8	0.988 5
	$p_{1,3}$		1.000 0		0.999 7		0.999 7		0.974 0	
	$p_{2,5}$		0.995 6		1.000 0		0.998 5		0.987 1	
New"	$p_{0,1}$	$\eta_{0,0}$	0.999 5	0.999 3	0.999 4	0.999 8	1.000 0	0.999 4	0.977 9	0.954 4
	$p_{1,3}$	$\eta_{1,1}$	0.999 7	0.999 7	0.999 9	0.999 9	0.999 7	0.999 9	0.985 0	0.969 6
	$p_{2,5}$	$\eta_{2,2}$	0.998 4	0.999 8	0.999 8	1.000 0	0.999 2	0.999 3	0.986 1	0.993 7

inertia weight and in Fig. 3 with inertia weight. The acceleration constant may take values greater than 1.0. It is possible due to the properties of the analysed test environments. Polynomiographs for the proposed schemes look smoother compared to other methods, this is primarily caused by the improved convergence behaviour.

6.2 Obtaining an Aesthetic Pattern

Table 4 presents the average iteration values for algorithms without inertia weight (I) and with their use (II). The values of the coefficients for the algorithms without inertia weight are presented in Tab. 5, and with the inertia weight in Tab. 6. One way to obtain aesthetic patterns is to vary the number of iterations creating the polynomiograph significantly. It is associated with a considerable extension of the average iteration value. The number of iterations can be differentiated by increasing the inertia weight and decreasing the acceleration constant. Changes in the p parameters allow the control of the particle dynamics in different areas of the image. We can conclude – the greater the number of coefficients, the greater the ability to control particle dynamics.

The generalised form of iteration proposed in the article gives such possibilities. The possibilities of creating patterns with aesthetic features are presented in Fig. 4 and 5. The polynomiographs in Fig. 5 show much greater particle dynamics due to the use of the inertia weight.

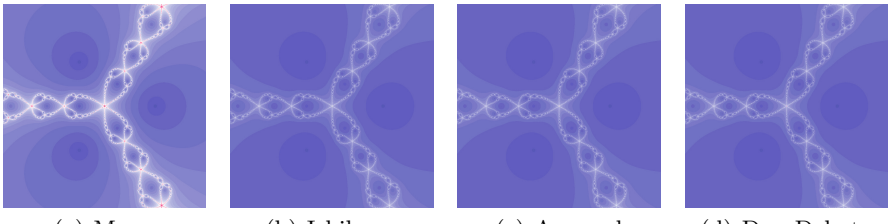
Table 3. Values of the optimised coefficients for algorithms with inertia weight for the considered approach minimizing the number of iterations.

Test environment			A		B		C		D	
Iteration	Coefficients		Coefficients		Coefficients		Coefficients		Coefficients	
Mann	$p_{0,1}$	$\omega_{0,0}$	0.919 0	0.000 3	0.739 8	0.000 4	0.902 6	0.028 8	0.834 0	0.000 3
		$\eta_{0,0}$		1.150 8		1.381 7		1.386 9		1.158 8
Ishikawa	$p_{0,1}$	$\omega_{0,0}$	0.981 9	0.001 4	0.990 4	0.000 2	0.999 6	0.002 4	0.964 3	0.000 1
	$p_{1,3}$	$\eta_{0,0}$	0.996 1	1.079 2	0.999 8	1.019 9	0.971 5	1.172 7	0.999 3	0.981 1
Agarwal	$p_{0,1}$	$\omega_{0,0}$	0.985 2	0.000 0	0.951 3	0.000 1	0.997 4	0.013 0	0.928 5	0.001 0
	$p_{1,3}$	$\eta_{0,0}$	0.916 8	1.105 9	0.988 8	1.031 3	0.740 7	1.499 2	0.986 8	0.997 1
Das–Debata	$p_{0,1}$	$\omega_{0,0}$	0.830 3	0.002 3	0.902 6	0.000 4	0.999 9	0.011 2	0.688 0	0.000 2
	$p_{1,3}$	$\eta_{0,0}$	0.998 7	1.061 5	0.999 8	1.019 1	0.999 7	1.095 1	0.997 9	0.989 6
		$\omega_{1,1}$		0.000 3		0.000 3		0.009 6		0.000 7
		$\eta_{1,1}$		1.275 0		1.097 0		1.499 0		1.348 6
Khan–Cho–	$p_{0,1}$	$\omega_{0,0}$	0.998 9	0.001 5	0.996 9	0.000 2	0.999 3	0.013 8	0.996 6	0.000 1
–Abbas	$p_{1,3}$	$\eta_{0,0}$	0.784 5	1.321 9	0.781 9	1.307 3	0.908 8	1.222 9	0.737 4	1.341 7
		$\omega_{1,1}$		0.000 6		0.000 3		0.006 2		0.002 6
		$\eta_{1,1}$		1.075 8		0.995 8		1.499 3		0.950 0
generalised	$p_{0,1}$	$\omega_{0,0}$	0.712 3	0.001 3	0.748 3	0.000 3	1.000 0	0.005 4	0.651 5	0.000 4
Agarwal	$p_{1,3}$	$\eta_{0,0}$	0.972 4	1.070 7	0.980 1	1.039 2	0.972 5	1.134 3	0.901 6	1.100 6
		$\omega_{1,1}$		0.000 0		0.003 2		0.011 9		0.013 4
		$\eta_{1,1}$		1.491 2		1.351 7		1.498 3		1.455 1
		$\omega_{2,2}$		0.024 2		0.004 6		0.197 8		0.007 1
		$\eta_{2,2}$		1.069 4		0.987 1		1.483 5		0.972 6
New'	$p_{0,1}$	$\omega_{0,0}$	0.994 6	0.000 0	0.999 9	0.000 0	0.998 9	0.001 1	0.950 9	0.000 1
	$p_{1,3}$	$\eta_{0,0}$	0.997 5	1.191 6	0.990 3	1.162 1	0.994 1	1.496 3	0.948 9	0.997 0
	$p_{2,5}$		0.749 4		0.875 4		0.638 8		0.988 6	
New"	$p_{0,1}$	$\omega_{0,0}$	0.993 7	0.006 9	0.996 8	0.032 4	0.975 1	0.000 5	0.999 2	0.026 5
	$p_{1,3}$	$\eta_{0,0}$	0.850 4	1.144 7	0.999 0	0.980 9	0.993 7	1.499 0	0.985 2	0.992 8
	$p_{2,5}$	$\omega_{1,1}$	0.939 0	0.009 7	0.692 5	0.054 7	0.692 3	0.005 0	0.866 8	0.001 5
		$\eta_{1,1}$		1.429 2		1.474 6		1.497 8		0.929 9
		$\omega_{2,2}$		0.003 2		0.000 8		0.003 8		0.000 9
		$\eta_{2,2}$		0.966 6		1.468 0		1.464 6		1.144 1

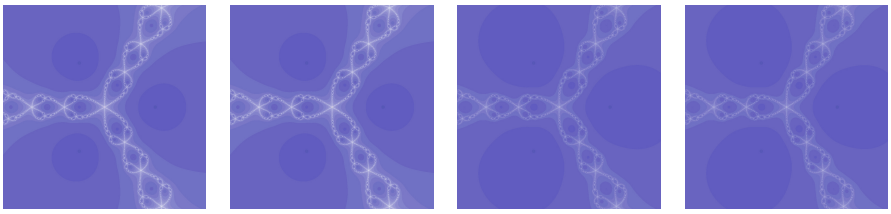
Table 4. The average number of iterations of the analysed algorithms for the considered approach generating aesthetic patterns.

Test / Iteration	A		B		C		D	
	I	*II*	*I*	*II*	*I*	*II*	*I*	*II*
New'	21.26	14.50	45.87	122.41	9.30	9.60	19.83	15.33
New"	6.44	26.42	29.44	58.04	13.19	17.88	10.88	27.07

Even slight changes in the coefficient values can cause large visual changes on the polynomiograph. The coefficients were selected during a series of experiments based on the observation of changes on a polynomiograph. The possibilities of discussing these issues are limited only by the volume of the article.

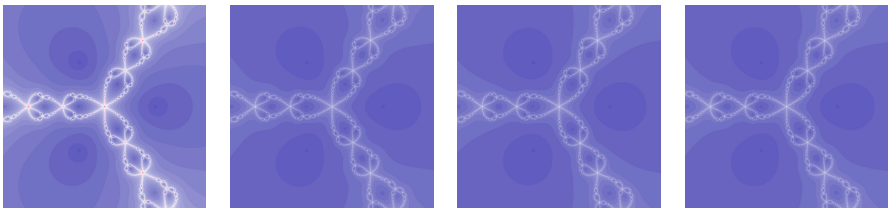


(a) Mann (b) Ishikawa (c) Agarwal (d) Das–Debata

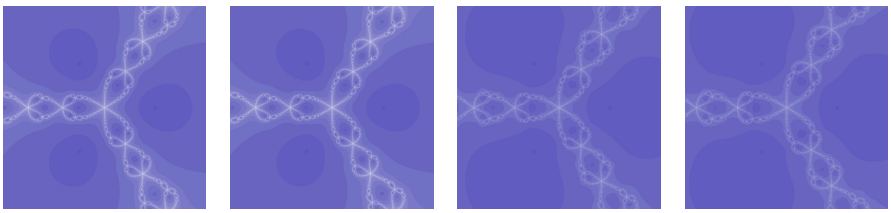


(e) Khan–Cho–Abbas (f) gen. Agarwal (g) New' (h) New"

Fig. 2. Polynomiographs of the iterations without inertia weights.



(a) Mann (b) Ishikawa (c) Agarwal (d) Das–Debata



(e) Khan–Cho–Abbas (f) gen. Agarwal (g) New' (h) New"

Fig. 3. Polynomiographs of the iterations with inertia weights.

7 Conclusions

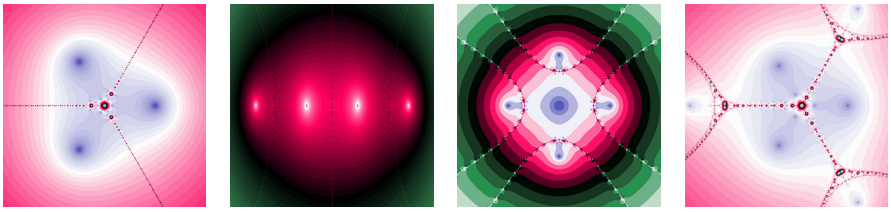
The article proposes a generalised form of iteration. Based on it, a new iteration for one and several mappings is proposed. It was shown that the proposed algorithm enables fast root-finding and creating patterns with aesthetic features. These tasks, due to the algorithm tuning, are contradictory goals. The gener-

Table 5. Values of the selected coefficients for algorithms without inertia weight for the considered approach generating aesthetic patterns.

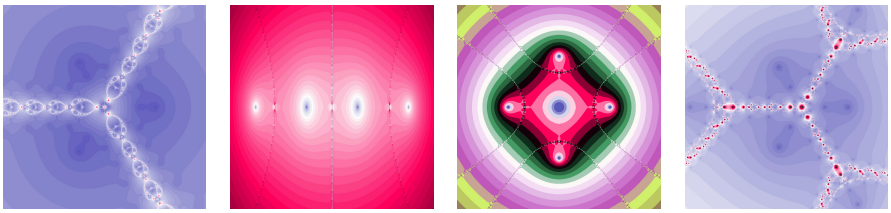
Test environment			A		B		C		D	
Iteration	Coefficients		Coefficients		Coefficients		Coefficients		Coefficients	
New'	$p_{0,1}$	$\eta_{0,0}$	0.50	0.10	0.20	0.10	0.20	0.40	0.10	0.20
	$p_{1,3}$		0.80		0.10		0.50		0.20	
	$p_{2,5}$		0.30		0.10		0.10		0.50	
New"	$p_{0,1}$	$\eta_{0,0}$	0.10	0.10	0.04	0.20	0.50	0.20	0.20	0.20
	$p_{1,3}$	$\eta_{1,1}$	0.20	0.70	0.10	0.15	0.20	0.15	0.10	0.50
	$p_{2,5}$	$\eta_{2,2}$	0.10	0.20	0.10	0.40	0.30	0.40	0.30	0.40

Table 6. Values of the selected coefficients for algorithms with inertia weight for the considered approach generating aesthetic patterns.

Test environment			A		B		C		D	
Iteration	Coefficients		Coefficients		Coefficients		Coefficients		Coefficients	
New'	$p_{0,1}$	$\omega_{0,0}$	0.200 0	0.800 0	0.100 0	0.750 0	0.350 0	0.750 0	0.500 0	0.500 0
	$p_{1,3}$	$\eta_{0,0}$	0.500 0	0.500 0	0.450 0	0.600 0	0.100 0	0.600 0	0.250 0	0.400 0
	$p_{2,5}$		0.400 0		0.400 0		0.350 0		0.100 0	
New"	$p_{0,1}$	$\omega_{0,0}$	0.200 0	0.900 0	0.200 0	0.900 0	0.200 0	0.900 0	0.300 0	0.700 0
	$p_{1,3}$	$\eta_{0,0}$	0.100 0	0.200 0	0.300 0	0.300 0	0.100 0	0.300 0	0.300 0	0.300 0
	$p_{2,5}$	$\omega_{1,1}$	0.100 0	0.900 0	0.100 0	0.900 0	0.200 0	0.900 0	0.400 0	0.800 0
		$\eta_{1,1}$		0.200 0		0.200 0		0.300 0		0.400 0
		$\omega_{2,2}$		0.900 0		0.500 0		0.400 0		0.800 0
		$\eta_{2,2}$		0.100 0		0.100 0		0.200 0		0.300 0



(a) (b) (c) (d)



(e) (f) (g) (h)

Fig. 4. Polynomyographs with aesthetic features obtained using the iterations without inertia weights.

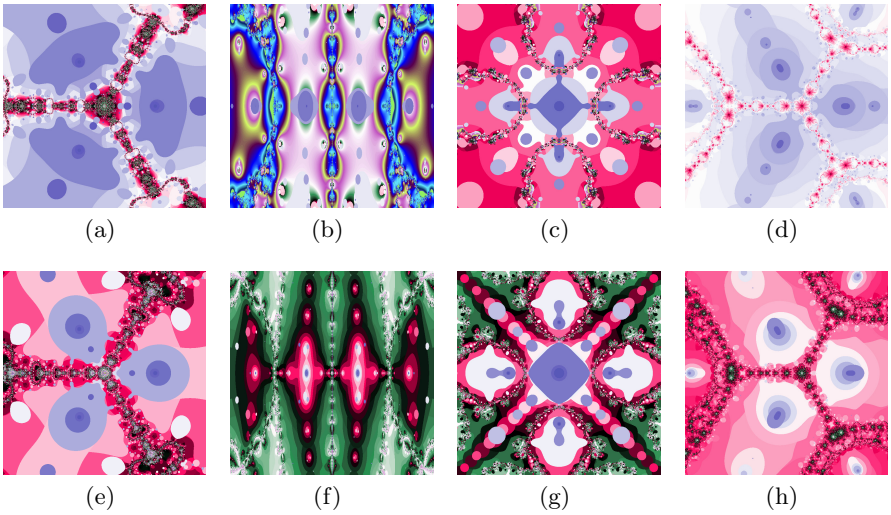


Fig. 5. Polynomyographs with aesthetic features obtained using the iterations with inertia weights.

alised form of iteration will allow for the creation of many new iterations. It can also become the basis for hybridisation with other algorithms.

References

1. Agarwal, R., O'Regan, D., Sahu, D.: Iterative construction of fixed points of nearly asymptotically nonexpansive mappings. Journal of Nonlinear and Convex Analysis **8**(1), 61–79 (2007)
2. Ardelean, G., Cosma, O., Balog, L.: A comparison of some fixed point iteration procedures by using the basins of attraction. Carpathian Journal of Mathematics **32**(3), 277–284 (2016)
3. Cheney, W., Kincaid, D.: Numerical Mathematics and Computing, 6th Edition. Brooks/Cole, Pacific Groove, CA (2007)
4. Das, G., Debata, J.: Fixed points of quasinonexpansive mappings. Indian Journal of Pure and Applied Mathematics **17**(11), 1263–1269 (1986)
5. Gdawiec, K., Kotarski, W.: Polynomiography for the polynomial infinity norm via Kalantari's formula and nonstandard iterations. Applied Mathematics and Computation **307**, 17–30 (2017). https://doi.org/10.1016/j.amc.2017.02.038
6. Gościniak, I., Gdawiec, K.: Visual analysis of dynamics behaviour of an iterative method depending on selected parameters and modifications. Entropy **22**(7), 734 (2020). https://doi.org/10.3390/e22070734
7. Howes, L., Munshi, A.: The OpenCL specification. http://www.khronos.org/registry/OpenCL/specs/opencl-2.0.pdf (2015)
8. Ishikawa, S.: Fixed points by a new iteration method. Proceedings of the American Mathematical Society **44**(1), 147–150 (1974). https://doi.org/10.1090/S0002-9939-1974-0336469-5

9. Kalantari, B.: Polynomial Root-Finding and Polynomiography. World Scientific, Singapore (2009). https://doi.org/10.1142/9789812811837
10. Khan, S., Cho, Y., Abbas, M.: Convergence to common fixed points by a modified iteration process. Journal of Applied Mathematics and Computing **35**(1), 607–616 (2011). https://doi.org/10.1007/s12190-010-0381-z
11. Mann, W.: Mean value methods in iteration. Proceedings of the American Mathematical Society **4**(3), 506–510 (1953). https://doi.org/10.1090/S0002-9939-1953-0054846-3
12. Petković, I., Rančić, L.: Computational geometry as a tool for studying root-finding methods. Filomat **33**(4), 1019–1027 (2019). https://doi.org/10.2298/FIL1904019P
13. Picard, E.: Mémoire sur la théorie des équations aux dérivées partielles et la méthode des approximations successives. Journal de Mathématiques Pures et Appliquées **6**(4), 145–210 (1890)
14. Wall, M.: GAlib, a C++ library of genetic algorithm components. http://lancet.mit.edu/ga/ (2008)