PSO-Based Newton-Like Method and Iteration Processes in the Generation of Artistic Patterns

Ireneusz Gościniak(✉) and Krzysztof Gdawiec

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

Abstract. In artistic pattern generation one can find many different approaches to the generation process. One of such approaches is the use of root finding methods. In this paper, we present a new method of generating artistic patterns with the use of root finding. We modify the classical Newton's method using a Particle Swarm Optimization approach. Moreover, we introduce various iteration processes instead of the standard Picard iteration used in the Newton's method. Presented examples show that using the proposed method we are able to obtain very interesting and diverse patterns that could have an artistic application, e.g., in texture generation, tapestry or textile design etc.

Keywords: Generative art · Root finding · Dynamics · Iterations Visualization

1 Introduction

One of the most elusive goals in computer aided design is artistic design and pattern generation. This involves diverse aspects: analysis, creativity and development [1]. A designer has to deal with all of these aspects in order to obtain an interesting pattern, which later could be used in jewellery design, carpet design, as a texture etc. Usually the most work during the design stage is carried out by a designer manually, especially in the cases in which the designed pattern should contain some unique, unrepeatable artistic features. Therefore, it is highly useful to develop methods (e.g. automatic, semi-automatic) that will assist pattern generation, and will make the whole process easier.

In the literature we can find many artistic pattern generation methods. They involve different approaches to the generation process, e.g., fractals [2], neural networks [3], shape grammars [4] etc. One of the popular methods of generating artistic patterns is the use of root finding methods and visualization of their behaviour. This method is called polynomiography [5] and images created by it are called polynomiographs. In the generation process one can use a single root finding method [6] or a combination of them [7].

G. Bebis et al. (Eds.): ISVC 2018, LNCS 11241, pp. 47–56, 2018.
https://doi.org/10.1007/978-3-030-03801-4_5

Evolutionary algorithms are algorithms for optimization – particularly for solving the minimization problem, but they can also be adopted to root finding. One of very popular algorithms in this group is the Particle Swarm Optimization (PSO). The particle movement in PSO can be described by the following equation:

$$\mathbf{z}_i' = \mathbf{z}_i + \mathbf{v}_i', \tag{1}$$

where $\mathbf{z}_i'$ – the current position of the ith particle in a D dimensional environment, $\mathbf{z}_i$ – the previous position of the ith particle, $\mathbf{v}_i'$ – the current velocity of the ith particle in a D dimensional environment that is given by the following formula:

$$\mathbf{v}_i' = \omega\mathbf{v}_i + \eta_1 r_1(\mathbf{z}_{pbest\ i} - \mathbf{z}_i) + \eta_2 r_2(\mathbf{z}_{gbest} - \mathbf{z}_i), \tag{2}$$

where $\mathbf{v}_i$ – the previous velocity of the ith particle, ω – inertia weight ($\omega \in [0,1]$), η_1, η_2 – acceleration constants ($\eta_1, \eta_2 \in (0,1]$), r_1, r_2 – random numbers generated uniformly in the $[0,1]$ interval, $\mathbf{z}_{pbest\ i}$ – the best position of the ith particle, $\mathbf{z}_{gbest}$ – the global best position of the particles. The best position and the global best position of particles are updated in each iteration. The particle behaviour depends on inertia weight (ω) and acceleration constants (η_1, η_2). The inertia weight helps particle to escape from a not promising area and acceleration constants direct the particle to an extreme – the values of the parameters are selected during the tuning process.

The behaviour of particles can be very complicated in evolutionary algorithms [8,9]. So the use of this algorithms group – especially the PSO algorithm – can give rise to new artistic patterns. Thus, in this paper we propose modification of the Newton's method using the PSO approach and various iteration processes.

The rest of the paper is organized as follows. Section 2 introduces a root finding algorithm that is based on the Newton method and the PSO approach. Next, Sect. 3 introduces iteration processes known in literature. Then, in Sect. 4 an algorithm for creating artistic images is presented. Some examples of patterns obtained with the proposed algorithm are presented in Sect. 5. Finally, Sect. 6 gives short concluding remarks.

2 PSO-Based Newton Method

The Newton method is one of methods to solve a system of D non-linear equations with D variables [10]. Let $f_1, f_2, \ldots, f_D : \mathbb{R}^D \to \mathbb{R}$ and let

$$\mathbf{F}(z^1, z^2, \ldots, z^D) = \begin{bmatrix} f_1(z^1, z^2, \ldots, z^D) \\ f_2(z^1, z^2, \ldots, z^D) \\ \vdots \\ f_D(z^1, z^2, \ldots, z^D) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0}. \tag{3}$$

Assume that $\mathbf{F} : \mathbb{R}^D \to \mathbb{R}^D$ is a continuous function which has continuous first partial derivatives. Thus, to solve the equation $\mathbf{F}(\mathbf{z}) = \mathbf{0}$, where

$\mathbf{z} = [z^1, z^2, \ldots, z^D]$, using the Newton method we select a starting point $\mathbf{z}_0 = [z_0^1, z_0^2, \ldots, z_0^D]$ and then use the following iterative formula:

$$\mathbf{z}_{n+1} = \mathbf{z}_n - \mathbf{J}^{-1}(\mathbf{z}_n)\mathbf{F}(\mathbf{z_n}) \quad n = 0, 1, 2, \ldots, \tag{4}$$

where

$$\mathbf{J}(\mathbf{z}) = \begin{bmatrix} \frac{\partial f_1}{\partial z_1}(\mathbf{z}) & \frac{\partial f_1}{\partial z_2}(\mathbf{z}) & \ldots & \frac{\partial f_1}{\partial z_D}(\mathbf{z}) \\ \frac{\partial f_2}{\partial z_1}(\mathbf{z}) & \frac{\partial f_2}{\partial z_2}(\mathbf{z}) & \ldots & \frac{\partial f_2}{\partial z_D}(\mathbf{z}) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_D}{\partial z_1}(\mathbf{z}) & \frac{\partial f_D}{\partial z_2}(\mathbf{z}) & \ldots & \frac{\partial f_D}{\partial z_D}(\mathbf{z}) \end{bmatrix} \tag{5}$$

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

Introducing $\mathbf{N}(\mathbf{z}) = -\mathbf{J}^{-1}(\mathbf{z})\mathbf{F}(\mathbf{z})$ the Newton method can be represented in the following form:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{N}(\mathbf{z}_n), \quad n = 0, 1, 2, \ldots. \tag{6}$$

To solve (3) the following PSO approach can be used:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{v}_{n+1}, \tag{7}$$

where $\mathbf{z}_0 \in \mathbb{R}^D$ is a starting position, $\mathbf{v}_0 = [0, 0, \ldots, 0]$ is a starting velocity, $\mathbf{v}_{n+1}$ is the current velocity of particle, $\mathbf{z}_n$ is the previous position of particle. The algorithm sums the position of the particle $\mathbf{z}_n$ with its current velocity $\mathbf{v}_{n+1}$. The current velocity of the particle is determined by the inertia weight and the acceleration constants:

$$\mathbf{v}_{n+1} = \omega\mathbf{v}_n + \eta\mathbf{N}(\mathbf{z}_n), \tag{8}$$

where $\mathbf{v}_n$ – the previous velocity of particle, $\omega \in [0, 1)$ – inertia weight, $\eta \in (0, 1]$ – acceleration constant.

The inertia weight (ω) and the acceleration constant (η) selection allows to change particle dynamics, which in consequence creates different patterns.

3 Iteration Processes

The Picard iteration is widely used in computational tasks which are based on iterative processes. This iteration has the following form

$$\mathbf{z}_{n+1} = \mathbf{T}(\mathbf{z}_n). \tag{9}$$

Let us notice that (7) uses the Picard iteration, where $\mathbf{T} : \mathbb{R}^D \to \mathbb{R}^D$ is given by the following formula

$$\mathbf{T}(\mathbf{z}_n) = \mathbf{z}_n + \mathbf{v}_{n+1}. \tag{10}$$

In the literature there exist many other types of iterations. The three most widely used iterations are the following:

1. The Mann iteration [11]:

$$\mathbf{z}_{n+1} = (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \tag{11}$$

where $\alpha_n \in (0, 1]$ for all $n \in \mathbb{N}$. The Mann iteration for $\alpha_n = 1$ reduces to the Picard iteration.
2. The Ishikawa iteration [12]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{12}$$

where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$. The Ishikawa iteration reduces to the Mann iteration when $\beta_n = 0$ and to the Picard iteration when $\alpha_n = 1$, $\beta_n = 0$ for all $n \in \mathbb{N}$.
3. The Agarwal iteration [13] (S-iteration):

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}(\mathbf{z}_n) + \alpha_n \mathbf{T}(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{13}$$

where $\alpha_n, \beta_n \in [0, 1]$ for all $n \in \mathbb{N}$. The S-iteration reduces to the Picard iteration when $\alpha_n = 0$, or $\alpha_n = 1$ and $\beta_n = 0$.

A review of various iteration processes and their dependencies can be found in [14].

The introduced so far iterations used only one mapping $\mathbf{T}$, but in the literature we can find also the use of iterations that use more than one mapping [7]. The most basic iteration of this type are the following:

1. The Das-Debata iteration [15]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{z}_n + \alpha_n \mathbf{T}_2(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{14}$$

where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$. The Das-Debata iteration for $\mathbf{T}_1 = \mathbf{T}_2$ reduces to the Ishikawa iteration.
2. The Khan-Cho-Abbas iteration [16]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}_1(\mathbf{z}_n) + \alpha_n \mathbf{T}_2(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{15}$$

where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$. The Khan-Cho-Abbas iteration reduces to the Agarwal iteration when $\mathbf{T}_1 = \mathbf{T}_2$.
3. The generalized Agarwal's iteration [16]:

$$\begin{aligned} \mathbf{z}_{n+1} &= (1-\alpha_n)\mathbf{T}_3(\mathbf{z}_n) + \alpha_n \mathbf{T}_2(\mathbf{u}_n), \\ \mathbf{u}_n &= (1-\beta_n)\mathbf{z}_n + \beta_n \mathbf{T}_1(\mathbf{z}_n),\ n = 0, 1, 2, \ldots, \end{aligned} \tag{16}$$

where $\alpha_n \in (0, 1]$ and $\beta_n \in [0, 1]$ for all $n \in \mathbb{N}$. The generalized Agarwal iteration reduces to the Khan-Cho-Abbas iteration when $\mathbf{T}_1 = \mathbf{T}_3$ and to the Agarwal iteration when $\mathbf{T}_1 = \mathbf{T}_2 = \mathbf{T}_3$.

4 Artistic Patterns Generation Method

To generate the artistic patterns we use Algorithm 1, which is very similar to the polynomiography. In the algorithm one of the iteration methods I_q presented in Sec. 3 is selected. Moreover, we select parameters ω, η for a single mapping $\mathbf{T}$ or $\omega_1, \omega_2, \omega_3$ and η_1, η_2, η_3 for $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3$ (depending on the chosen iteration). The maximum number of iterations m which algorithm should make, accuracy of the computations ε and a colouring function $C : \mathbb{N} \to \{0, 1, \ldots, 255\}^3$ are also chosen. Then, for each $\mathbf{z}_0$ in the considered area $\mathbf{A}$ we iterate it using the chosen iteration and mappings. The iterations of the algorithm proceed till the convergence criterion:

$$\|\mathbf{F}(\mathbf{z}_n)\| < \varepsilon \tag{17}$$

is satisfied or the maximum number of iterations is reached. A colour corresponding to the performed number of iterations is assigned to $\mathbf{z}_0$ using colouring function C.

Algorithm 1. Artistic Patterns Generation Method

Input: $\mathbf{F}$ – function, $\mathbf{A} \subset \mathbb{R}^D$ – solution space, m – the maximum number of iterations, I_q – iteration method, $q \in [0, 1]^N$ – parameters of the iteration I_q, ω, ω_1, ω_2, ω_3, η, η_1, η_1, η_2, η_3 – parameters defining functions $\mathbf{T}$, $\mathbf{T}_1$, $\mathbf{T}_2$, $\mathbf{T}_3$, C – colouring function, ε – accuracy

Output: visualization of the dynamics

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

The solution space $\mathbf{A}$ is defined in a D-dimensional space, thus the algorithm returns patterns in this space. For $D = 2$, a single image is obtained. When $D > 2$ cross section of $\mathbf{A}$ with a two-dimensional plane for visualization can be made.

5 Examples

In this section we present some examples obtained with the proposed method.

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 examples we will use the following complex polynomials:

1. $p_1(c) = c^3 - 1 = x^3 - 3xy^2 - 1 + (3x^2y - y^3)i$
   the roots: 1, $-0.5 - 0.866025i$, $-0.5 + 0.866025i$,
2. $p_2(c) = c^4 - 10c^2 + 9 = x^4 - 6x^2y^2 + y^4 - 10x^2 + 10y^2 + 9 + (4x^3y - 4xy^3 - 20xy)i$
   the roots: -3, -1, 1, 3,
3. $p_3(c) = c^5 - c = x^5 - 10x^3y^2 + 5xy^4 - x + (5x^4y - 10x^2y^3 + y^5 - y)i$
   the roots: -1, $-i$, 0, i, 1,
4. $p_4(c) = c^6 + 10c^3 - 8 = x^6 - 15x^4y^2 + 15x^2y^4 - y^6 + 10x^3 - 30xy^2 - 8 + (6x^5y - 20x^3y^3 + 6xy^5 + 30x^2y - 10y^3)i$
   the roots: -2.207, $-0.453 - 0.785i$, $-0.453 + 0.785i$, 0.906, $1.103 - 1.911i$, $1.103 + 1.911i$.

Having a complex polynomial equation $p(c) = 0$ we can transform it into a system of two equations with two real variables, i.e.,

$$\mathbf{F}(x, y) = \begin{bmatrix} \Re(p(x + iy)) \\ \Im(p(x + iy)) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{0}, \tag{18}$$

where $\Re(c), \Im(c)$ denote the real and imaginary part of a complex number c, respectively.

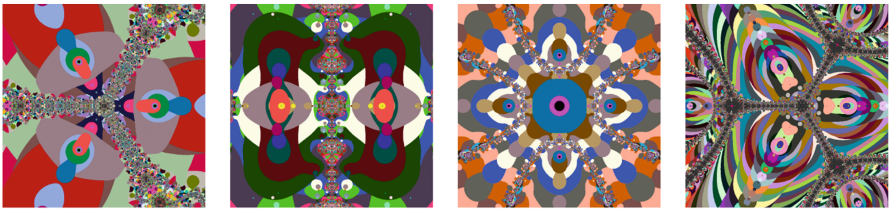
In the examples the same colourmap will be used, which is presented in Fig. 1. The other common parameters used in the examples are the following: $m = 128$, $\varepsilon = 0.1$, image resolution 800×800 pixels. The areas depend on the polynomial and are the following: $\mathbf{A}_1 = [-2.0, 2.0]^2$, $\mathbf{A}_2 = [-4.0, 4.0] \times [-2.0, 2.0]$, $\mathbf{A}_3 = [-2.0, 2.0]^2$, $\mathbf{A}_4 = [-2.3, 1.7] \times [-2.0, 2.0]$.

Fig. 1. Colour map used in the examples

The particles behaviour (dynamics) depends on the acceleration constant (η) and inertia weight (ω). The increase in values of acceleration constant and inertia weight increases the number of image details (increases the image dynamics). Moreover, also the parameters used in the various iteration processes influence particle's dynamics.

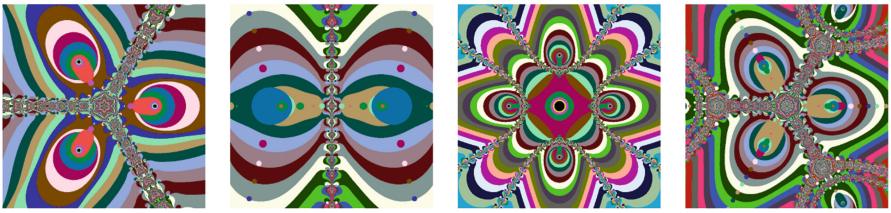
Examples of patterns generated with the use of the Picard iteration for the four considered polynomials and different values of ω and η are presented in Fig. 2. The patterns for the Mann iteration for the considered test functions are presented in Fig. 3. The same values of the parameters were used to create these patterns. We can observe that the obtained patterns have similar features. Comparing the images obtained with the Mann iteration with the ones obtained with the Picard iteration we see that the shapes of the patterns have change in a significant way. The most noticeable change is observed in case of the second polynomial (Fig. 2(b) and Fig. 3(b)).

The patterns for the Ishikawa iteration for the considered test functions are presented in Fig. 4. Images (a) and (b) were generated using $\omega = 0.7$ and $\eta = 0.3$,



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

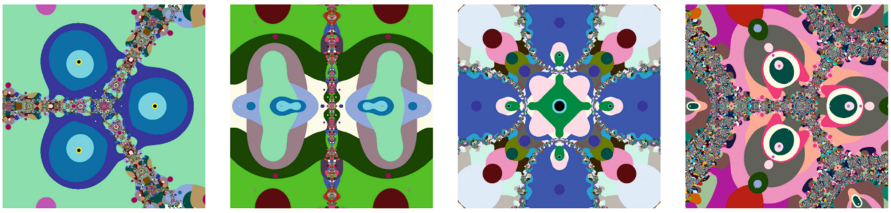
Fig. 2. Patterns generated with the Picard iteration: (a) $\omega = 0.9$, $\eta = 0.1$; (b) $\omega = 0.6$, $\eta = 0.9$; (c) $\omega = 0.4$, $\eta = 0.5$; (d) $\omega = 0.9$, $\eta = 0.03$



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

Fig. 3. Patterns generated with the Mann iteration for $\alpha = 0.3$, $\omega = 0.5$, $\eta = 0.6$

whereas the images (c) and (d) were generated using $\omega = 0.8$ and $\eta = 0.2$. The obtained patterns have similar dynamics – the increase in the inertia weight can be compensated by the decrease of the acceleration constant. The introduction in the iteration process of the second step and in consequence adding a second parameter (β) increases the possibilities of dynamics control. From the obtained images we see that the shapes of the patterns have changed in a significant way comparing to the patterns obtained with the Picard and Mann iterations.

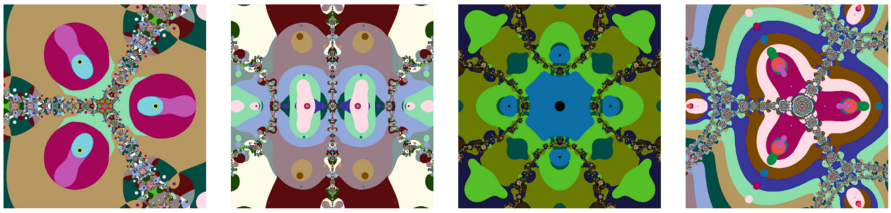


(a) (b) (c) (d)

Fig. 4. Patterns generated with the Ishikawa iteration for $\alpha = 0.9$, $\beta = 0.9$ and: (a), (b) $\omega = 0.7$, $\eta = 0.3$; (c), (d) $\omega = 0.8$, $\eta = 0.2$

The patterns generated with the last iteration that uses only one function – the Agarwal iteration – are presented in Fig. 5. In this iteration the function

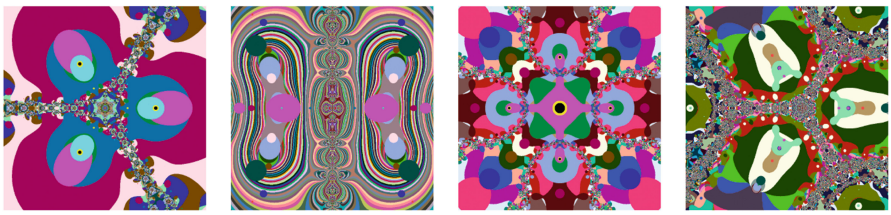
is evaluated three times. This gives more possibilities to control the dynamics and in consequence the shape of the pattern. The impact of the α parameter on dynamics seems to be smaller.



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

Fig. 5. Patterns generated with the Agarwal iteration for: (a) $\alpha = 0.7$, $\beta = 0.5$, $\omega = 0.7$, $\eta = 0.3$; (b) $\alpha = 0.5$, $\beta = 0.9$, $\omega = 0.6$, $\eta = 0.6$; (c) $\alpha = 0.9$, $\beta = 0.9$, $\omega = 0.3$, $\eta = 0.9$; (d) $\alpha = 0.9$, $\beta = 0.5$, $\omega = 0.3$, $\eta = 0.3$

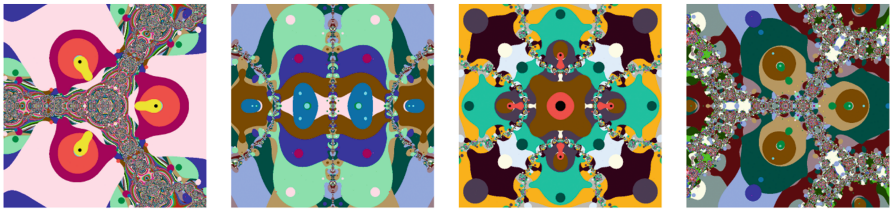
In the next examples pattens generated with the iterations that use more than one function will be presented. We start with the Das-Debata iteration. Patterns generated with this iteration are presented in Fig. 6. Patterns were generated for different values of the parameters. Changes in the value of the α parameter more strongly affect the dynamics change than the changes in the value of parameter β. Moreover, we can observe more complex change of patterns' shape comparing to the iterations with one function.



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

Fig. 6. Patterns generated with the Das-Debata iteration for: (a) $\alpha = 0.7$, $\beta = 0.9$, $\omega_1 = 0.5$, $\eta_1 = 0.6$, $\omega_2 = 0.5$, $\eta_2 = 0.2$; (b) $\alpha = 0.5$, $\beta = 0.9$, $\omega_1 = 0.6$, $\eta_1 = 0.5$, $\omega_2 = 0.9$, $\eta_2 = 0.3$; (c) $\alpha = 0.9$, $\beta = 0.6$, $\omega_1 = 0.7$, $\eta_1 = 0.3$, $\omega_2 = 0.9$, $\eta_2 = 0.2$; (d) $\alpha = 0.6$, $\beta = 0.8$, $\omega_1 = 0.9$, $\eta_1 = 0.3$, $\omega_2 = 0.5$, $\eta_2 = 0.2$

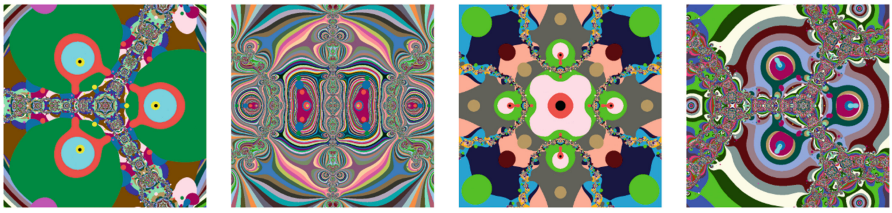
The patterns generated with the Khan-Cho-Abas iteration for the considered test functions are presented in Fig. 7. Similarly to the Das-Debata iteration the change of patterns' shape is very complex. Moreover, its use gives more possibilities to obtain diverse patterns.



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

Fig. 7. Patterns generated with the Khan-Cho-Abas iteration for: (a) $\alpha = 0.5$, $\beta = 0.9$, $\omega_1 = 0.6$, $\eta_1 = 0.6$, $\omega_2 = 0.8$, $\eta_2 = 0.8$, $\omega_3 = 0.6$, $\eta_3 = 0.6$; (b) $\alpha = 0.7$, $\beta = 0.5$, $\omega_1 = 0.7$, $\eta_1 = 0.3$, $\omega_2 = 0.3$, $\eta_2 = 0.7$, $\omega_3 = 0.7$, $\eta_3 = 0.3$; (c) $\alpha = 0.9$, $\beta = 0.5$, $\omega_1 = 0.3$, $\eta_1 = 0.3$, $\omega_2 = 0.6$, $\eta_2 = 0.6$, $\omega_3 = 0.3$, $\eta_3 = 0.3$; (d) $\alpha = 0.9$, $\beta = 0.5$, $\omega_1 = 0.3$, $\eta_1 = 0.3$, $\omega_2 = 0.6$, $\eta_2 = 0.6$, $\omega_3 = 0.3$, $\eta_3 = 0.3$

In the last example – Fig. 8 – patterns generated using the generalized Agarwal iteration are presented. The possibility of independent setting of all parameters of the algorithm gives the greatest possibilities to control the dynamics of the created pattern. It gives the possibility to obtain differentiated image features.



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

Fig. 8. Patterns generated with the generalized Agarwal iteration for: (a) $\alpha = 0.5$, $\beta = 0.7$, $\omega_1 = 0.9$, $\eta_1 = 0.3$, $\omega_2 = 0.3$, $\eta_2 = 0.9$, $\omega_3 = 0.5$, $\eta_3 = 0.3$; (b) $\alpha = 0.9$, $\beta = 0.9$, $\omega_1 = 0.3$, $\eta_1 = 0.9$, $\omega_2 = 0.9$, $\eta_2 = 0.3$, $\omega_3 = 0.1$, $\eta_3 = 0.9$; (c) $\alpha = 0.7$, $\beta = 0.7$, $\omega_1 = 0.7$, $\eta_1 = 0.7$, $\omega_2 = 0.3$, $\eta_2 = 0.3$, $\omega_3 = 0.5$, $\eta_3 = 0.5$; (d) $\alpha = 0.7$, $\beta = 0.5$, $\omega_1 = 0.7$, $\eta_1 = 0.3$, $\omega_2 = 0.3$, $\eta_2 = 0.7$, $\omega_3 = 0.9$, $\eta_3 = 0.3$

6 Conclusions

In this paper, we presented a modification of the Newton's method using the PSO approach and various iteration processes. Moreover, we proposed artistic patterns generation method that is based on the introduced PSO-based Newton's method. The presented examples showed that using the proposed method we are able to obtain very interesting and diverse artistic patterns.

References

1. Wannarumon, S., Unnanon, K., Bohez, E.: Intelligent computer system for jewelry design support. Comput. Aided Des. Appl. **1**(1–4), 551–558 (2004)
2. Soo, S., Yu, K., Chiu, W.: Modeling and fabrication of artistic products based on IFS fractal representation. Comput. Aided Des. **38**(7), 755–769 (2006)
3. Setti, R.: Generative dreams from deep belief networks. In: Soddu, C., Colabella, E. (eds.) Generative Art 2015: Proceeding of XVIII Generative Art Conference, pp. 260–273. Domus Argenia Publisher, Milan (2015)
4. Jia, C., Ming-Xi, T.: Integrating shape grammars into a generative system for Zhuang ethnic embroidery design exploration. Comput. Aided Des. **45**(3), 591–604 (2013)
5. Kalantari, B.: Polynomiography and applications in art, education and science. Comput. Graph. **28**(3), 417–430 (2004)
6. Kalantari, B.: Polynomial Root-Finding and Polynomiography. World Scientific, Singapore (2009)
7. Gdawiec, K.: Fractal patterns from the dynamics of combined polynomial root finding methods. Nonlinear Dyn. **90**(4), 2457–2479 (2017)
8. Gosciniak, I.: Immune algorithm in non-stationary optimization task. In: 2008 International Conference on Computational Intelligence for Modelling Control Automation, pp. 750–755, December 2008
9. Gosciniak, I.: Discussion on semi-immune algorithm behaviour based on fractal analysis. Soft Comput. **21**(14), 3945–3956 (2017)
10. Cheney, W., Kincaid, D.: Numerical Mathematics and Computing, 6th edn. Brooks/Cole, Pacific Grove (2007)
11. Mann, W.: Mean value methods in iteration. Proc. Am. Math. Soc. **4**(3), 506–510 (1953)
12. Ishikawa, S.: Fixed points by a new iteration method. Proc. Am. Math. Soc. **44**(1), 147–150 (1974)
13. Agarwal, R., O'Regan, D., Sahu, D.: Iterative construction of fixed points of nearly asymptotically nonexpansive mappings. J. Nonlinear Convex Anal. **8**(1), 61–79 (2007)
14. Gdawiec, K., Kotarski, W.: Polynomiography for the polynomial infinity norm via Kalantari's formula and nonstandard iterations. Appl. Math. Comput. **307**, 17–30 (2017)
15. Das, G., Debata, J.: Fixed points of quasi-nonexpansive mappings. Indian J. Pure Appl. Math. **17**(11), 1263–1269 (1986)
16. Khan, S., Cho, Y., Abbas, M.: Convergence to common fixed points by a modified iteration process. J. Appl. Math. Comput. **35**(1), 607–616 (2011)