20

Particle Swarm Optimization – Modifications and Application

Adam Slowik
Department of Electronics and Computer Science
Koszalin University of Technology, Koszalin, Poland

CONTENTS

20.1 Introduction

The particle swarm optimization (PSO) [1] algorithm and ant colony optimization (ACO) [2] algorithm are the two first algorithms which began the new era of research called swarm intelligence [7]. The PSO algorithm is a global optimization technique which is inspired by the cooperation behavior (social behavior) of flocks of birds. This algorithm has many common features with evolutionary computing techniques [8]. The PSO algorithm also operates on a randomly created population (called a swarm) of potential solutions (called particles). The optimal solution searches by creating subsequent generations of solutions. However, the genetic operators such as crossover and mutation do not exist in this algorithm. In the PSO algorithm the particles move towards the actual optimal solution/solutions in the search space. The main idea of optimization using the PSO algorithm is based on changes of velocity (acceleration) for each particle moving toward the *Pbest* position and *Lbest* position for the so-called local version of the PSO algorithm or toward the *Pbest* position and *Gbest* position for the so-called global version of the PSO algorithm. The acceleration is randomly modified independently for each direction. This chapter is organized as follows. In Section 20.2 a short introduction to the PSO algorithm is given. The global version of the PSO algorithm (GPSO) and local version of the PSO algorithm (LPSO) are presented and discussed in detail. In Section 20.3, some selected modifications of the PSO algorithm such as velocity clamping, inertia weight, constriction coefficient and acceleration coefficient are described. In Section 20.4, the application of the PSO algorithm to the design of IIR digital filters with non-standard amplitude characteristics is demonstrated.

20.2 Original PSO algorithm in brief

The original PSO algorithm in global version (GPSO) and local version (LPSO) can be presented using pseudo-code in Algorithm 23. The code which is used only for the local version of the algorithm is underlined with a dotted line, and the code which is only for the global version is underlined with a solid line.

20.2.1 Description of the original PSO algorithm

In Algorithm 23 the pseudo-code of the PSO algorithm was presented. Now in this section we will discuss this algorithm in detail. Before we start, we should determine the $D-th$ dimensional objective function $OF(.)$ which is carried out in step 1. In step 2, we determine the range of the variability $[P_{i,j}^{min}, P_{i,j}^{max}]$ for each $j-th$ decision variable ($j \in [1, D]$). In step 3, we determine such PSO algorithm parameters as number of particles in the swarm (P), values of learning factors (c_1 and c_2), and for the LPSO algorithm only we must additionally determine the number of returned results ($Re \in [1, N]$), the type of the neighborhood topology and the number of the nearest neighbors (Ne). In step 4, we randomly create the swarm P which consists of N particles. Each particle P_i is a $D-th$ dimensional vector and is presented as follows:

$$P_i = \{P_{i,1}, P_{i,2}, P_{i,3}, ..., P_{i,D-1}, P_{i,D}\}$$

where: $P_{i,1}, P_{i,2}$, and so on, are the values of the particular decision variables (from the previously determined range $\left[P_j^{min}, P_j^{max}\right]$).

Algorithm 23 Pseudo-code of the original PSO.

```
1: determine the $D-th$ dimensional objective function $OF(.)$
2: determine the range of variability for each $j-th$ dimension $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$
3: determine the PSO algorithm parameter values such as $N$ – number of particles in the swarm, $c_1$, $c_2$ – learning factor, number of returned results $Re \in [1, N]$, the topology and number of nearest neighbors $Ne$
4: randomly create swarm $P$ which consists of $N$ particles (each particle is a $D$-dimensional vector)
5: create the $D$-dimensional $Gbest$ vector
6: for each $i$-th particle $P_i$ from swarm $P$ do
7:     create a $Pbest_i$ particle equal to $i$-th particle $P_i$
8:     create a $Lbest_i$ particle for each particle $P_i$
9:     create a velocity $V_i$ for each particle $P_i$
10:    evaluate the particle $P_i$ using $OF(.)$ function
11: end for
12: assign the best particle $P_i$ to the $Gbest$; for each particle $Lbest_i$ assign the best particle among $Ne$ nearest neighbors of particle $P_i$
13: while termination condition not met do
14:     for each $i$ particle in the swarm $P$ do
15:         for each $j$ dimension do
16:             update the velocity $V_{i,j}$ using formula
17:             $V_{i,j} = V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Gbest_j - P_{i,j})$
18:             $V_{i,j} = V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Lbest_{i,j} - P_{i,j})$
19:             update the particle $P_{i,j}$ using formula $P_{i,j} = P_{i,j} + V_{i,j}$
20:             check if newly created value $P_{i,j}$ is within the range $\left[P_{i,j}^{min}, P_{i,j}^{max}\right]$ if not correct it
21:         end for
22:         evaluate the particle $P_i$ using $OF(.)$ function
23:         if $OF(P_i)$ is better than $OF(Pbest_i)$ then
24:             assign the particle $P_i$ to the particle $Pbest_i$
25:         end if
26:         select the best particle among $Ne$ nearest neighbor of particle $P_i$ and assign it to particle $T$
27:         if $OF(T)$ is better than $OF(Lbest_i)$ then
28:             assign the particle $T$ to the particle $Lbest_i$
29:         end if
30:     end for
31:     select the best particle from swarm $P$ and assign it to particle $T$
32:     if $OF(T)$ is better than $OF(Gbest)$ then
33:         assign the particle $T$ to the particle $Gbest$
34:     end if
35: end while
36: return the $Gbest$ as a result; select $Re$ the best particles among $Lbest$ particles and return these particles as a result
```

In step 5, we create the $D-th$ dimensional particle $Gbest$. This step is only executed in the GPSO algorithm. In the particle $Gbest$ the best global solution will be stored. Next (step 6), we create a $D-th$ dimensional particle $Pbest_i$ for each particle P_i. At the start the values in the particle $Pbest_i$ are the same as they are in particle P_i (see step 7). In the next generations of the PSO algorithm, in the particle $Pbest_i$, the best particle which was found until now for particle P_i is stored. In step 8, we create a $D-dimensional$ particle $Lbest_i$ for each particle P_i. Step 8 is only executed in the LPSO algorithm. In the $Lbest_i$ particle, the best particle which was found for P_i particle until now among its Ne neighbors is stored. At the start the values stored in $Lbest_i$ particle are equal to zero.

In step 9, we create a $D-th$ dimensional velocity vector V_i for each particle P_i. At the start, each $j-th$ value in vector V_i is equal to zero. In step 10, we evaluate the particle P_i using the previously defined objective function $OF(.)$. If the main goal of the algorithm is a minimization of the objective function then the best particle P_i is the particle with the lowest value of objective function $OF(.)$. If the main goal of the algorithm is a maximization of the objective function then the best particle P_i is the particle with the highest value of the objective function $OF(.)$. In step 12, we select and assign the best (from the whole swarm) particle P_i to the particle $Gbest$ (only in the GPSO algorithm) or we select and assign to particle $Lbest_i$ the best particle among Ne nearest neighbors of particle P_i (only in the LPSO algorithm). In step 13, the main loop of the algorithm is started. This loop is executed while the termination condition is not met. In the practical applications of the PSO algorithm we can mention three main termination criteria: maximal number of generations, maximal time of PSO algorithm operations, and PSO algorithm convergence. In step 16, we update the values from velocity vector V_i for each $j-th$ dimension. In the GPSO algorithm we use the equation from step 17, and in the LSPO algorithm we apply the equation from step 18. In both steps (17 and 18), the r_1 and r_2 are the randomly chosen values from the range [0, 1). The r_1 and r_2 values are different for each $V_{i,j}$ value. In step 19, we update all the $j-th$ decision variable values in each $i-th$ particle. In step 20, we check if newly created value $P_{i,j}$ is within the range $[P_{i,j}^{min}, P_{i,j}^{max}]$. If the value of the $j-th$ decision variable for $i-th$ particle is higher than the $P_{i,j}^{max}$ ($P_{i,j} > P_{i,j}^{max}$) then the value $P_{i,j}^{max}$ is assigned to the $j-th$ decision variable in $i-th$ particle ($P_{i,j} = P_{i,j}^{max}$). If the value of the $j-th$ decision variable for $i-th$ particle is lower than the $P_{i,j}^{min}$ ($P_{i,j} < P_{i,j}^{min}$) then the value $P_{i,j}^{min}$ is assigned to the $j-th$ decision variable in $i-th$ particle ($P_{i,j} = P_{i,j}^{min}$). In step 22, we evaluate the particle P_i using objective function $OF(.)$. In step 23, we check whether the value of objective function $OF(.)$ for particle P_i is better than the value of objective function $OF(.)$ for particle $Pbest_i$. If yes, then we assign the particle P_i to its corresponding particle $Pbest_i$ (step 24). In step 26, we select the best particle among the Ne nearest neighbor of particle P_i and then we assign this particle to temporary particle T. In step 27, we check whether the objective function $OF(.)$ for particle T is better than the objective function $OF(.)$ for particle $Lbest_i$. If yes, we assign the particle T to the particle $Lbest_i$. The steps 26-29 are only executed for the LPSO algorithm. In step 31, we select the best particle from the whole swarm P and next, we assign this particle to temporary particle T. In step 32, we check whether the objective function $OF(.)$ for particle T is better than the objective function $OF(.)$ for particle $Gbest$. If yes, we assign the particle T to the particle $Gbest$. The steps 31-34 are only executed for the GPSO algorithm. In step 36, we return the particle $Gbest$ as a result (only in the GPSO algorithm) or we select Re, the best particles among all $Lbest_i$ particles ($i \in [1, N]$) and return these particles as a result.

20.3 Modifications of the PSO algorithm

20.3.1 Velocity clamping

In the original PSO algorithm, the velocity value quickly explodes to the large values. One of the solutions for this problem is the introduction of the limited step sizes as shown in equation 20.1

$$V_{i,j} = \begin{cases} V_{i,j} & \text{if } |V_{i,j}| < V_j^{max} \\ V_j^{max} & \text{if } |V_{i,j}| \geq V_j^{max} \end{cases} \tag{20.1}$$

where: the V_j^{max} is a maximal value of velocity for $j-th$ decision variable. The V_j^{max} value can be equal by the all PSO generations or can be computed using formula 20.2 [4] or formula 20.5 [5].

$$V_j^{max} = \left(1 - \left(\frac{t}{n_t}\right)^h\right) \cdot V_j^{max} \tag{20.2}$$

where: t is a current number of PSO algorithm generation, n_t is an assumed maximal number of generations, h is a positive constant which can be chosen by trial-and-error.

$$V_j^{max} = \delta \cdot \left(P_j^{max} - P_j^{min}\right) \tag{20.3}$$

where: P_j^{max} and P_j^{min} are respectively the maximum and minimum values of the search domain for the $j-th$ decision variable, $\delta \in (0, 1]$ is a constant which can be chosen by trial-and-error.

20.3.2 Inertia weight

Inertia weight was developed to control the exploration and exploitation properties of the PSO algorithm. When we use the inertia weight factor the formula for velocity update is as follows for the GPSO algorithm:

$$V_{i,j} = \omega \cdot V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Gbest_j - P_{i,j}) \tag{20.4}$$

and for the LPSO algorithm:

$$V_{i,j} = \omega \cdot V_{i,j} + c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + c_2 \cdot r_2 \cdot (Lbest_{i,j} - P_{i,j}) \quad (20.5)$$

where: ω is a factor called an inertia weight which significantly affects the convergence and exploration-exploitation trade-off in PSO. The typical value for the inertia weight factor is from the range $[0.4, 0.9]$ [3]. When $\omega \geq 1$ then velocities increase over time (swarm diverges) and particles fail to change direction toward more promising regions. When the $0 < \omega < 1$ then particles decelerate and more promising regions can be found. Of course the convergence of the PSO algorithm is also dependent on the values of c_1 and c_2. In the exploration and exploitation trade-off, we can say that the large values of ω favor exploration, and the small values of ω promote exploitation. It is hard to determine which value of ω is the best value because it is highly dependent on the problem which is being solved. Sometimes, the decreasing inertia weight is used in PSO using the following formula.

$$\omega = \frac{(t^{max} - t)^n}{(t^{max})^n} \cdot (\omega_{max} - \omega_{min}) + \omega_{min} \quad (20.6)$$

where: $\omega \in [\omega_{min}, \omega_{max}]$, t^{max} is a maximal number of generations, t is a current number of generations.

20.3.3 Constriction coefficient

The constriction coefficient was developed to ensure convergence to a stable point without the need for velocity clamping. The new equation for velocity $V_{i,j}$ value (for the global version of the PSO algorithm) is as follows.

$$V_{i,j} = \gamma \cdot [V_{i,j} + \phi_1 \cdot (Pbest_{i,j} - P_{i,j}) + \phi_2 \cdot (Gbest_j - P_{i,j})] \quad (20.7)$$

where: γ is a constriction coefficient ($\gamma \in [0, 1]$) and is defined as follows.

$$\gamma = \frac{2 \cdot \kappa}{|2 - \phi - \sqrt{\phi \cdot (\phi - 4)}|} \quad (20.8)$$

and $\phi = \phi_1 + \phi_2$, $\phi_1 = c_1 \cdot r_1$, $\phi_2 = c_2 \cdot r_2$, the κ parameter controls the exploration ($\kappa \approx 0 \rightarrow$ fast convergence) and exploitation ($\kappa \approx 1 \rightarrow$ slow convergence) properties of the algorithm. If $\phi > 4$ and $\kappa \in [0, 1]$ then the swarm is guaranteed to converge to a stable point. Typically, the constriction factor is set to be $\gamma = 4.1$ and $c_1 = c_2 = 2.05$ [6].

20.3.4 Acceleration coefficients c_1 and c_2

The coefficient c_1 and c_2 are named as acceleration coefficients and they are responsible for the cognition part of the PSO algorithm (coefficient c_1) and the social part of the PSO algorithm (coefficient c_2). If $c_1 > 0$ and $c_2 = 0$ then we have a cognition only model of the PSO algorithm. Of course, if $c_1 = 0$ and $c_2 > 0$ then we have a social only model of the PSO algorithm. When $c_2 > c_1$ then the PSO algorithm is more suitable for the solving of unimodal problems. On the other hand when $c_1 < c_2$ then the algorithm PSO is more suitable for solving multimodal problems. In addition, we can introduce the adaptive changes of acceleration coefficient values using a formula.

$$c_1 = \left(c_1^{min} - c_1^{max}\right) \cdot \frac{t}{t^{max}} + c_1^{max} \tag{20.9}$$

$$c_2 = \left(c_2^{max} - c_1^{min}\right) \cdot \frac{t}{t^{max}} + c_2^{min} \tag{20.10}$$

where: $c_1 \in \left[c_1^{min}, c_1^{max}\right]$, $c_2 \in \left[c_2^{min}, c_2^{max}\right]$, t is a current number of generations ($t \in [1, t^{max}]$), t^{max} is a maximal assumed number of generations.

20.4 Application of PSO algorithm for IIR digital filter design

20.4.1 Problem description

In general the design of digital filters with non-standard amplitude characteristics is a very complex problem. The objective function describing this problem is a multimodal function, therefore the gradient optimization techniques can easily be stuck at local optima. Of course, digital filters with standard amplitude characteristics can be designed using existing standard approximations as for example with Butterworth, Cauer or Chebyshev. However, this problem becomes more complicated if the digital filter is supposed to possess non-standard amplitude characteristics (as for example in phase or amplitude equalizers). Then the standard approximations are not useful for us. The transfer function $H(z)$ in z domain for the IIR (Infinite Impulse Response) digital filter is given by formula 20.11.

$$H(z) = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + ... + b_{n-1} \cdot z^{-(n-1)} + b_n \cdot z^{-n}}{1 - \left(a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + ... + a_{n-1} \cdot z^{-(n-1)} + a_n \cdot z^{-n}\right)} \tag{20.11}$$

where: $b_0, ..., b_n$ and $a_1, ..., a_n$ are the transfer function coefficients, n is a filter order (positive integer value). The main objective for the PSO algorithm is to find such a set of transfer function coefficients ($b_0, ..., b_n$ and $a_1, ..., a_n$) which ensure that the digital filter will be stable (all poles of the transfer function $H(z)$ must be located inside the unitary circle in the z plane), and its amplitude characteristics will fulfill all design assumptions.

20.4.2 How can the PSO algorithm be used for this problem?

We want to create an optimization method which leads to a gradual improvement of the designed filter parameters. We would like to obtain the digital filter with the lowest deviation of amplitude characteristics from the idealized one. The proposed method is based on the local version of the PSO algorithm and it consists of eight steps which are as follows. In the first step, the initial swarm P which consists of N particles P_i (each particle is a D-dimensional vector; the dimension depends on the filter order n; if filter order is n then the number of dimensions D is equal to $2 \cdot n + 1$) is randomly created. The designed filter coefficients are coded in the particle P_i as follows.

$$P_i = \{b_0\ b_1\ b_2\ ...\ b_{n-1}\ b_n\ a_1\ a_2\ ...\ a_{n-1}\ a_n\}$$

Next, we create the $Pbest_i$, $Lbest_i$ and V_i vectors for each $i-th$ particle P_i and one vector $Gbest$ for whole swarm. Vector $Pbest_i$ determines the best position of the particle P_i in the search space among all positions obtained so far. Vector $Lbest_i$ determines the best position of another particle among NN nearest neighbors of particle P_i found until now. Vector V_i represents the value of particle P_i velocity. Vector $Gbest$ determines the global best solution found by the PSO algorithm. At the start vector P_i is assigned to the $Pbest_i$ vector for $i-th$ particle, vectors $Lbest_i$, V_i and $Gbest$ are equal to zero. Additionally, in the case when, during the run of algorithm, a return to the first step from the seventh step occurs then the vector $Gbest$ is assigned to the randomly chosen particle P_i from the swarm.

In the second step, each particle P_i is evaluated using objective function $OF(.)$. In order to compute the value of $OF(.)$ for particle P_i, the FFT (Fast Fourier Transform) is performed separately for the filter coefficients from nominator (coefficients b_k) and denominator (coefficients a_k). When we have FFT results, we can compute the amplitude characteristics $H(f)$ [dB] for the digital filter represented by filter coefficients which are stored in particle P_i as follows.

$$H(f) = 20 \cdot log_{10} \left(\sqrt{H_{real}(f)^2 + H_{imaginary}(f)^2} \right) \tag{20.12}$$

If we have the amplitude characteristics we can compute the objective function $OF(.)$ for particle P_i using the following formula.

$$OF(.) = \sum_{g=1}^{G} Error(f_g) + \sum_{m=1}^{M} Stability_m \tag{20.13}$$

where:

$$Error(f_g) = \begin{cases} |H(f_g) - C_1(f_g)| & \text{if } H(f_g) > C_1(f_g) \\ |H(f_g) - C_2(f_g)| & \text{if } H(f_g) < C_2(f_g) \\ 0 & \text{if } H(f_g) \in [C_2(f_g), C_1(f_g)] \end{cases} \tag{20.14}$$

$$Error(f_g) = \begin{cases} (|z_s| - 1) \cdot p + p & \text{if } |z_s| \geq 1 \\ 0 & \text{if } |z_s| < 1 \end{cases} \tag{20.15}$$

where: M is the number of poles of the transfer function 20.11, G is the number of output samples from FFT transform divided by 2 (we have assumed $G = 256$), p is a value of the penalty (we have assumed $p = 10^5$), f_g is a $g-th$ value of normalized frequency ($f_g \in [0, 1]$; Nyquist frequency = 1), $|z_s|$ is the module of the $s-th$ pole of the transfer function in the z plane, $C_1(f_g)$ is an upper constraint of the amplitude characteristics for frequency f_g, $C_2(f_g)$ is a lower constraint of the amplitude characteristics for frequency f_g, $H(f_g)$ is a value of amplitude characteristics for frequency f_g.

The value of objective function $OF(.)$ is higher when the sum of the absolute values of deviations between constraints and amplitude characteristics $H(f)$ for particle P_i is higher (first part of objective function 20.13). Additionally, the value of $OF(.)$ increases when the digital filter represented by particle P_i is not stable (second part of objective function 20.13). The PSO algorithm minimizes objective function $OF(.)$.

Next, when the value of the computed $OF(.)$ function for particle P_i is lower than the value written down in vector $Pbest_i$ then the values from particle P_i are assigned to the vector $Pbest_i$.

In the third step, for each particle P_i the best particle $Lbest_i$ is selected among NN nearest neighbors of the particle P_i. The similarity between particles was determined by the values of the $OF(.)$. The two particles are more similar to each other when their values of $OF(.)$ are close to each other.

In the fourth step, the vector V_i is computed for each particle P_i using the following formula:

$$V_{i,j} = \gamma \cdot V_{i,j} + \gamma \cdot c_1 \cdot r_1 \cdot (Pbest_{i,j} - P_{i,j}) + \gamma \cdot c_2 \cdot r_2 \cdot (Lbest_{i,j} - P_{i,j}) \tag{20.16}$$

where: j is the number of the decision variable ($j \in [1, D]$), γ is a scaling coefficient (we have assumed $\gamma = 0.729$), c_1 and c_2 are the learning coefficients (usually $c_1 = c_2$, we have assumed $c_1 = c_2 = 0.3$), r_1 and r_2 are random real numbers with uniform distribution selected separately for each $j - th$ dimension ($r_1 \in [0, 1]$ and $r_2 \in [0, 1]$).

In the fifth step, the new particle P_i is created using the following formula:

$$P_{i,j} = P_{i,j} + r_3 \cdot V_{i,j} \tag{20.17}$$

where: r_3 is a random real number selected separately for each $j-th$ dimension ($r_3 \in [0, 1]$).

In the sixth step, one randomly created particle is inserted into the place of the worst particle in the whole swarm. The worst particle means the particle P_i with the highest value of the $OF(.)$ function. Due to this the convergence of the presented algorithm is improved.

In the seventh step, we check whether the particle P_i for which the value of $OF(P_i) = 0$ is found. If such a particle is not found then the ninth step of the algorithm is executed. If such a particle P_i is found then this particle is assigned to *Gbest* particle and the $OF(.)$ function is modified by decreasing the allowed values of deviations of amplitude characteristics for the designed digital filter.

In the eighth step, the algorithm jumps to the first step.

In the ninth step, the algorithm termination criterion is checked. The algorithm is stopped when the absolute value of the difference between $OF(.)$ function for the particles *Gbest* in generation t_1 and in generation $t_1 + t$ is lower than ϵ. As a final result of the algorithm operation, we obtain the values stored in the current *Gbest* particle. If the termination criterion is not fulfilled then the algorithm jumps to the second step.

20.4.3 Description of experiments

The method presented was tested by the design of the $10 - th$ order ($n = 10, D = 2 \cdot n + 1 = 21$) IIR digital filter with linearly falling amplitude characteristics. The parameters of amplitude characteristics are as follows. The value of attenuation is equal to 0 [dB] for normalized frequency $f = 0$ and the value of attenuation is equal to 40 [dB] for normalized frequency $f = 1$. At the start of the algorithm, we have assumed that the maximal deviation value between amplitude characteristics represented by particle P_i and the digital filter design assumptions for any normalized frequency value cannot be higher than 10 [dB]. During algorithm operation, the maximal deviation value is decreased with 1 [dB] when a digital filter which fulfills all design assumptions is found. After achieving the maximal deviation value equal to 1 [dB], the value of this parameter was decreasing by 0.1 [dB] when the digital filter which fulfilled all design assumptions was found. Designing a stable digital filter whose amplitude characteristics fulfill all design assumptions with the lowest maximal deviation value from the ideal amplitude characteristics is the main goal of the presented algorithm. The values of the algorithm parameters are as follows. The number of particles $N = 100$, dimension of solution space $D = 21$, nearest neighbor parameter $NN = 3$, $t = 500$ generations and $\epsilon = 0.0001$. The initial values for particle P_i in each $j - th$ dimension were randomly chosen from the range $[-1, 1]$.

20.4.4 Results obtained

The IIR digital filter which fulfills all design assumptions was found after 19332 generations. This filter is stable (all poles of the transfer function are located in the unitary circle in the z plane). The maximal value of the amplitude characteristics' deviation from the ideal one does not exceed 0.3 [dB]. In Table 20.1 the number of generations which are required to design the IIR digital filter with assumed design assumptions (having deviations between ideal amplitude characteristics and amplitude characteristics generated by designed filter lower than maximal acceptable deviations) are shown.

TABLE 20.1
The number of generations required to obtain a digital filter with deviations of amplitude characteristics is lower than the prescribed maximal deviations from the ideal amplitude characteristics.

Maximal deviations [dB]								
10	9	8	7	6	5	4	3	2
15	21	32	50	98	172	230	304	446

Maximal deviations [dB]								
1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
601	810	1050	1170	1321	2012	2415	8585	–

We can see that only 15 generations are required to obtain a digital filter with deviations of attenuation lower than 10 [dB], 570 generations are required for deviations not higher than 1 [dB], and 8585 generations are required for deviations lower than 0.3 [dB]. The $10 - th$ order IIR digital filter cannot be found for deviations of attenuation lower than 0.2 [dB].

20.5 Conclusions

In this paper the PSO algorithm, in a local and global version, was presented in detail. Some modifications of the PSO algorithm were demonstrated and discussed. Finally, the exemplary application of the PSO algorithm to a real-world problem was shown. The PSO algorithm was used for IIR digital filter design with non-standard amplitude characteristics. As shown in Section 20.4, it is possible to design digital filters with non-standard amplitude characteristics using the PSO algorithm. The designed digital filter fulfills all design assumptions and is a stable filter (all poles of transfer function are located in a unitary circle in the z plane). Using the PSO algorithm we can obtain higher automation of the digital filter design process, because expert knowledge concerning the filter design is not required. Also, it is worth noting that swarm based digital filter design is fundamentally different from the traditional filter design process. The swarm digital filter design process possesses fewer constraints than the design process which is based on human (designer) knowledge and experience [9]. The human designer is not only limited by the technology, but also by their own habits, intelligence and so on. The application of the PSO algorithm to digital filter design allows these limitations to be avoided and provides access to new possibilities.

References

1. J. Kennedy, R.C. Eberhart. “Particle swarm optimization” in *Proc. of IEEE International Conference on Neural Networks*, 1995, pp. 1942-1948.
2. A. Colorni, M. Dorigo, V. Maniezzo. “Distributed optimization by ant colonies” in *Proc. of the European Conf. on Artificial Life*, 1991, pp. 134-142.
3. J. Xin, G. Chen, Y. Hai. “A particle swarm optimizer with multi-stage linearly-decreasing inertia weight” in *Computational Sciences and Optimization*, vol. 1, 2009, pp. 505-508.
4. H. Fan. “A modification to particle swarm optimization algorithm”. *Engineering Computations*, vol. 19(8), pp. 970-989, 2002.
5. I.K. Gupta. “A review on particle swarm optimization”. *International Journal of Advances Research in Computer Science and Software Engineering*, vol. 5(4), pp. 618-623, 2015.
6. Y. Yang, J. Wen, X. Chen. “Improvements on particle swarm optimization algorithm for velocity calibration in microseismic monitoring”. *Earthquake Science*, vol. 28(4), pp. 263-273, 2015.
7. A. Slowik, H. Kwasnicka. “Nature inspired methods and their industry applications – swarm intelligence algorithms”. *IEEE Transactions on Industrial Informatics*, vol. 14(3), pp. 1004-1015, 2018.
8. M.H. Yar, V. Rahmati, H.R.D. Oskouei. “A survey on evolutionary computation: methods and their applications in engineering”. *Modern Applied Science*, vol. 10(11), pp. 131-139, 2016.
9. A. Slowik, M. Bialko. “Design and optimization of IIR digital filters with non-standard characteristics using particle swarm optimization” in *Proc. of 14th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2007*, pp. 162-165, 2007.