

Improvements on particle swarm optimization algorithm for velocity calibration in microseismic monitoring

Yue Yang · Jian Wen · Xiaofei Chen

Received: 17 February 2015 / Accepted: 8 June 2015 / Published online: 24 July 2015
© The Author(s) 2015. This article is published with open access at Springerlink.com

Abstract In this paper, we apply particle swarm optimization (PSO), an artificial intelligence technique, to velocity calibration in microseismic monitoring. We ran simulations with four 1-D layered velocity models and three different initial model ranges. The results using the basic PSO algorithm were reliable and accurate for simple models, but unsuccessful for complex models. We propose the staged shrinkage strategy (SSS) for the PSO algorithm. The SSS-PSO algorithm produced robust inversion results and had a fast convergence rate. We investigated the effects of PSO's velocity clamping factor in terms of the algorithm reliability and computational efficiency. The velocity clamping factor had little impact on the reliability and efficiency of basic PSO, whereas it had a large effect on the efficiency of SSS-PSO. Reassuringly, SSS-PSO exhibits marginal reliability fluctuations, which suggests that it can be confidently implemented.

Keywords Particle swarm optimization · Staged shrinkage strategy (SSS) · Global optimization (GO) · Geophysical inversion · Microseismic velocity calibration

1 Introduction

Inversion problems are basic issues in geophysics. Solutions to these problems can be classified into deterministic or probabilistic methods, or into local or global optimization methods. Most local optimization algorithms are deterministic, whereas global optimization algorithms are stochastic (Sen and Stoffa 2013).

With careful initialization, local optimizations techniques, such as the Newton, quasi-Newton (e.g., the BFGS), and conjugate-gradient (CG) methods, can obtain stable results based on gradient information after several iterations (Gill et al. 1981; Tarantola 2005). However, because the initialization may be unreliable, we cannot be certain that the iterations converge to the global minima. Compared with local optimization methods, global optimization (GO) methods, such as Simulated Annealing (SA) and genetic algorithms (GAs), typically need a huge amount of forward calculations to obtain a global optima. The crucial issues for GO methods are the convergence speed and computational efficiency (Kiranyaz et al. 2013).

Particle swarm optimization (PSO) (Russ et al. (1996)) is an efficient global optimization method. PSO has similar characteristics to evolution algorithms (EAs). These methods are stochastic (e.g., the SA) and are population-based evolutionary algorithms (e.g., the GAs) (Banks et al. 2007; Kiranyaz et al. 2013). PSO has been shown to outperform the GA methods (Angeline 1998). It has been extensively applied because of its fast convergence rate and simple implementation (Poli 2008). However, PSO is significantly slower to converge in certain circumstances, because it can converge prematurely and get trapped in local optima. Therefore, it is important to improve PSO so that it can be applied to the seismic inversion.

Y. Yang (✉) · J. Wen · X. Chen
School of Earth and Space Sciences, University of Science and Technology of China, Hefei 230026, China
e-mail: yyzf@mail.ustc.edu.cn

Y. Yang · J. Wen · X. Chen
Laboratory of Seismology and Physics of Earth's Interior,
University of Science and Technology of China,
Hefei 230026, Anhui, China

Y. Yang · J. Wen · X. Chen
Mengcheng National Geophysical Observatory,
Hefei 230026, Anhui, China

PSO improvements can be categorized into two sets: improvements to parameters and modifications to evolution rules. Inertia weight (Shi and Eberhart 1998) and constriction factors (Clerc 1999) have been introduced to the original PSO formulation to better control a particles' velocity. Carlisle and Dozier (2001) attempted to determine explicit and implicit off-the-shelf parameters for PSO. However, rather than offering a panacea for the parameter selection process, their work raised awareness regarding the importance of appropriate parameter values (Xu and Rahmat-Samii 2007; Zhang et al. 2005a, b). Other PSO improvements modified the particle learning rules, or were hybrids with other GO methods (Liang et al. 2006; Mendes et al. 2004; Ratnaweera et al. 2004).

In this paper, we first briefly introduce the principles of the PSO algorithm using the examples of synthetic inversion for velocity calibration in microseismic monitoring. We explore the inevitable limitations tackling with complex models. Then, we describe the staged shrinkage strategy (SSS) for improving the PSO algorithm, which avoids premature and speeds up the convergence rate. Finally, we present simulation results to demonstrate the superiority of our proposed SSS-PSO algorithm. We also investigated the influence of the built-in velocity clamping factor on the reliability and efficiency for the PSO and SSS-PSO algorithms.

2 PSO algorithm and its limitations

2.1 Fundamentals of the PSO algorithm

PSO was proposed by Eberhart and Kennedy (1995). It maintains a population of particles that represent potential solutions in the search space. These particles have two physical characteristics: location and velocity. The locations represent potential solutions. The velocities represent a kinetic property of the particles. The optimization rules follow mathematical formulas. A flow diagram for PSO is shown in Fig. 1. In a d -dimension model space, the particle swarm size is N , with x_{id} and v_{id} denoting the position and velocity of particle i ($i = 1, 2, 3, \dots, N$). p_{id} and p_{gd} are the personal best and global best positions for particle i . The rules to update a particle's velocity and position are

$$\begin{cases} v_{id}(k+1) = v_{id}(k) + c_1 r_1 [p_{id}(k) - x_{id}(k)] + c_2 r_2 [p_{gd}(k) - x_{id}(k)] \\ x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \end{cases}, \tag{1}$$

where r_1 and r_2 are random numbers in the range $[0, 1]$, and c_1 and c_2 are acceleration constants (typically $c_1 = c_2 = 2$). The first part of the velocity update equation is called the "velocity inertia" term, the second is the "cognition" term (indicating that the particles are influenced by their own

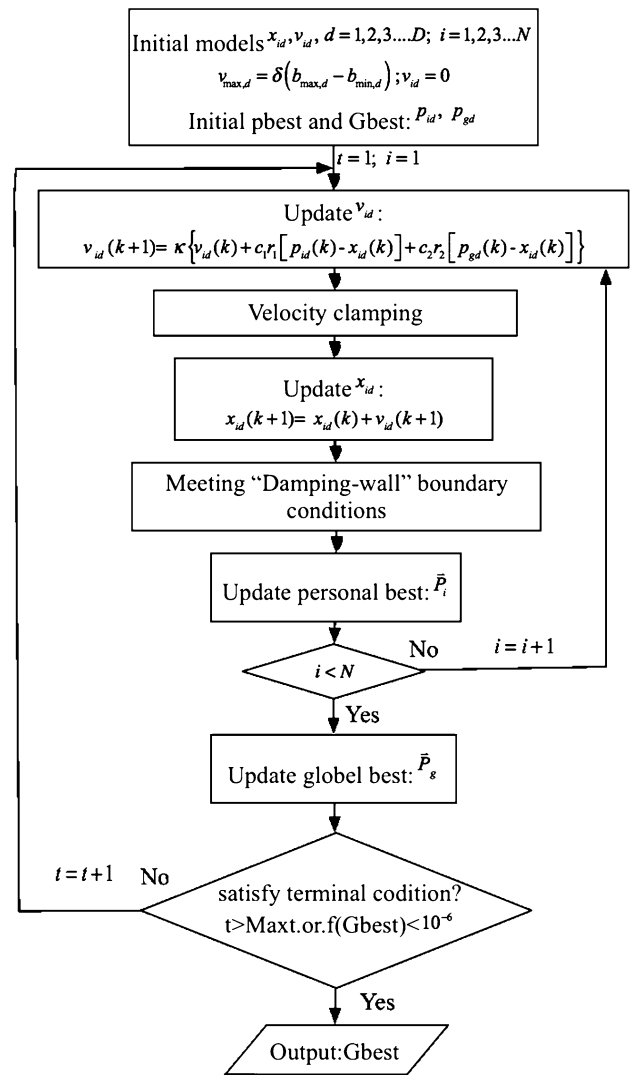


Fig. 1 Flow chart of the basic PSO implementation

past), and the third is the "social" term (indicating that the particles learn from swarm intelligence) (Eberhart and Kennedy 1995; Shi and Eberhart 1998).

The algorithm for updating the personal best is

$$\bar{P}_i(k+1) = \begin{cases} \bar{x}_i(k+1), & \text{when } F[\bar{x}_i(k+1)] < F[\bar{P}_i(k)] \\ \bar{P}_i(k), & \text{when } F[\bar{P}_i(k)] < F[\bar{x}_i(k+1)] \end{cases}, \tag{2}$$

where \bar{P}_i is the personal best memory of \bar{x}_i . F is the objective function defined in Eq. 7.

To restrict the maximum step at each iteration and prevent overflow, the maximum velocity ($v_{max,d}$) should be a proportion of the range of the particle search space. This proportion is called the velocity clamping factor (δ). According to Russ et al. (1996) and Clerc (1999), the mathematical relationship between these parameter is

$$v_{\max,d} = \delta(b_{\max,d} - b_{\min,d}), \quad \text{for } \delta \in (0, 1], \quad (3)$$

where $v_{\max,d}$ is the allowed maximum velocity of particles, δ is the velocity clamping factor, and $b_{\max,d}$ and $b_{\min,d}$ are the maximum and minimum location values of particles at d -th dimension.

So if the velocity update results in a step that is too large, the maximum velocity limits the velocities as follows:

$$v_{id} = \begin{cases} v_{\max,d} & \text{when } v_{id} > v_{\max,d} \\ v_{id} & \text{when } |v_{id}| \leq v_{\max,d} \\ -v_{\max,d} & \text{when } v_{id} < -v_{\max,d} \end{cases} \quad (4)$$

Obviously, unlike acceleration constants that balance the local and global search, the velocity clamping factor controls the convergence speed. A larger velocity clamping factor allows big steps and contributes to a fast convergence rate, but increases the probability that the method will get trapped in local optima. A smaller velocity clamping factor constrains the particle step size, slows down the optimization process, and increases the particles diversity (Poli et al. 2007). Essentially, the acceleration constants and velocity clamping factor have corporative roles in the convergence of the PSO algorithm.

Inertia weights (Shi and Eberhart 1998) have been introduced to original PSO velocity update formula.

$$v_{id}(k + 1) = wv_{id}(k) + c_1r_1[p_{id}(k) - x_{id}(k)] + c_2r_2[p_{gd}(k) - x_{id}(k)]. \quad (5)$$

Here, w is the inertia weight. The inertia weight can take various values (Rezaee Jordehi and Jasni 2013). It works with velocity clamping factor to have a better control on velocity.

Then, constriction factors (Clerc 1999) were introduced to the original PSO formula as follows:

$$v_{id}(k + 1) = \kappa\{v_{id}(k) + c_1r_1[p_{id}(k) - x_{id}(k)] + c_2r_2[p_{gd}(k) - x_{id}(k)]\} \\ \kappa = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}, \quad \varphi = c_1 + c_2, \quad \varphi > 4. \quad (6)$$

where K is the constriction factor. Typically, the constriction factor is set to be $K = 4.1$, and $c_1 = c_2 = 2.05$. Although the constriction factors were designed to control particles velocity without velocity clamping process, but simulation tests have shown that constriction-based PSO performance is better using velocity clamping (Eberhart and Shi (2000)).

The boundary conditions can also affect the performance of the algorithm (Xu and Rahmat-Samii 2007; Zhang et al. 2005b). In this study, we used the “damping-wall” boundary condition as follows:

$$(x_{id}, v_{id}) = \begin{cases} (b_{\max,d}, -rv_{id}), & \text{if } x_{id} > b_{\max,d} \\ (b_{\min,d}, -rv_{id}), & \text{if } x_{id} < b_{\min,d} \end{cases}, \quad (7)$$

where $b_{\max,d}$ and $b_{\min,d}$ are the boundaries of the particles’ values in the d -th dimension; and r is a random number in $[0, 1]$.

2.2 Numerical examples of the PSO algorithm applied to velocity calibration

Velocity calibration is an indispensable data processing step in microseismic monitoring, which obtains reliable initial models for microseismic data analysis (Cipolla et al. 2012; Maxwell et al. 2010). Various methods have been applied to this problem (Pei et al. 2009; Warpinski et al. 2005; Warpinski and Du 2013). Velocity calibration is a nonlinear optimization with limited data because there is a lack of string shots and receivers. There is no accepted approach for reliably and effectively solving this problem without a priori information (log data or other resources). In this section, we present some numerical examples that applied the PSO algorithm to velocity calibration in microseismic monitoring with synthetic data.

To apply the PSO method to this problem, we chose four 1-D layered velocity models as presented in Pei et al. (2009). The models are shown in Fig. 2. We generated synthetic travel time data using the accurate two-point ray tracing method (Tian and Chen 2005). The objective function is

$$F = \sqrt{\frac{1}{n} \sum_{i=1}^n (t_{vp,i}^{\text{obs}} - t_{vp,i}^{\text{cal}})^2}, \quad (8)$$

where F is the objective function, n is the number of sensor shot direct wave picks, and $t_{vp,i}^{\text{obs}}$ and $t_{vp,i}^{\text{cal}}$ are forward and inversion calculations of the direct wave travel times.

When applying the basic PSO algorithm to recover the 1-D layered velocity models with synthetic travel time data, we used a constriction factor-based formula, the velocity clamping factor $\delta = 0.1$, and the population of particles $N = 30$ (Carlisle and Dozier 2001). The maximum number of iterations for outer loop was set to $t_{\max} = 1000$, and we set the maximum for the objective function to $F \leq 10^{-6}$ s. The PSO algorithm terminated when either of these conditions was exceeded. Four test models are shown in Fig. 2. For each test model, we ran 300 inversions with random initial models. These 300 random initial models were generated from three different initial model ranges. There were 100 random initial models for each given range. We set three different initial model ranges to test algorithm behavior, which are shown in Fig. 3 in the cyan line with two dots at both ends at each layer.

The inversion results are shown in Fig. 3. The sub-figures in row 1 to 4 (from top to bottom) are the inversion results for models 1 to 4; the sub-figures in each row (from left to right) correspond to the inversion results of 300

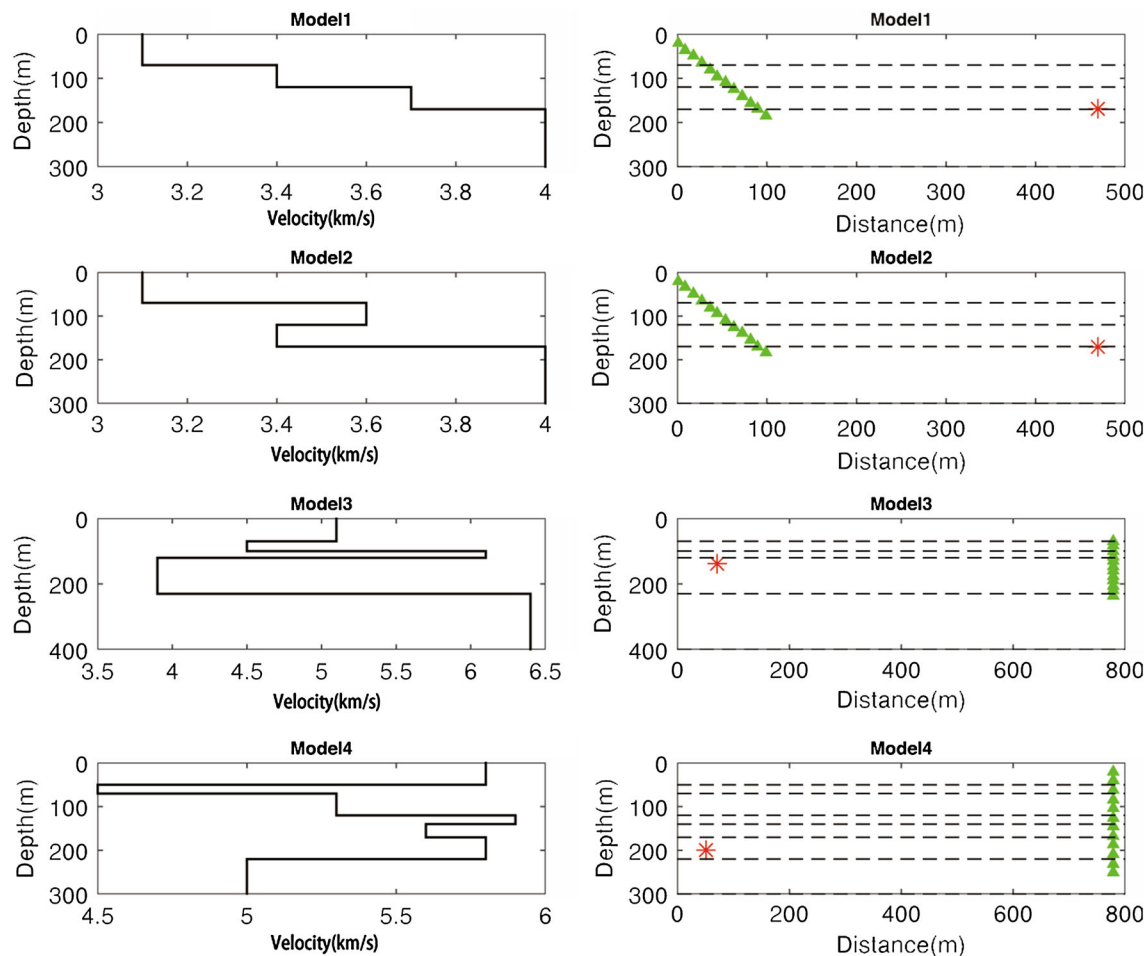


Fig. 2 Forward models and schematic geometry showing receiver arrays and perforation shot locations

simulations with three initial model ranges. For simplicity, we denote the sub-figure in row i and column j as sub-figure (i, j) . Obviously, each sub-figure shows the results of 100 inversions with random initial models within a certain given model range. We can see that the inversion results for the relatively simple models (models 1 and 2) are very good, even when the initial models are quite different from the true models in sub-figs. (1, 3) and (2, 3). However, the inversion results for models 3 and 4 (relatively complicated models) are not good. Sub-figs. (3, 1) to (3, 3) show that the inversion results for model 3 are quite good in general, except for the top layer. Note that no geophones are deployed in the top layer, so there is no ray path constraint. The inversion results for model 4 are shown in sub-figs. (4, 1) to (4, 3). Although the inversion results were more accurate when the initial model ranges were narrow, they did not generally converge to the true model. In particular, the second layer (low velocity layer) was the least accurate for any of the initial model ranges. Note that we cannot guarantee that the PSO inversion convergence for this model within 1000 iterations achieves the terminal objective function $F = 10^{-6}$.

2.3 Limitations of basic PSO

In four-layer models, basic PSO iterated less than 100 times to reach the terminal objective function value. In each layer, the velocity deviations were <0.0001 km/s. The successful inversion results for models 1 and 2 indicate that basic PSO algorithm is insensitive to the initial models and stable in repeated simulations. However, the results for model 4 show that the objective function values may stop decreasing sometimes in repeated inversions. This problem could not be alleviated by increasing the maximum number of iterations from 1000 to 5000 (Fig. 4).

3 Algorithm modifications

3.1 Modifications of basic PSO

In the geophysical inversion problem, the inversion parameters have different sensitivities to the objective function. The difference is significant in some situations, as

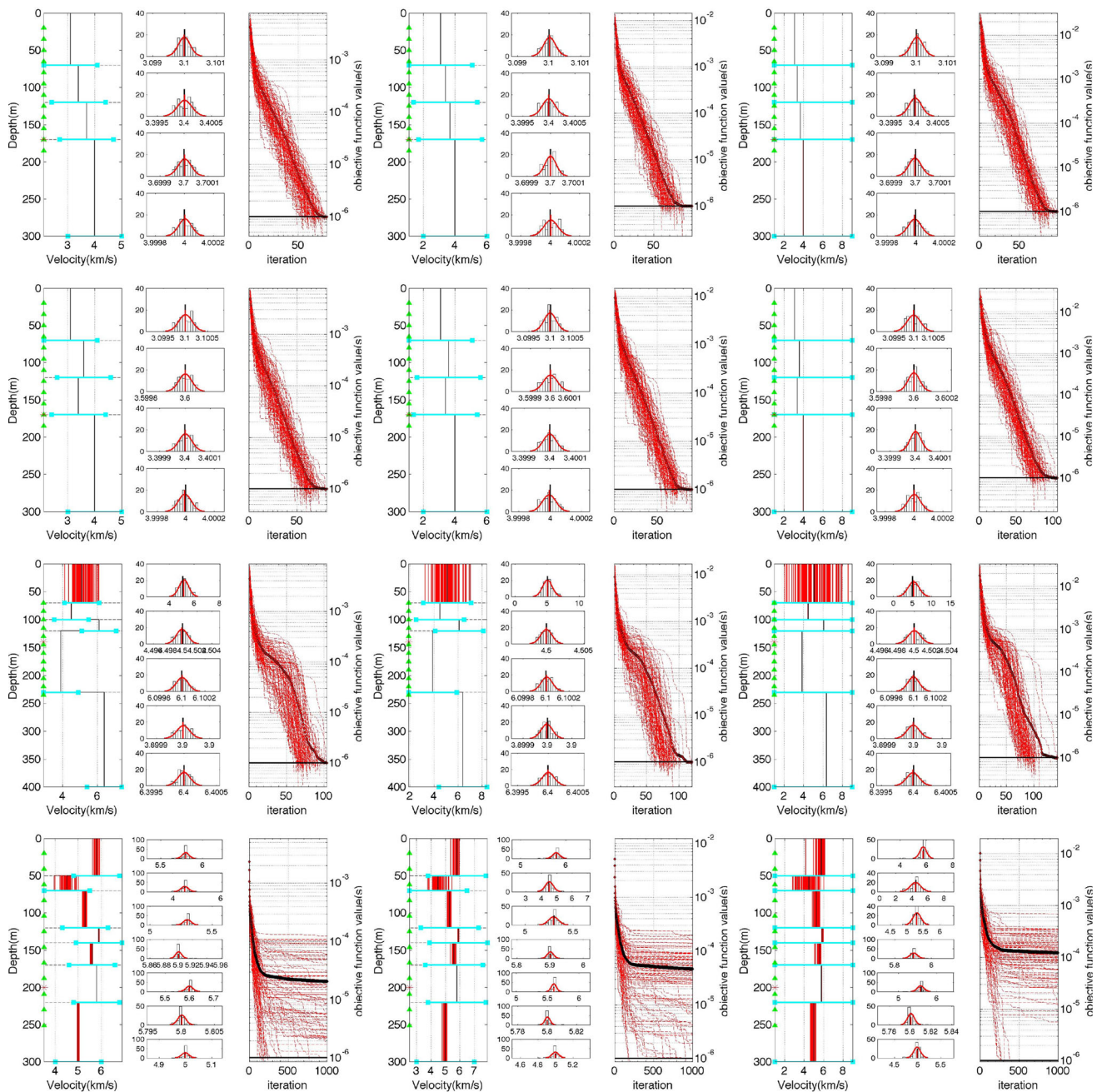


Fig. 3 PSO inversion results with 1000 maximum iterations. In each sub-plot, the plot on the left displays 100 repeated inversion results in *red*, which are occasionally overlapped by the *black lines* that represent the forward model. In each layer, the *cyan line* with two dots at both ends shows the initial model range. The plot in the *middle* gives histograms of each layers inversion results. The plot on the *right* depicts the objective function values, with the iterations represented by *red lines*. The *black line* is the average of the best values from 100 repeated inversions

revealed in model 4. The inversion results for the second layer (from the surface) in model 4 were the most scattered around the true value. The inversion parameter cannot be accurately calculated. The objective function because trapped in local optima, and stopped decreasing before satisfying the termination criteria. So modifying the search

space according to the dimension and the evolutionary stage of the particles is feasible and reasonable. With this in mind, we propose SSS, which encourages the process to escape from local optima.

This modified PSO algorithm is abbreviated to SSS-PSO, and its flow chart is shown in Fig. 5. Firstly, we

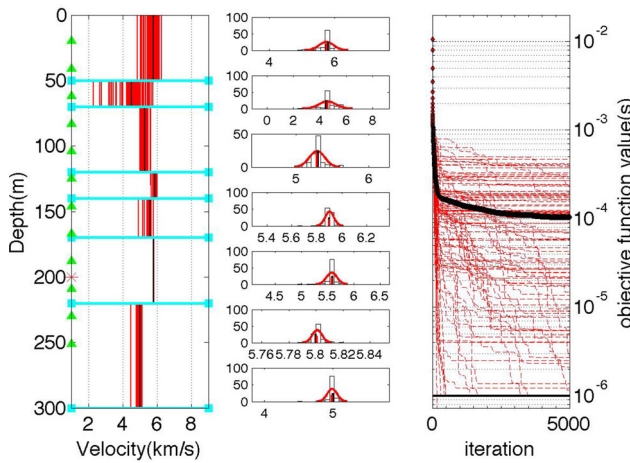


Fig. 4 PSO inversion results with 5000 maximum iterations. The plot on the left displays 100 repeated inversion results in red, which are occasionally overlapped by the black lines that represent the forward model. In each layer, the cyan line with two dots shows the initial model range. The plot in the middle gives histograms of each layers inversion results. The plot on the right depicts the objective function values, with the iterations represented by red lines. The black line is the average of the best values from 100 repeated inversions

initialize the global best memory ($g_{h,d}^{memo}$) and update it with each iteration. h is the global best change times. Then, we calculate the maximum, minimum, and average values of the particles and their personal best positions ($x_{max,d}$, $x_{min,d}$, $x_{ave,d}$ and $y_{max,d}$, $y_{min,d}$, $y_{ave,d}$). The maximal and minimal global best positions are acquired from the global best memory ($g_{h,d}^{memo}$)

$$g_{max,d}^{memo} = \begin{cases} \max(g_{1:h,d}^{memo}); & \text{when } h < h_{block} \\ \max(g_{h_{block}:h,d}^{memo}); & \text{when } h \geq h_{block} \end{cases}, \quad (9)$$

$$g_{min,d}^{memo} = \begin{cases} \min(g_{1:h,d}^{memo}); & \text{when } h < h_{block} \\ \min(g_{h_{block}:h,d}^{memo}); & \text{when } h \geq h_{block} \end{cases}$$

where h_{block} is 30 here which used to calculate $g_{h,d}^{memo}$. In the early evolution stage, the particles' global best positions may not be local minimums, and should be excluded. Otherwise, the global best positions will undermine the efficiency of the boundary shrinkage. If we use a fixed length that results in $g_{max,d}^{memo}$ and $g_{min,d}^{memo}$ through blocking, we may fail to achieve the global best position. Admittedly,

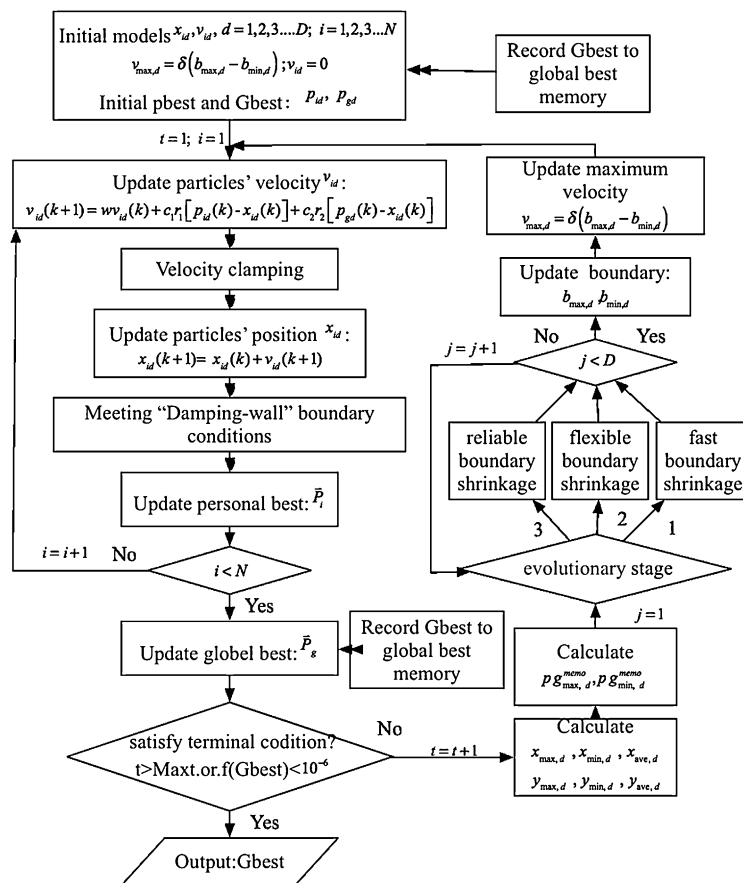


Fig. 5 Flow chart of the SSS modified PSO implementation

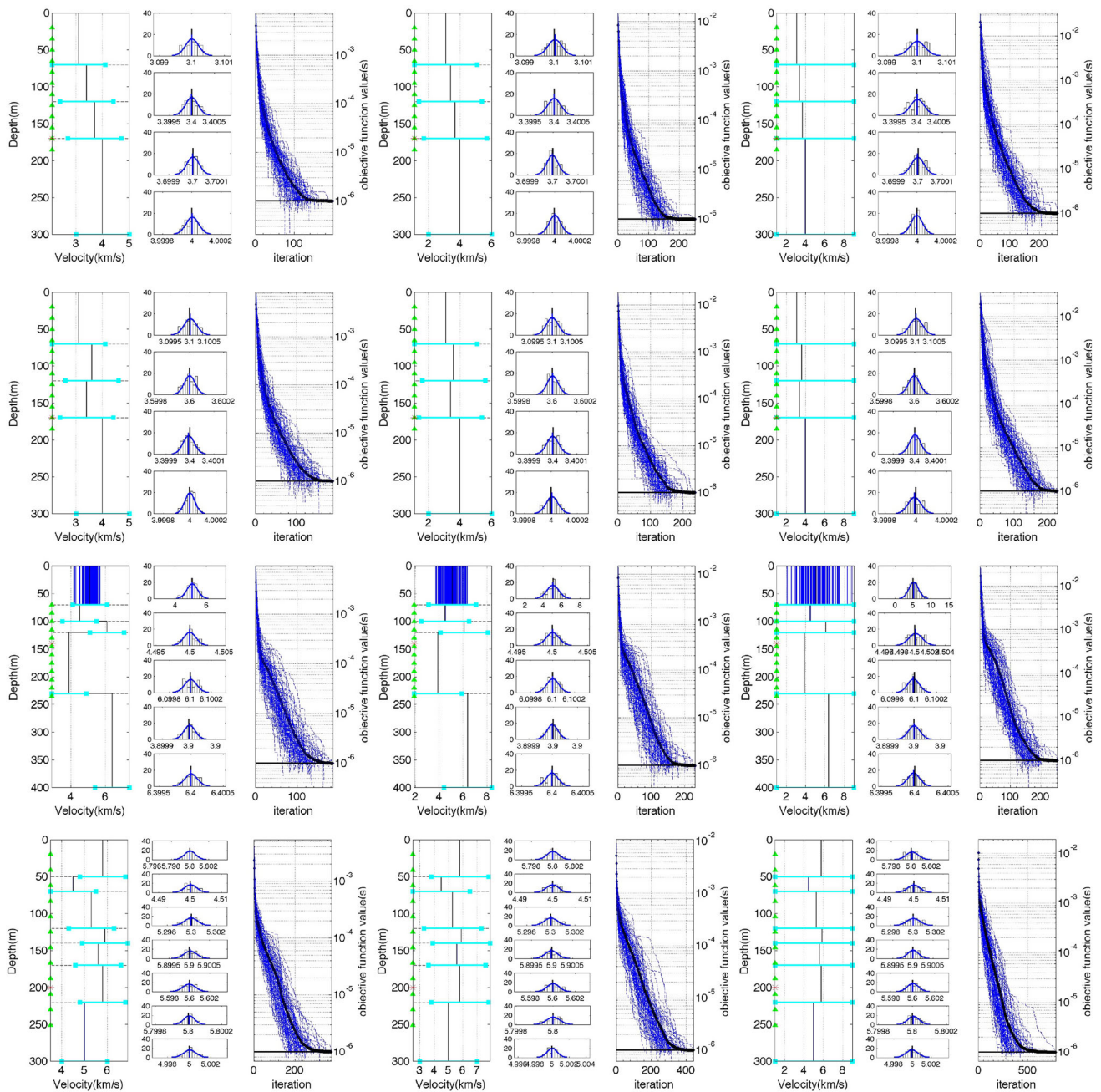


Fig. 6 SSS-PSO inversion results with 5000 maximum iterations. In each sub-plot, the plot on the left displays 100 repeated inversion results in blue, which are occasionally overlapped by the *black lines* that represent the forward model. In each layer, the *cyan line* with two dots at both ends shows the initial model range. The plot in the *middle* gives histograms of each layers inversion results. The plot on the *right* depicts the objective function values, with the iterations represented by *blue lines*. The *black line* is the average of the best values from 100 repeated inversions

more detailed work is required. However, our work suggests that $h_{\text{block}} = 30$ and the other following parameters for SSS implementations work well in velocity calibration tests.

Secondly, each inversion parameter dimension is sorted into a specified stage according to the positions of the particles. If the boundary space shrinks to 20 % of the

initial space range, we defined the evolutionary situation as stage 2. If the boundary space shrinks to 4 % of the initial range, we define the situation as stage 3. Otherwise, the boundary space is 20 %–100 % of the initial range, and is in stage 1. Then, we update the boundary range according to the following rules:

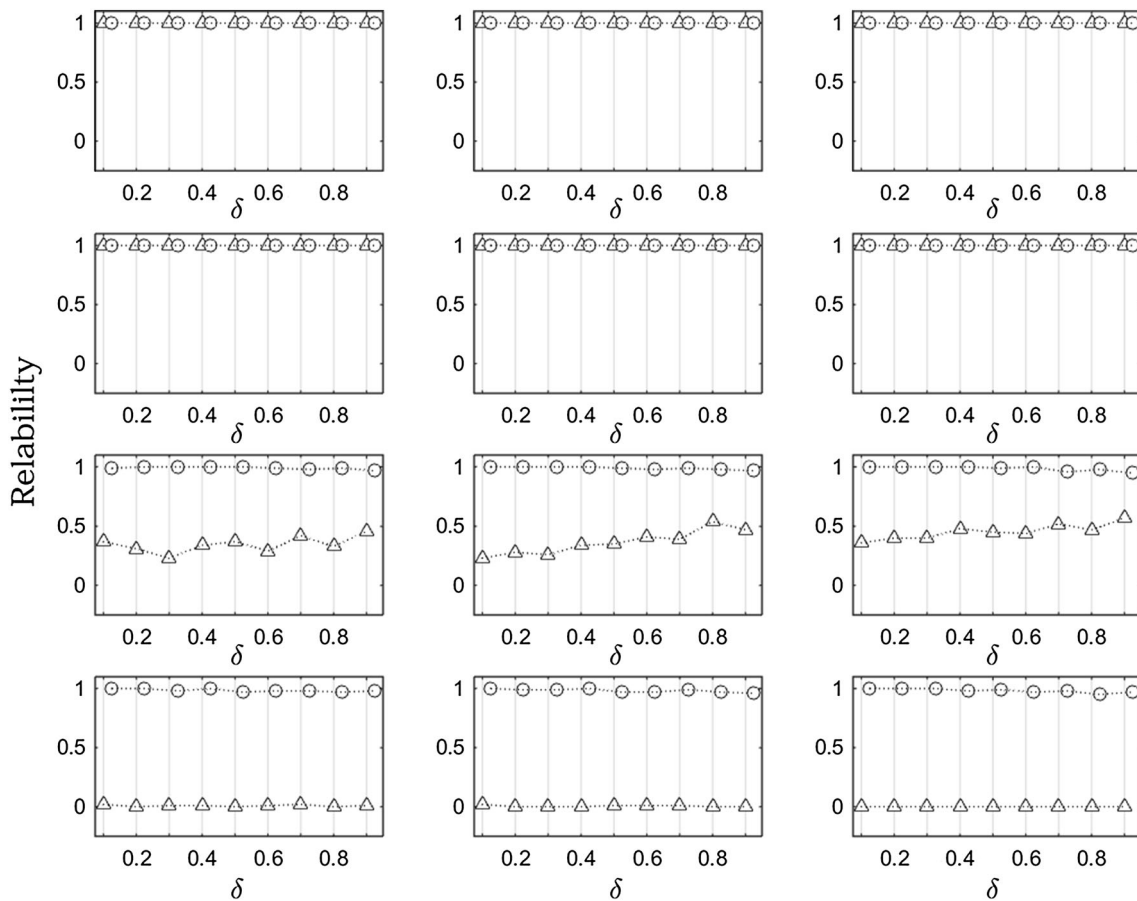


Fig. 7 Reliabilities of different models with varying velocity clamping factors. The triangle and circles represent the basic and SSS-PSO results, respectively. The plots from left to right are the results for three different initial ranges. The plots from top to bottom are the inversions for four different models. In each plot, the velocity clamping factor changed from 0.1 to 0.9

$$\begin{aligned}
 & b_d = b_{\max,d} - b_{\min,d} \\
 & \left\{ \begin{aligned}
 & b_{\max1,d} = \max(g_{\max,d}^{\text{memo}} + s_4 r b_d, x_{\max,d} + s_4 r b_d, y_{\max,d} + s_4 r b_d) \\
 & b_{\max2,d} = \max(x_{\text{ave},d} + (s_2 + s_3 r) b_d, y_{\text{ave},d} + (s_2 + s_3 r) b_d) \\
 & b_{\max,d} = \min(b_{\max,d}^{\text{ini}}, x_{\max,d} + s_1 r b_d, \max(b_{\max1,d}, b_{\max2,d}))
 \end{aligned} \right. , \\
 & \left\{ \begin{aligned}
 & b_{\min1,d} = \min(g_{\min,d}^{\text{memo}} - s_4 r b_d, x_{\min,d} - s_4 r b_d, y_{\min,d} - s_4 r b_d) \\
 & b_{\min2,d} = \min(x_{\text{ave},d} - (s_2 + s_3 r) b_d, y_{\text{ave},d} - (s_2 + s_3 r) b_d) \\
 & b_{\min,d} = \max(b_{\min,d}^{\text{ini}}, x_{\min,d} - s_1 r b_d, \min(b_{\min1,d}, b_{\min2,d}))
 \end{aligned} \right.
 \end{aligned} \tag{10}$$

where $b_{\max,d}$, $b_{\min,d}$, and b_d are the maximum, minimum, and range values of the particles' positions, $b_{\max,d}^{\text{ini}}$ and $b_{\min,d}^{\text{ini}}$ represent the initial range, r is a random number independently generated in $[0,1]$, and s_1, s_2, s_3 and s_4 are SSS parameters. Each inversion parameter dimension exhibits a fast convergence in stage 1 because the parameters are set to $s_1 = 0.4, s_2 = 0.3, s_3 = 0.1$, and

$s_4 = 0.2$. The convergence slows in stage 2 ($s_1 = 0.5, s_2 = 0.1, s_3 = 0.6$, and $s_4 = 0.3$) with the randomness boundary extension, and increases again in stage 3 ($s_1 = 0.3, s_2 = 0.3, s_3 = 0.3$, and $s_4 = 0.2$). Then, the boundary of the dimension shrinks to a very small range that includes the global optima.

Finally, we update the boundary conditions and maximum velocity.

3.2 Numerical experiments using the SSS-PSO algorithm

As shown in Sect. 2.2, basic PSO fails to retrieve the model parameters for model 4. We applied the SSS-PSO to this problem to determine its inversion performance. The inversion results are shown in Fig. 6. The fast and robust convergence of the objective function demonstrates the algorithm's performance. In repeated inversions, the objective function converged to the termination

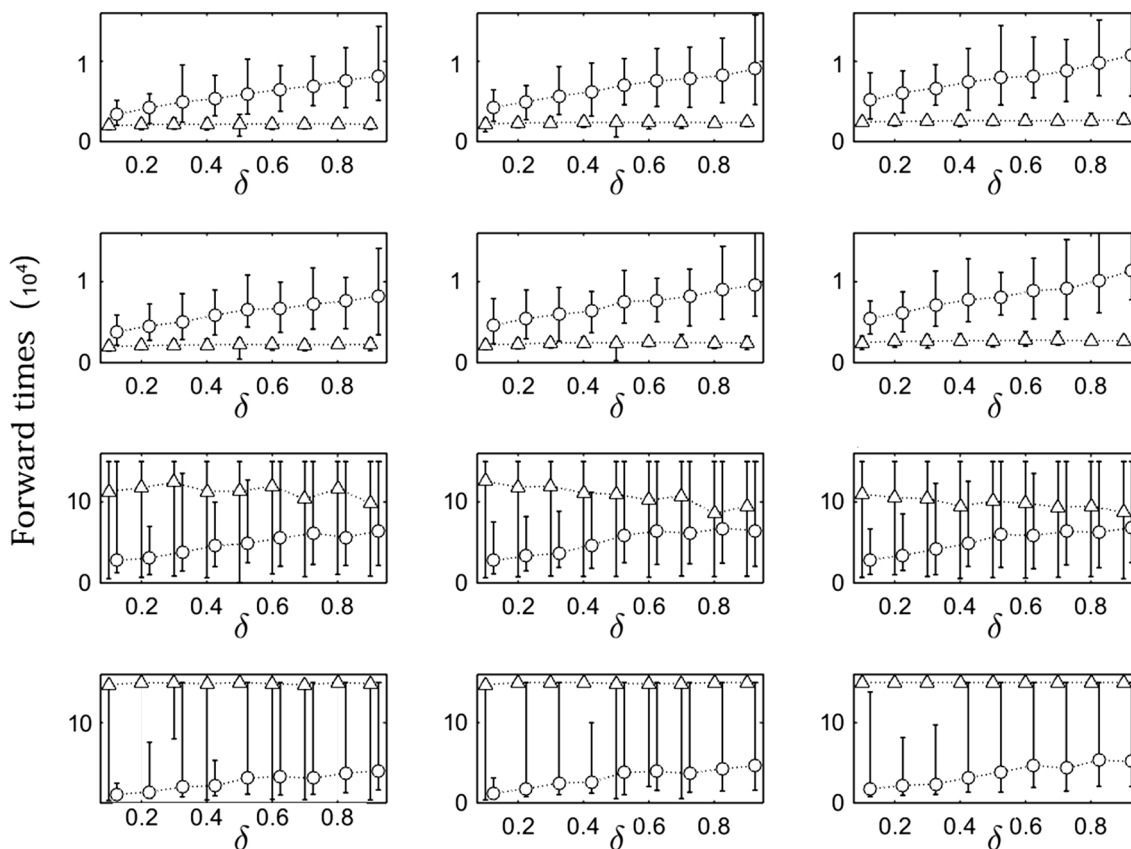


Fig. 8 Error bars of forward evaluations of different models with varying velocity clamping factors. The triangle and circles represent the basic and SSS-PSO results, respectively. The plots from left to right are the results for three different initial ranges. The plots from top to bottom are the inversions for four different models. In each plot, the velocity clamping factor changed from 0.1 to 0.9

criteria within the maximum number of iterations. It can be seen that although without requiring a prior information, the performance of the SSS-PSO is stable in repeated runs. These merits mean that SSS-PSO can be applied to velocity calibration application cases in complex models.

4 Sensitivity of PSO and SSS-PSO to velocity clamping factor

The velocity clamping factor (δ) is important to the performance of PSO and SSS-PSO. To investigate how δ affects the reliability and efficiency of PSO and SSS-PSO, we ran PSO and SSS-PSO inversions for nine δ values, ranging from 0.1 to 0.9. The inversion results are shown in Figs. 7 and 8.

We define the reliability of the inversion performance as follows:

$$\text{reliability}(\delta, \text{Succ}) = \frac{\text{Succ}_\delta}{\text{Total}} \times 100\% \tag{11}$$

where Succ_δ is the number of successful inversions in $\text{Total} = 100$ repeated runs. Success is defined as reaching the objective function value ($F = 10^{-6}$) within the maximum number of iterations ($t_{\text{max}} = 5000$). Figure 7 shows the statistical reliabilities of the four models and three initial ranges. The reliabilities for all these cases were mainly insensitive to δ , although there were some small variations for model 3. Thus, the reliabilities were almost independent of δ , regardless of the high or low reliabilities. Another important feature of these results is that the SSS-PSO reliabilities were almost 100 % in all cases, whereas the basic PSO reliability was 100 % for models 1 and 2, a little above 0 % for model 4, and approximately 50 % for model 3. This indicates that the SSS-PSO inversion was systematically superior to the basic PSO inversion, and may guarantee convergence. Moreover, the reliabilities of the SSS-PSO inversion for models 3 and 4 decreased slightly as δ increased, which suggests that we should use small δ values.

Computational efficiency is another important measure for assessing an inversion algorithm. In the basic PSO and

SSS-PSO algorithms, we only need forward calculations after updating each particle position. So the time taken by forward evaluations is equal to the number of iterations multiplied by the size of the population ($N = 30$). Forward evaluations represent the most computational time, so they correlate strongly with algorithm efficiency. Therefore, it is important to determine the sensitivity of δ to the forward evaluations.

In Fig. 8, in terms of the time taken by forward simulations, the velocity clamping factor has a marginal influence on the basic PSO efficiency, but is very important to the SSS-PSO efficiency. In the basic PSO results for simple cases (Models 1 and 2), the forward simulations are stable to the changes in the velocity clamping factor and the parameter range of initial model. In models 3 and 4, on most occasions, the method used the maximum number of forward simulations, because the statistical reliability was approximately 50 % for model 3 and 0 % for model 4. For the SSS-PSO algorithm, small values of the velocity clamping factor ($\delta \approx 1$) can significantly reduce the number of forward simulations, thus the small value of δ is a preferable.

5 Conclusions

In this study, we improved the inversion performance of the basic PSO algorithm by incorporating the SSS, and applied our method to velocity calibration in microseismic monitoring. We demonstrated that the SSS-PSO algorithm is robust and efficient. The SSS-PSO algorithm steadily and speedily converged to the solution, regardless of the complexity of the target models and the randomness of the initial models. Furthermore, we investigated the influence of choices of the velocity clamping factor (δ) on the reliability and efficiency of the basic and SSS-PSO algorithms. We found that basic PSO is insensitive to δ , whereas SSS-PSO is not (especially the searching efficiency of). The SSS-PSO algorithm was robust to all the tested values of δ . Moreover, our simulations suggest that a small velocity clamping factor ($\delta = 0.1$) is preferred for SSS-PSO, in these microseismic velocity calibration problems.

We validated the performance of the algorithms by running 100 repeated inversions with random initial models, to differentiate the discrepancies caused by the intrinsic uncertainties of geophysical inversion problems from artifacts and errors produced by the inversion strategies. Additionally, because there is no panacea for GO problems according to the “no free lunch” theorems (Wolpert and Macready 1997), we need an inversion strategy that is aimed at specific geophysical inversion problem. Any such strategies should be tested before being applied. We cannot guarantee that the SSS-PSO will converge to global optima for any given problem, which is the same for all other GO

methods. We have proposed and validated this method for microseismic velocity calibration models. The tests in this paper are limited, and further investigation is needed with noisy data, which will be proceeded in our future work.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

- Angeline PJ (1998) Evolutionary optimization versus particle swarm optimization: philosophy and performance differences, evolutionary programming VII. Springer, Berlin, pp 601–610. doi:10.1007/BFb0040811
- Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Nat Comput* 7(1):109–124. doi:10.1007/s11047-007-9050-z
- Carlisle A, Dozier G (2001) An off-the-shelf PSO. In: Proceedings of the workshop on particle swarm optimization. Purdue School of Engineering and Technology, Indianapolis, pp 1–6
- Cipolla CL, Maxwell SC, Mack MG (2012) Engineering guide to the application of microseismic interpretations. In: SPE hydraulic fracturing technology conference. Society of Petroleum Engineers, 6–8 February, The Woodlands, Texas, USA. doi:10.2118/152165-MS
- Clerc M (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Evolutionary computation*. doi:10.1109/CEC.1999.785513
- Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, pp 39–43. doi:10.1109/MHS.1995.494215
- Eberhart RC, Simpson P, Dobbins R (1996) Computational intelligence PC tools, chap. 6. Academic Press Professional Inc., Diego, CA, pp 212–226
- Eberhart RC, Shi Y (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 congress on evolutionary computation, vol. 1. IEEE Xplore, pp 84–81. doi:10.1109/CEC.2000.870279
- Gill PE, Murray W, Wright MH (1981) Practical optimization. Academic press, London, pp 1–26
- Kiranyaz S, Ince T, Gabbouj M (2013) Multidimensional particle swarm optimization for machine learning and pattern recognition. Springer, Berlin, pp 101–148
- Liang J, Qin K, Suganthan PN, Subramanian B (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans Evol Comput* 10(3):281–295. doi:10.1109/TEVC.2005.857610
- Maxwell SC, Underhill WB, Bennett L, Woerpel C, Martinez A (2010) Key criteria for a successful microseismic project, 19–22 September, Florence, Italy. doi:10.2118/134695-MS
- Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. *IEEE Trans Evol Comput* 8(3):204–210. doi:10.1109/TEVC.2004.826074
- Pei DH, Quirein JA, Cornish BE, Quinn D, Warpinski NR (2009) Velocity calibration for microseismic monitoring: a very fast simulated annealing (VFSA) approach for joint-objective optimization. *Geophysics* 74(6):WCB47–WCB55. doi:10.1190/1.3238365

- Poli R (2008) Analysis of the publications on the applications of particle swarm optimisation. *J Artif Evol Appl*. doi:[10.1155/2008/685175](https://doi.org/10.1155/2008/685175)
- Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. In: *Encyclopedia of machine learning*. Springer, pp 760–766
- Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans Evol Comput* 8(3):240–255. doi:[10.1109/TEVC.2004.826071](https://doi.org/10.1109/TEVC.2004.826071)
- Rezaee Jordehi A, Jasni J (2013) Parameter selection in particle swarm optimisation: a survey. *J Exp Theor Artif Intell* 25(4):527–542. doi:[10.1080/0952813X.2013.782348](https://doi.org/10.1080/0952813X.2013.782348)
- Sen MK, Stoffa PL (2013) *Global optimization methods in geophysical inversion*. Cambridge University Press, Cambridge, pp 1–280
- Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: *The 1998 IEEE international conference on evolutionary computational proceedings*. IEEE World Congress on Computational Intelligence, IEEE, pp 69–73
- Tarantola A (2005) *Inverse problem theory and methods for model parameter estimation*. SIAM, Philadelphia, pp 1–342
- Tian Y, Chen X (2005) A rapid and accurate two-point ray tracing method in horizontally layered velocity model. *Acta Seismol Sin* 18(2):154–161
- Warpinski NR, Du J (2013) Velocity building for microseismic hydraulic fracture mapping in isotropic and anisotropic media. In: *SPE hydraulic fracturing technology conference*. Society of Petroleum Engineers, 4–6 February, The Woodlands, Texas, USA
- Warpinski NR, Sullivan RB, Uhl J, Waltman C, Machovoe S (2005) Improved microseismic fracture mapping using perforation timing measurements for velocity calibration. *SPE J* 10(1):14–23. doi:[10.2118/84488-PA](https://doi.org/10.2118/84488-PA)
- Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1:67–82. doi:[10.1109/4235.585893](https://doi.org/10.1109/4235.585893)
- Xu SH, Rahmat-Samii Y (2007) Boundary conditions in particle swarm optimization revisited. *IEEE Trans Antennas Propag* 55(3):760–765
- Zhang L-P, Yu H-J, Hu S-X (2005a) Optimal choice of parameters for particle swarm optimization. *J Zhejiang University Sci-A* 6(6):528–534. doi:[10.1007/BF02841760](https://doi.org/10.1007/BF02841760)
- Zhang WJ, Xie XF, Bi DC (2005b) Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. In: *Proceedings of the 2004 congress on evolutionary computation IEEE*, vol 2, pp 2307–2311. doi:[10.1109/CEC.2004.133118](https://doi.org/10.1109/CEC.2004.133118)