Effectiveness of particle swarm optimization technique in dealing with noisy data in inverse heat conduction analysis

Conference Paper · July 2009

CITATIONS
0

READS
237

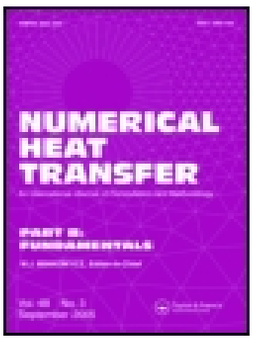
2 authors:

Soheyl Vakili
University of British Columbia - Vancouver
16 PUBLICATIONS **143** CITATIONS

SEE PROFILE

M. S. Gadala
University of British Columbia - Vancouver
145 PUBLICATIONS **2,398** CITATIONS

SEE PROFILE



Numerical Heat Transfer, Part B: Fundamentals

ISSN: 1040-7790 (Print) 1521-0626 (Online) Journal homepage: http://www.tandfonline.com/loi/unhb20

Effectiveness and Efficiency of Particle Swarm Optimization Technique in Inverse Heat Conduction Analysis

S. Vakili & M. S. Gadala

To cite this article: S. Vakili & M. S. Gadala (2009) Effectiveness and Efficiency of Particle Swarm Optimization Technique in Inverse Heat Conduction Analysis, Numerical Heat Transfer, Part B: Fundamentals, 56:2, 119-141, DOI: 10.1080/10407790903116469

To link to this article: http://dx.doi.org/10.1080/10407790903116469

Numerical Heat Transfer, Part B, 56: 119–141, 2009

ISSN: 1040-7790 print/1521-0626 online
DOI: 10.1080/10407790903116469

Taylor & Francis
Taylor & Francis Group

EFFECTIVENESS AND EFFICIENCY OF PARTICLE SWARM OPTIMIZATION TECHNIQUE IN INVERSE HEAT CONDUCTION ANALYSIS

S. Vakili and M. S. Gadala
Department of Mechanical Engineering, The University of British Columbia, Vancouver, British Columbia, Canada

Three variations of the particle swarm optimization (PSO) method are used to solve the boundary inverse heat conduction problem, in one, two, and three dimensions. Both steady and transient problems are studied. It is shown that PSO can be successfully applied to inverse heat conduction problems, and can alleviate some of the stability problems of the classical approaches. The computational costs of the three variations of PSO (basic, repulsive, and complete repulsive) are compared with each other, and with an implementation of the genetic algorithm. For these problems, some variants of PSO are proven to be more efficient than other algorithms. Also, the effectiveness of PSO in dealing with noisy domains is investigated.

1. INTRODUCTION

An inverse heat transfer problem is a problem in which the boundary conditions, initial conditions, geometry, or material properties are not fully specified. In this research, we focused mainly on finding the missing boundary conditions. These problems are called boundary inverse problems, and they have numerous applications in various fields of engineering. Because the effects of changes in boundary conditions are usually damped and lagged in the interior measurement points, the problem is typically an ill-posed one and displays a high level of sensitivity to measurement errors. In general, the uniqueness and the stability of an inverse heat conduction solution are not guaranteed [1]. This ill-posedness is commonly treated by using one or a combination of some regularization techniques, e.g., Tikhonov regularization [2], the future information method [1], or the iterative regularization technique [3].

In the last two decades, various numerical algorithms have been examined to obtain a reliable inverse heat transfer solution. The most commonly used methods are the least-square regularization method [1, 4], the sequential function specification approach [3–5], the space-marching technique [6], the conjugate gradient algorithm [7, 8], the steepest-descent method [9], the model-reduction method [10], genetic algorithms (GA) [11, 12], and artificial neural networks [13–15].

In this research, the particle swarm optimization (PSO) technique, a relatively new global optimizer, with very few applications in heat transfer literature, is applied

Received 23 March 2009; accepted 3 June 2009.

Address correspondence to M. S. Gadala, Department of Mechanical Engineering, The University of British Columbia, 2324 Main Mall, Vancouver, BC, Canada V6T-1Z4. E-mail: gadala@mech.ubc.ca

NOMENCLATURE

c_p	specific heat, J/kg°C
c_0	self-confidence parameter (inertia coefficient)
c_1, c_2, c_3, c_4	acceleration parameters
C	equivalent heat capacity matrix
f	objective function
g	best global solution
h	convection heat transfer coefficient, W/m^2°C
k	thermal conductivity, W/m°C
K	equivalent heat conductivity matrix
n	number of particles in a swarm
n_1, n_2	number of simulations done for the statistical t test
N	finite-element shape function
N_E	number of function evaluations (solutions of the direct problem)
p	best solution of a particle
q	heat flux, W
q^b	heat generation per unit volume, W/m^3
Q	equivalent thermal load vector
r	normally distributed random vector in [0, 1]
t	statistical t value
T	temperature, °C
U	value of objective function/U value for the Mann-Whitney test
v	particle velocity
x	particle position
α	regularization parameter
β	coefficient for penalizing negative temperatures
γ	coefficient for penalizing negative heat fluxes
ρ	density, kg/m^3
σ	standard deviation of the added error

Subscripts

c	related to conduction
f	fluid
h	related to convection
i	particle number in the swarm/ index number in the arrays of numbers
r	related to radiation
x, y, z	directions in Cartesian coordinate system

Superscripts

b	related to a source term
e	of an element
h	related to convection
m	iteration number in particle swarm
r	related to radiation
s	related to surface heat flux

to solve the inverse heat conduction problem. Applications of particle swarm optimization to the inverse heat conduction problem are rarely reported in the literature. Moreover, the performance of PSO and its advantages over the classical approaches have not been studied in detail. Also, the ability of PSO to deal with noisy temperature readings has not been adequately investigated. In this research, various inverse heat conduction test cases are solved using this technique. PSO is proved to be effective in some cases where the classical methods fail, and its performance is found to be superior to some other approaches.

Due to the popularity of the genetic algorithm in optimization, it has been used widely in solving inverse problems. The efficiency of PSO has been frequently compared with other heuristic search techniques, such as GA, by other researchers in several applications. Hassan et al. [16] compared the effectiveness and efficiency of the GA and PSO in solving some benchmark optimization test cases, as well as some engineering design optimization problems. They used a statistical comparison approach, similar to the one used in this research, and found that obtaining high-quality results is computationally more efficient by PSO than by GA. Mouser and Dunn [17] compared the performance of PSO and GA when applied to the optimization of a structural dynamic model. They used a GA to optimize the properties

of a PSO and a GA algorithm in order to find the best possible performance, and found that PSO outperforms the GA for their problem. However, the performance of these two optimization methods in inverse heat conduction problems has not been compared in the literature. This is one of the topics of the present research.

Other than the efficiency of the algorithm, i.e., its lower computational cost, another important aspect in choosing an inverse algorithm for a specific problem is its effectiveness. The method should be able to produce acceptable results, even when dealing with noisy data. The experimental data that are used with the inverse algorithms always include some experimental errors. In a heat transfer analysis, these errors can be due to the limited accuracy of the measurement devices, such as thermocouples. Also, the installation of the thermocouples, either on the species or inside it, will generally distort the original temperature field. Although there has been a considerable amount of research to understand these errors, and to prevent or compensate for them [18–20], such errors will remain and must be accounted for. Therefore, a practical inverse algorithm should be able to handle them, and remain stable in the presence of a reasonable amount of noise in the measurements. Particle swarm optimization has been found to be very stable and efficient in handling noise in other optimization applications [21, 22]. In fact, it seems that in some cases, the presence of noise might help PSO avoid local minima and premature convergence [21]. There is no detailed investigation in the literature of the ability of particle swarm optimization to handle noisy temperature domains. In this study, the PSO method is tested in the presence of noise in the temperature readings. It is shown to be able to handle moderate amounts of noise. Also, the effects of performance parameters that can generate a more noise-resistant version of PSO are investigated.

2. OPTIMIZATION ALGORITHMS

Three variations of the PSO algorithm are used in this study. Their performance is compared with each other and with an implementation of the GA. This section provides a brief description of these methods.

2.1. Particle Swarm Optimization

Particle swarm optimization is a recent high-performance algorithm that can also be used to solve inverse problems. The method is based on the social behavior of species in nature, e.g., a swarm of birds or a school of fish. It was originally developed by Eberhart and Kennedy in 1995 [23].

In this method, if a member of the swarm finds a desirable position, it will influence the traveling path of the rest of the swarm members. Every member searches in its vicinity, and not only learns from its own experience (obtained in the previous iterations), but also benefits from the experiences of the other members of the swarm, especially from the experience of the best performer. The original PSO algorithm includes the following components [24]:

Particle position vector x: For each particle, this vector stores its current location in the search domain. These are the values for which the value of the objective function is calculated, and the optimization problem is solved.

Particle velocity vector v: For every particle, this vector determines the magnitude and direction of change in the position of that particle in the next iteration. This is the factor that causes the particles to move around the search space.

Best solution of a particle p: For a specific particle, this is the position that has produced the lowest value of the objective function (the best solution with the lowest error in our case).

Best global solution g: This is the best single position found by all particles of the swarm, i.e., the single p point that produces the lowest value for the objective function, among all the swarm members.

The number of particles in the swarm (n) needs to be specified at the beginning. Fewer particles in the swarm means that less computational effort is needed in each iteration, but possibly a higher number of iterations is required. On the other hand, a larger population will have a higher computational expense in each iteration, but is expected to require fewer iterations to get to the global optimum point. Earlier studies have shown that a smaller population is normally preferred [25, 26]. This was also observed in the present study; however, its effect seems to be insignificant.

The steps involved in the basic PSO are detailed below [24].

1. Randomly initialize the positions and velocities for all of the particles in the swarm.
2. Evaluate the fitness of each swarm member (objective function value at each position point).
3. At iteration m, the velocity of the particle i, is updated as

$$v_i^{m+1} = c_0 v_i^m + c_1 r_1 \left(p_i^m - x_i^m\right) + c_2 r_2 \left(g^m - x_i^m\right) \tag{1}$$

where x_i^m and v_i^m are the position and velocity of particle i at the mth step, respectively; p_i^m and g^m are the best positions found up to now by this particle and by the whole swarm, respectively; c_0 is called the inertia coefficient or the self-confidence parameter and is usually between zero and one; c_1 and c_2 are the acceleration coefficients that pull the particles toward the local and global best positions; and r_1 and r_2 are random vectors in the range of (0, 1). The suggested value for c_0 is 0.7, and for c_1 and c_2 is 1.43 [24]. The ratio between these three parameters controls the effect of the previous velocities and the trade-off between the global and local exploration capabilities.

We also need a value for the maximum velocity. A rule of thumb requires that, for a given dimension, the maximum velocity, $v_i^{\max}$, should be equal to one-half the range of possible values for the search space. For example, if the search space for a specific dimension is the interval [0, 100], we will take a maximum velocity of 50 for this dimension. If the velocity obtained from Eq. (1) is higher than $v_i^{\max}$, then we will substitute the maximum velocity instead of v_i^{m+1}. The reason for having this maximum allowable velocity is to prevent the swarm from "explosion" (divergence). Another popular way of preventing divergence is a technique called "constriction," which dynamically scales the velocity update [24]. The first method is used in this research.

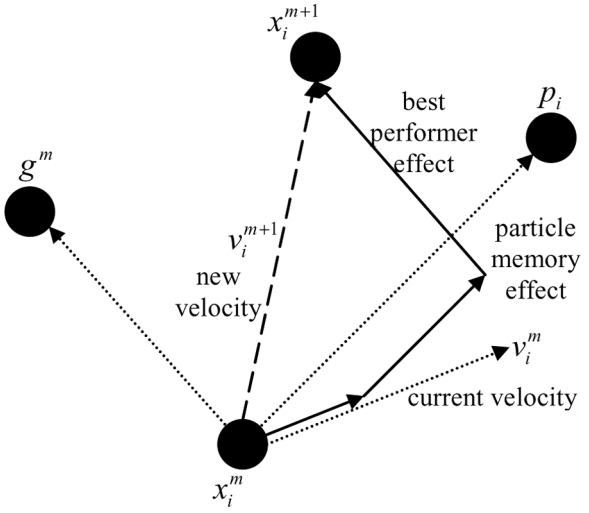


Figure 1. Velocity and position updates in the basic particle swarm optimization algorithm.

4. Update the position of each particle using the updated velocity and assuming unit time:

$$x_i^{m+1} = x_i^m + v_i^{m+1} \tag{2}$$

5. Repeat (2)–(4) until a convergence criterion (an acceptable fitness value or a certain maximum number of iterations) is satisfied.

The above steps are illustrated in Figure 1, and Figure 2 shows a flowchart of the whole process.

The above procedure explains the basic PSO algorithm. However, our implementation is somewhat different. As mentioned above, the relation between the self-confidence parameter, c_0, and the acceleration coefficients, c_1 and c_2, determines the trade-off between the local and global search capabilities. As we progress in time through iterations, we get closer to the best value. Thus, a reduction in the value of the self-confidence parameter will limit the global exploration, and a more localized search will be performed. In this research, if the value of the best objective function is not changed in a certain number of iterations (10 iterations in our case), the value of c_0 is multiplied by a number less than one (0.95 for our problems), to reduce it (i.e., $c_0^{\text{new}} = 0.95 c_0^{\text{old}}$). These numbers are based mainly on the authors' experience, and the performance is not very sensitive to their exact values. Some other researchers have used a linearly decreasing function to calculate the value of the self-confidence parameter [25]. These techniques are called *dynamic adaptation*, and are very popular in the recent implementations of PSO [27].

The random initialization component and the iterative progress of the members in PSO are similar to the GA. However, unlike GAs, a particle swarm system has a memory. Also, the knowledge of the best solution is shared among all particles. Compared to the GA, the PSO is known to be simpler to understand

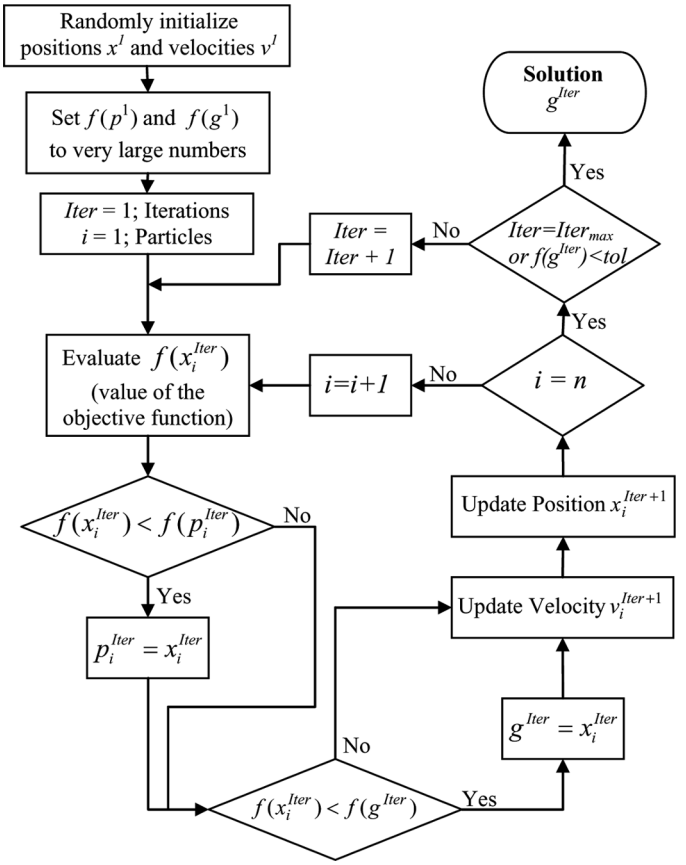


Figure 2. Flowchart of the basic particle swarm optimization procedure.

and implement, has fewer parameters to adjust, and has lower computational cost [23]. This latter point is investigated further in this research when we deal with inverse heat conduction problems. Furthermore, PSO benefits from one major advantage of GA, i.e., it does not require gradient information of the objective function. Due to its advantages, PSO has been successfully applied to a wide range of practical engineering problems in recent years [24, 28].

2.2. Repulsive Particle Swarm Optimization

Unfortunately, the basic PSO may get trapped in a local minimum, which can result in a slow convergence rate or even premature convergence, especially for complex problems with many local optima. Therefore, several variants of PSO have been developed to improve the performance of the basic algorithm [28]. Some variants try to add a chaotic acceleration factor to the position update equation, in order to prevent the algorithm from being trapped in local minima [29]. Others try to modify the velocity update equation to achieve this goal. One of these variants

is called *repulsive particle swarm optimization* (RPSO), and is based on the idea that repulsion between the particles can be effective in finding the global minimum [30, 31]. The velocity update equation for RPSO is

$$v_i^{m+1} = c_0 v_i^m + c_1 r_1 \left(p_i^m - x_i^m\right) + c_2 r_2 \left(p_j^m - x_i^m\right) + c_3 r_3 v_r \tag{3}$$

where p_j^m is the best position of a randomly chosen other particle among the swarm, c_3 is an acceleration coefficient, r_3 is a random vector in the range of (0, 1), and v_r is a random velocity component. Here c_2 is -1.43, and c_3 is 0.5. These values are based on recommendations by Clerc [24]. Based on our own experience, minor changes in these values do not significantly affect the performance.

The newly introduced third term on the right-hand side of Eq. (3), with always a negative coefficient (c_2), causes a repulsion between the particle and the best position of a randomly chosen other particle. Its role is to prevent the population from being trapped in a local minimum. The fourth term generates noise in the particle's velocity in order to take the exploration to new areas in the search space. Once again, we are gradually decreasing the value of the self-confidence parameter. Note that the third term on the right-hand side of Eq. (1), i.e., the tendency toward the global best position, is not included in a repulsive particle swarm algorithm in most of the literature. We have not applied any other changes to the original method.

2.3. Complete Repulsive Particle Swarm Optimization

The repulsive particle swarm optimization technique, as mentioned in the previous section and as used in the literature, does not benefit from the global best position found. A modification to RPSO is introduced in this research, which also uses the tendency toward the best global point. We call it *complete repulsive particle swarm optimization* (CRPSO). The velocity update equation for CPRSO is

$$v_i^{m+1} = c_0 v_i^m + c_1 r_1 \left(p_i^m - x_i^m\right) + c_2 r_2 \left(g_i^m - x_i^m\right) + c_3 r_3 \left(p_j^m - x_i^m\right) + c_4 r_4 v_r \tag{4}$$

By having both an attraction toward the particle's best performance and a repulsion from the best performance of a random particle, we are trying to create a balance between the local and global search operations.

2.4. Genetic Algorithm

The genetic algorithm is probably the most popular stochastic optimization method. It is also widely used in many heat transfer applications, including inverse heat transfer analysis [32]. Similar to PSO, the GA starts its search from a randomly generated population. This population evolves over successive generations (iterations) by applying three major operations. The first operation is *selection*, which mimics the principle of "survival of the fittest" in nature. It finds the members of the population with the best performance, and assigns them to generate the new members for future generations. This is basically a sort procedure based on the

obtained values of the objective function. The number of elite members that are chosen to be the parents of the next generation is also an important parameter. Usually, a small fraction of the less fit solutions are also included in the selection, to increase the global capability of the search and prevent premature convergence. The second operator is called *reproduction* or *crossover*, which imitates mating and reproduction in biological populations. It propagates the good features of the parent generation into the offspring population. In numerical applications, this can be done in several ways. One way is to have each part of the array coming from one parent. This is normally used in binary-encoded algorithms. Another method that is more popular in real encoded algorithms is to use a weighted average of the parents to produce the children. The latter approach in used in this research. The last operator is *mutation*, which allows for global search of the best features, by applying random changes in random members of the generation. This operation is crucial in avoiding the local minima traps. More details about the algorithm may be found in [33, 34].

Among the many variations of GAs, in this study, we use a real encoded GA with roulette selection, intermediate crossover, and uniform high-rate mutation [33]. The crossover probability is 0.2, and the probability of adjustment mutation is 0.9. These settings were found to be the most effective based on our experience with this problem. A mutation rate of 0.9 may seem higher than normal. This is because we start the process with a random initial guess, which needs a higher global search capability. However, if a smarter initial guess is utilized, a lower rate of mutation may be more effective. Genes in the present application of the GA consist of arrays of real numbers, with each number representing the value of the heat flux at a certain time step, or a spatial location.

3. DIRECT CONDUCTION HEAT TRANSFER ANALYSIS FORMULATION

In this study, the finite-element method is used to solve the direct (forward) heat conduction problem. A brief description of the heat conduction equations and the discretized finite-element equations is given below for completeness. More detailed accounts may be found in [35–37].

3.1. Governing Equation and Boundary Conditions

The general governing equation for the 3-D conduction heat transfer problem can be written as

$$\frac{\partial}{\partial x}\left(k_x \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k_y \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(k_z \frac{\partial T}{\partial z}\right) + q^b = c_p \rho \frac{\partial T}{\partial t} \tag{5}$$

where T is the temperature (°C); q^b is the heat generation per unit volume (W/m^3); k_x, k_y, and k_z are the conductivities in the x, y, and z directions, respectively (W/m°C); ρ is the density (kg/m^3); c_p is the specific heat (J/kg°C); t is the time (s); and x, y, and z are the Cartesian coordinates of a point.

The boundary conditions may be one or a combination of the followings cases: prescribed temperature, prescribed heat flux, convective heat exchange, and/or radiation.

3.2. Finite-Element Formulation

We assume an approximation function for the temperature given by

$$T(\mathbf{x}) = N_i(\mathbf{x}) T_i \tag{6}$$

where $N_i(\mathbf{x})$ is the shape function with i varying from one to the number of nodes per element; the vector $\mathbf{x}$ has the components x, y, and z; and T_i are the nodal temperatures. Using a weighted residual Galerkin procedure, the final finite-element equations may be written as

$$[C] \cdot \left\{ \dot{T}^e \right\} + [[K_c] + [K_h] + [K_r]] \cdot \{T^e\} = \{Q\}^b + \{Q\}^s + \{Q\}^h + \{Q\}^r \tag{7}$$

where $\mathbf{C}$ is the equivalent heat capacity matrix; $\mathbf{K}$ is the equivalent heat conductivity matrix; $\mathbf{T}$ and $\dot{\mathbf{T}}$ are vectors of the nodal temperature and its derivative with respect to time, respectively; and $\mathbf{Q}$ is the equivalent load vector. Detailed expressions of the matrices can be found in the literature [35, 37].

A general family of solution algorithms may be obtained by introducing a parameter θ between zero and one, such that

$${}^{t+\theta\Delta t}\dot{\mathbf{T}} = \frac{1}{\Delta t}\left({}^{t+\Delta t}\mathbf{T} - {}^{t}\mathbf{T}\right) = \frac{1}{\theta \Delta t}\left({}^{t+\theta\Delta t}\mathbf{T} - {}^{t}\mathbf{T}\right) \tag{8}$$

$${}^{t+\theta\Delta t}\mathbf{T} = \theta\, {}^{t+\Delta t}\mathbf{T} + (1-\theta)\, {}^{t}\mathbf{T} \tag{9}$$

If $\theta = 0$, an explicit Euler forward method is obtained; if $\theta = 1/2$, an implicit trapezoidal rule is obtained; and if $\theta = 1$, an implicit Euler backward method is obtained. Substituting Eqs. (8) and (9) into Eq. (7) and applying Newton-Raphson iteration yields

$$\begin{aligned} {}^{t+\theta\Delta t}\left[\mathbf{K} + \left(\frac{1}{\theta\, \Delta t}\right) \cdot \mathbf{C}\right]^{(i-1)} \Delta \mathbf{T}^{(i)} = {}& {}^{t+\theta\, \Delta t}\left(\mathbf{Q}^b + \mathbf{Q}^s\right) + {}^{t+\theta\, \Delta t}\left(\hat{\mathbf{Q}}^h + \hat{\mathbf{Q}}^r\right)^{(i-1)} \\ & - {}^{t+\theta\, \Delta t}\left(\hat{\mathbf{Q}}^c + \hat{\mathbf{q}}^c\right)^{(i-1)} \end{aligned} \tag{10}$$

where $\theta \neq 0$. Depending on the value of θ, the procedure may be either conditionally stable ($\theta < 0.5$) or unconditionally stable ($\theta \geq 0.5$).

4. RESULTS AND DISCUSSION

Three test cases are set up and solved in this section. They are all *function specification* inverse problems, i.e., the missing boundary condition is an unknown function of space and/or time. The first and second cases (1-D and 2-D, respectively)

are transient, whereas the third one is a 3-D steady-state problem. The results obtained by PSO are presented, and in some cases are compared with the classical approaches. Also, the computational costs of the three variations of PSO are compared with each other for a given level of accuracy, and with the computational cost of the GA. Since the performance of these methods is affected by the choice of their parameters, it is very difficult to consider all possible combinations. The results that are presented in the following subsections were obtained by setting the parameters to the values that were mentioned in Section 2. However, the results are not heavily dependent on these values, and the trends will remain unchanged if a different set of reasonable parameters is used. Finally, these problems are solved again in the presence of noise. A random error with a normal distribution is added to the measured temperatures. PSO is then applied to restore the missing boundary condition. Its ability to handle the noise is measured, and its performance is improved by tuning some of the parameters.

While inverse heat conduction analysis has many applications in several aspects of engineering, the one that is particularly interesting for the authors is the design of an optimum controlled cooling process for the steel strips on the run-out table in a steel mill [37]. Using circular water jets, the steel strips are cooled down from around 1,000°C to a coiling temperature. The controlled cooling procedure is responsible for the microstructural properties of the steel strip, such as grain size, and hence is the determining factor in its mechanical properties, such as toughness and strength. In order to effectively realize the controlled cooling process, it is critical to obtain the precise heat transfer coefficients or heat flux values, and determine their distribution over the surface of the plate and over time. Some of the boundary conditions and material properties of the test cases in the following subsections are chosen to simulate this application for the steel industry. However, the performance and stability results are general, and should be applicable to other applications of inverse heat conduction analysis.

All of the cases were tested on a PC with an Intel Pentium D 2.80-GHz CPU, with 2 GB of RAM, and running Microsoft Windows XP Professional.

4.1. Test Case I: 1-D Transient Problem

PSO is applied to a 1-D inverse heat conduction problem to prove its capability in solving such problems. In this case, the triangular heat flux history, popularized by Beck et al. [1], is used as the input heat flux at the left end of the sample. For simplicity, the parameters are taken as $k = \rho c = L = 1$. The heat flux history is in the following form:

$$q(t) = \begin{cases} 0 & t < 0.24 \\ t - 0.24 & 0.24 \le t < 0.84 \\ -t + 1.44 & 0.84 \le t < 1.44 \\ 0 & t \ge 1.44 \end{cases} \tag{11}$$

Figure 3 shows the geometry of the problem, and its boundary conditions.

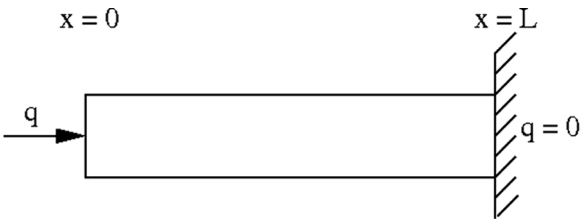


Figure 3. Problem geometry and boundary conditions.

The direct problem is solved and the temperatures at the right end ($x = L$) are obtained and stored. The goal is to minimize the difference between these temperatures (simulated measurements) and those that can be obtained by applying an arbitrary input heat flux. The most obvious choice of the objective function is some kind of norm of the difference between the measured temperatures and the calculated temperatures obtained by using the guessed heat flux values. However, it is always beneficial to include more constraints from our knowledge of the answer. Here, for example, we know that negative temperature and heat flux are not acceptable, so we can penalize those answers by adding some terms to the objective function. Another modification that is more challenging is to use some sort of regularization, i.e., penalizing the oscillations in the results. Here, we decided to penalize the second derivative of the heat flux, so the objective function will be

$$U = \sqrt{\sum_{i=1}^{n} (T_{\text{measured}_i} - T_{\text{calculated}_i})^2} + \gamma \sum_{i=1}^{n} [|\text{sign}(T_{\text{calculated}_i}) - 1| \cdot T_{\text{calculated}_i}] + \beta \sum_{i=1}^{n} [|\text{sign}(q_i) - 1| \cdot q_i] + \alpha \sum_{i=2}^{n-1} |q_{i+1} - 2q_i + q_{i-1}| \tag{12}$$

where T_{measured_i} and $T_{\text{calculated}_i}$ are the expected and calculated temperatures at the internal node i, respectively; γ and β are coefficients for penalty terms for negative temperature and heat flux, respectively; α is the regularization coefficient; and q is the boundary heat flux component.

The PSO method was applied to the above problem. Figure 4*a* shows the results after 200 iterations with no regularization term. Figure 4*b* shows the results after adding the regularization term and the same number of iterations. As can be seen from the figures, the quality of results improved considerably after the addition of the regularization term. One should notice that this improvement is not a feature of PSO, but is a very common feature for inverse problems, regardless of the solution method [1].

The three variants of PSO produced very similar results, so only the results of the basic algorithm are presented here. However, their computational costs were different, and will be discussed later in Section 4.4.

4.2. Test Case II: 2-D Transient Problem

A transient 2-D problem is solved using both the PSO method and a sequential function specification method [38]. The heat flux at the top surface (see Figure 5*a*) is

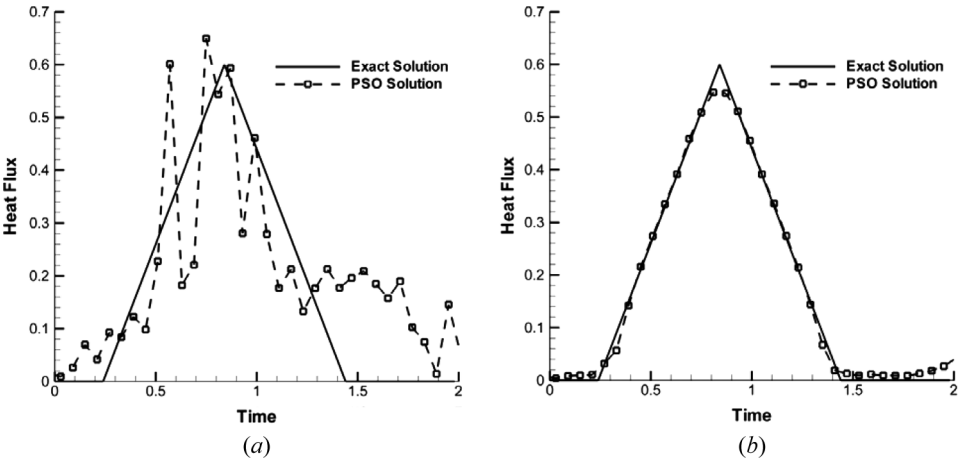


Figure 4. PSO results: (*a*) without regularization (L_2 norm of error = 0.127); (*b*) with regularization (L_2 norm of error = 0.014).

chosen to resemble the one in water cooling of steel strips on the run-out table in a steel mill. In these experiments, in order to determine the surface heat flux, or the heat transfer coefficient during the water cooling process, thermocouples are placed on the opposite side of the plate, or within the plate and 1.0 mm below the top surface, as depicted in Figure 5*b*. Inverse analysis is then used to obtain the boundary conditions on the surface, based on the readings of these internal thermocouples. As shown in Figure 5*b*, the thermocouple is located in a blind hole, with a diameter of 1.6 mm that is drilled from the bottom surface of the plate. The measurement junction of the thermocouple is fixed to the end surface of the hole, which is about

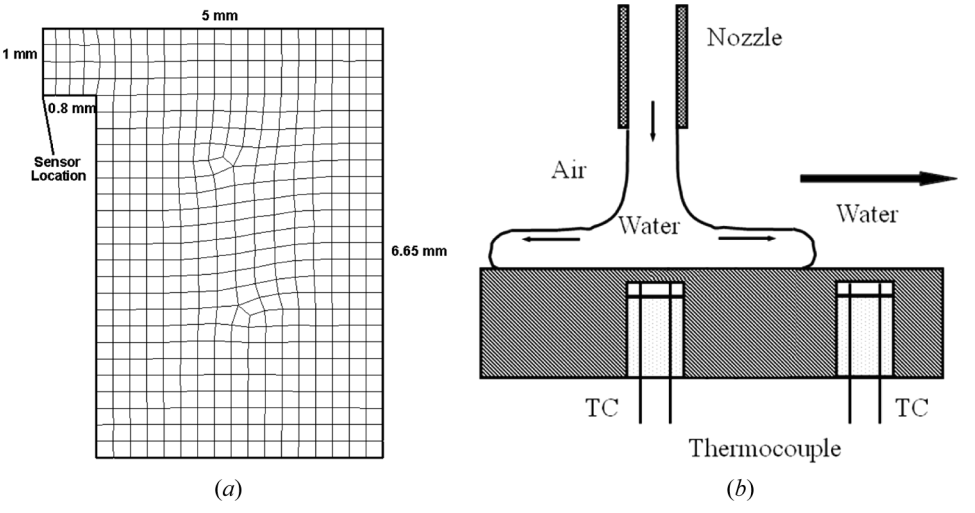


Figure 5. Case II: (*a*) model; (*b*) problem description [36].

1 mm below the top surface of the plate. The problem is solved as an axisymmetric case. The boundary condition of the top surface is prescribed heat flux. The other boundaries are assumed to be adiabatic. The left boundary is axisymmetric. The density ρ is 7,850 kg/m^3, C_p is 475 J/kg K, and the thermal conductivity k is 44.5 W/m K. These are the physical properties of the steel strips that are used in our controlled cooling experiment. Results are obtained at a point inside the object, which is the assumed position of a thermocouple. Inverse analysis is conducted to obtain the transient heat flux profile at the top surface.

The objective function here is again the norm of the difference between the measured temperatures and the calculated temperatures at the location of the thermocouple, obtained by direct simulation and a guessed value of the heat flux. Based on previous experience [36], a zero-order regularization term is preferred. The objective function can then be written as

$$U = \sum_{i=1}^{N} \left(T_{\text{measured}_i} - T_{\text{calculated}_i}\right)^2 + \alpha \sum_{i=1}^{N} q_i^2 \tag{13}$$

The same form of objective function is used for both the classical method and the particle swarm optimization technique. The value of α is normally between 10^{-12} and 10^{-10}. A more detailed study of the value of α is presented in Section 4.5. One of the main problems of the classical approaches, such as the sequential function specification method used in this example, is their instability when small time steps are used. Unlike direct problems, where the stability requirement gives the upper limit of the time-step size, in inverse problems the time step is bounded from below. Figure 6*a* shows the oscillation in the results obtained by the function specification method and a time-step size of 0.01 s, which corresponds to the onset of instability. For time steps smaller than this, the whole process diverges. PSO successfully produces, however, the results for the same time step size as presented in

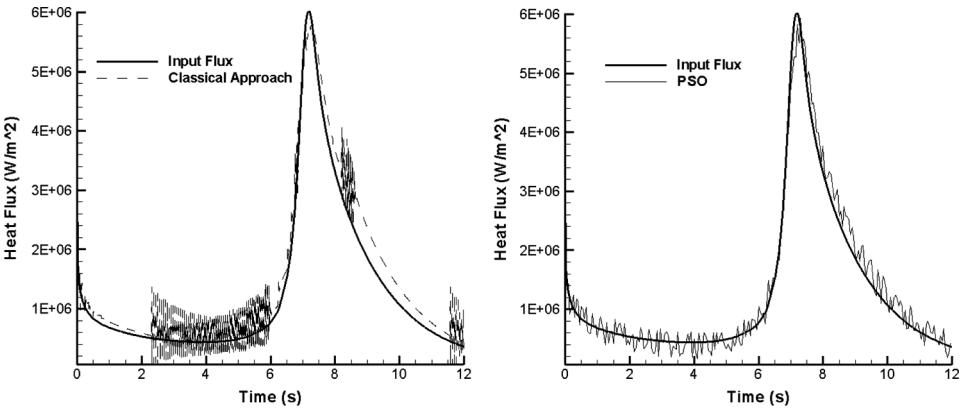


Figure 6. Heat flux verus time: (*a*) classical approach (L_2 norm of error = 3.3e5); (*b*) PSO (L_2 norm of error = 1.6e5).

Figure 6*b*. Note that the oscillations here are not due to the instability caused by the time-step size, and can be improved by performing more iterations. It is, important to mention however, that the time requirements for PSO are much higher than those for the classical function specification approaches. Like all other stochastical search techniques, PSO requires more time than the gradient-based methods. Again, the results are very similar for the three variations of PSO and GA. However, the computational costs are different, and are is discussed in Section 4.4.

4.3. Test Case III: 3-D Steady Problem

The third test case is a 3-D steady heat conduction problem in a slab, with dimensions of 3, 1, and 10, in the x, y, and z directions, respectively. The top surface of the slab ($y = 1$) is subjected to a specified temperature boundary condition that varies as $T_{\text{top}} = 25x + 100z + 10$. The other boundaries are adiabatic. The material properties are the same as in the previous test case. Figure 7 displays the temperature distribution inside the slab, obtained by 3-D finite-element modeling, as described in Section 3.2.

Unlike the previous test cases, where there was only one data point in the solution field, there are three sensors located inside the domain in this problem. They are located at the centerline with respect to the x and y coordinates ($y = 0.5$, $x = 1.5$), and at the z locations of 2, 5, and 8.

Again, the results are very similar for the three variations of PSO and GA. However, the computational costs are different, as discussed in Section 4.4. The results of the inverse analysis (temperature contours on the top surface) are shown in Figure 8 as dotted lines, along with the expected results, shown as solid lines in the same figure. Figure 9 shows the expected temperature distribution along the $x = 1.5$ (center line) on the top surface ($y = 1$), along with the temperature distribution that is restored by PSO.

4.4. Comparison of Efficiency

In this section, the performance of the three variations of PSO, introduced in Section 2, is compared with each other and with an implementation of the GA. The six different comparison tests performed are presented in Table 1. Because of the

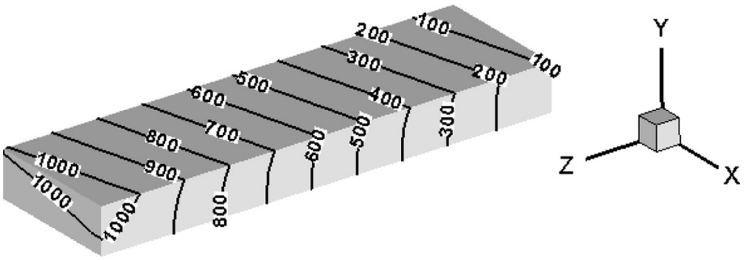


Figure 7. Temperature distribution inside the slab.

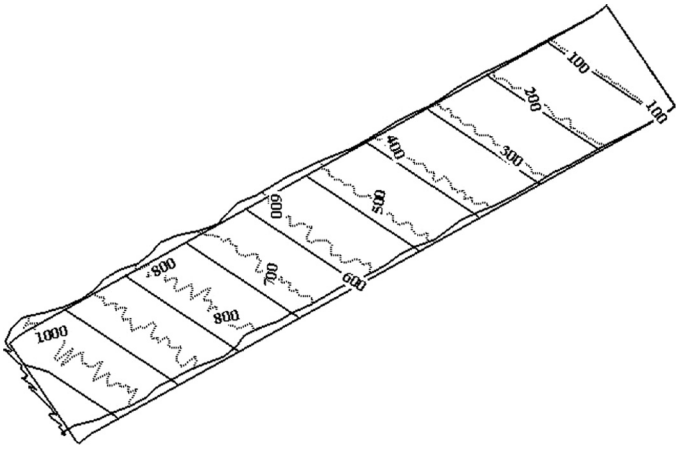


Figure 8. Top plane ($y = 1$) temperature contours: solid lines; expected input; dotted lines; output of the inverse analysis.

stochastic nature of these methods, they may perform differently for various runs. Therefore, a statistical approach is needed to compare their performances.

The statistical t test [39] is used to compare the number of function evaluation calls (the number of times the direct solver is executed) of 10 runs of each algorithm, which means a sample size of 10 is used. The purpose of performing this test is to determine whether the difference between our two means is large in comparison with

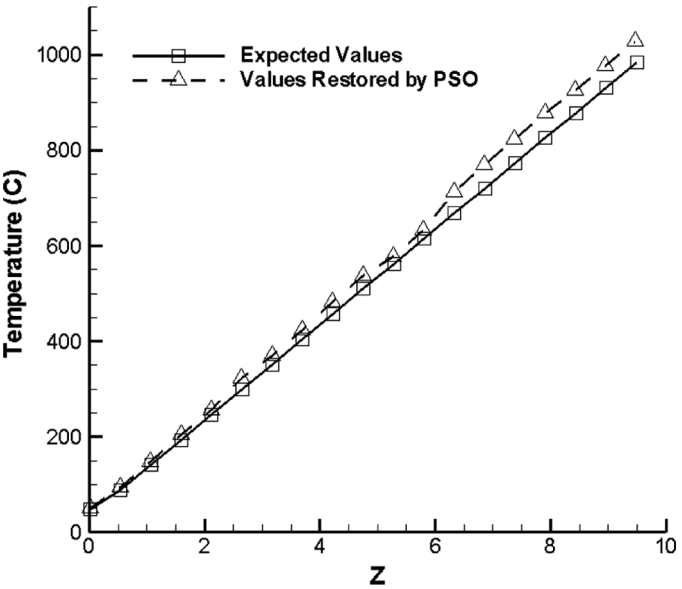


Figure 9. Temperature distribution (expected and restored by PSO) on the top surface ($y = 1$) along the $x = 1.5$ line.

Table 1. Performed comparison tests

Test no.	1	2	3	4	5	6
Method 1	GA	GA	GA	PSO	PSO	RPSO
Method 2	PSO	RPSO	CRPSO	RPSO	CRPSO	CRPSO

the standard deviation. All of the runs used in the performance evaluation test were tested for the accuracy of the results, and have proven to be able to produce acceptable results (as shown in the previous subsections).

If we represent the number of function evaluations for methods 1 and 2 by N_{E1} and N_{E2}, respectively, then the average number of function evaluations for method 1 is

$$\overline{N_{E1}} = \frac{\sum_{i=1}^{n_1} N_{E1_i}}{n_1} \tag{14}$$

where n_1 is the number of executions of method 1 (in these examples, $n_1 = n_2 = 10$). The same equation holds for method 2. The objective of the test is to check whether, generally, $N_{E2} < N_{E1}$. The null hypothesis in this test will be $N_{E2} \geq N_{E1}$. The test is a one-sided test of significance of comparison of two means [39], with a significance level of 5%, and $10 + 10 - 2 = 18$ degrees of freedom, which gives a critical t value of 1.73.

Table 2 shows the results of the comparison tests. The shaded values show the cases where the null hypothesis is accepted, i.e., the second method does not perform better than the first one. In the other cases, using the second method improves the performance, i.e., reduces the number of function evaluations, in at least 95% of occurrences.

Since an important requirement for the validity of the t-test results is the normality of the distributions for both categories that are compared, and it normally requires a larger number of simulations, the t-test statistics may not be a perfectly sound comparison tool. Thus, we have also used the Mann-Whitney nonparametric statistical test [39] to investigate our results. Table 3 shows the U values for the Mann-Whitney U test. Again, the lowest U values occur at the same cases as the t test, and only these two are not acceptable with a significance level of 5%, for a one-sided Mann-Whitney U test.

Table 2. Calculated t values for the combinations of two solution methods, three test cases; $t_{\text{critical}} = 1.73$

	Test Case 1	Test Case 2	Test Case 3
Test 1	4.88	10.08	8.26
Test 2	6.60	12.52	9.41
Test 3	10.78	17.76	13.41
Test 4	1.49	3.32	1.27
Test 5	4.63	7.59	7.92
Test 6	3.04	3.45	7.75

Table 3. Calculated U values for the combinations of two solution methods, three test cases

	Test Case 1	Test Case 2	Test Case 3
Test 1	97	100	100
Test 2	100	100	100
Test 3	100	100	100
Test 4	72	87.5	62
Test 5	94	100	100
Test 6	84	88	99

The following observations can be made regarding the results:

1. PSO variations are generally less computationally expensive than the GA in solving inverse heat conduction problems.
2. The advantage of PSO over the GA is more pronounced in more complex cases (more unknowns and fewer data points).
3. RPSO improves the performance of PSO only in the 2-D case, and even in that case the improvement level is not significant.
4. CRPSO generally performs slightly better than RPSO.

Table 4 shows the required number of computational units (i.e., solutions of the direct problem, and obtaining the norm of difference between that solution and the expected one), the relative speed-up of each variation of PSO with respect to the GA, and the relative computational cost of each method. From the table, speed-ups of up to 36% are possible when using some PSO variations instead of the GA in an inverse heat conduction problem. Table 5 shows the required time to perform one computational unit for each test case. Note that these values have no impact on the performance comparison of the tested optimization algorithms, and are similar for all the optimization methods.

The difference between the behaviors of PSO and the GA can be attributed to the different underlying mechanisms of these two evolutionary algorithms. For example, mutation in the GA occurs in all directions, while the mutation-like behavior in PSO is directional. These differences should be studied in more detail and the advantages of each should be exploited, in order to devise more effective solution approaches.

Table 4. Computational cost (number of computational units) of different solution methods, their speed-up with respect to GA, and their relative cost with respect to the most efficient method

	Test Case 1			Test Case 2			Test Case 3		
	No. of direct calcs.	Speed-up	Rel. cost	No. of direct calcs.	Speed-up	Rel. cost	No. of direct calcs.	Speed-up	Rel. cost
GA	5075	1.00	1.22	13499	1.00	1.36	23903	1.00	1.22
PSO	4582	1.11	1.10	11382	1.19	1.14	21119	1.13	1.08
RPSO	4427	1.15	1.06	10659	1.27	1.07	20872	1.15	1.06
CRPSO	4155	1.22	1.00	9939	1.36	1.00	19611	1.22	1.00

Table 5. Time required to perform one computational unit for each test case

	Test Case 1	Test Case 2	Test Case 3
Time for one computational unit (s)	1.65	10.82	5.41

4.5. Effectiveness in Handling Noisy Data

To study the effects of measurement errors in internal temperatures on the inversely calculated boundary conditions, random errors are imposed onto the calculated exact internal temperatures with the following equation:

$$T_m = T_{\text{exact}} + \sigma \cdot r \tag{15}$$

where T_m is the virtual internal temperature that is used in the inverse calculations instead of the exact temperature, T_{exact}; r is a normally distributed random variable with zero mean and unit standard deviation; and σ is the standard deviation. In this work, the maximum additive random error is $\pm 3^{\circ}\text{C}$.

There are several ways to make an inverse algorithm more stable when dealing with noisy data. For example, Gadala and Xu [38] have shown that increasing the number of "future time steps" in their sequential function specification algorithm resulted in greater stability. They have also demonstrated that increasing the regularization parameter α improves the ability of the algorithm to handle noisy data. However, the latter approach was shown to greatly increase the required number of iterations, and in many cases the solution may diverge. In this work, we first examine the effect of the regularization parameter, and then investigate an approach unique to the PSO method, to improve the effectiveness of the inverse algorithm in dealing with noise.

Figure 10 shows the effect of varying the regularization parameter value on the reconstructed heat flux, using the basic particle swarm optimization technique. Stable and accurate results are obtained for a range of values of $\alpha = 10^{-10}$ to 10^{-12}. These results are very close to those reported in [38], i.e., the proper values of α are very similar for the sequential function specification approach and PSO.

Another factor that can affect the performance of a PSO inverse approach in dealing with noisy data is the value of the self-confidence parameter c_0, or the ratio between this parameter and the acceleration coefficients. To show the effect of this parameter, the 2-D test case is again studied, with the same level of noise as in Figure 10, and a regularization parameter equal to 10^{-10} (Figure 10*a*). The acceleration coefficients are set to the default value of 1.42. The initial value of the self-confidence parameter, c_0, is changed from the default value of 0.7. The results are shown below.

As can be seen in Figure 11 (for $\alpha = 10^{-10}$), and more quantitatively in Table 6 (for $\alpha = 10^{-10}$ and 10^{-12}), increasing the value of the self-confidence parameter results in better handling of the noisy data. This trend was observed for values up to approximately 1.3, after which the results become worse, and will diverge.

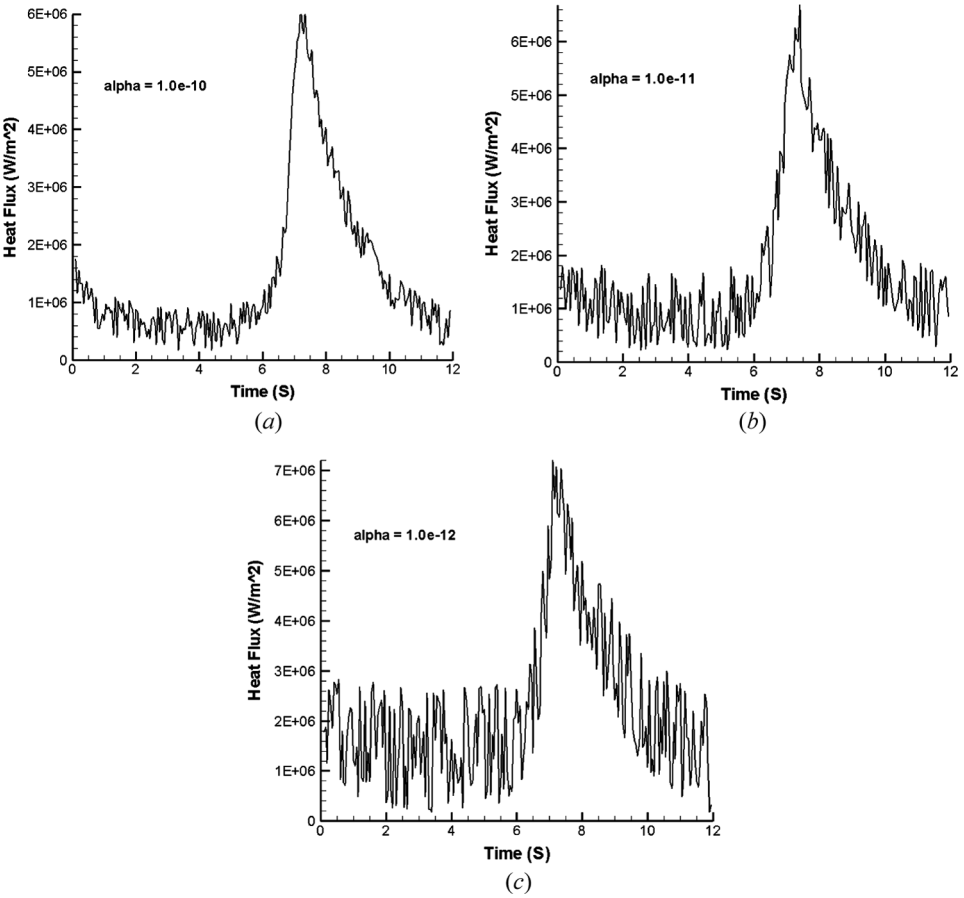


Figure 10. Effect of regularization parameter: (*a*) $\alpha = 10^{-10}$; (*b*) $\alpha = 10^{-11}$; (*c*) $\alpha = 10^{-12}$.

The same trend was also observed in the other test cases. One possible explanation is that increasing the ratio of the self-confidence parameter with respect to the acceleration coefficients results in a more global search in the domain, and therefore increases the capability of the method to escape from the local minima caused by the noise, and find values closer to the global minimum. This effect was observed to be weaker in highly noisy domains. However, in the presence of a moderate amount of noise (which is the case in most experimental setups), increasing the self-confidence ratio results in more effectiveness.

With the same set of performance parameters optimized from the previous part, the three variations of PSO were applied to the test cases where an error was introduced into the domain. Table 7 shows the effectiveness of these methods, with an initial self-confidence factor of 1.2. As can be seen, the best effectiveness is obtained by RPSO, followed closely by CRPSO. Considering the higher efficiency of CRPSO, it is still recommended for inverse heat conduction analysis.

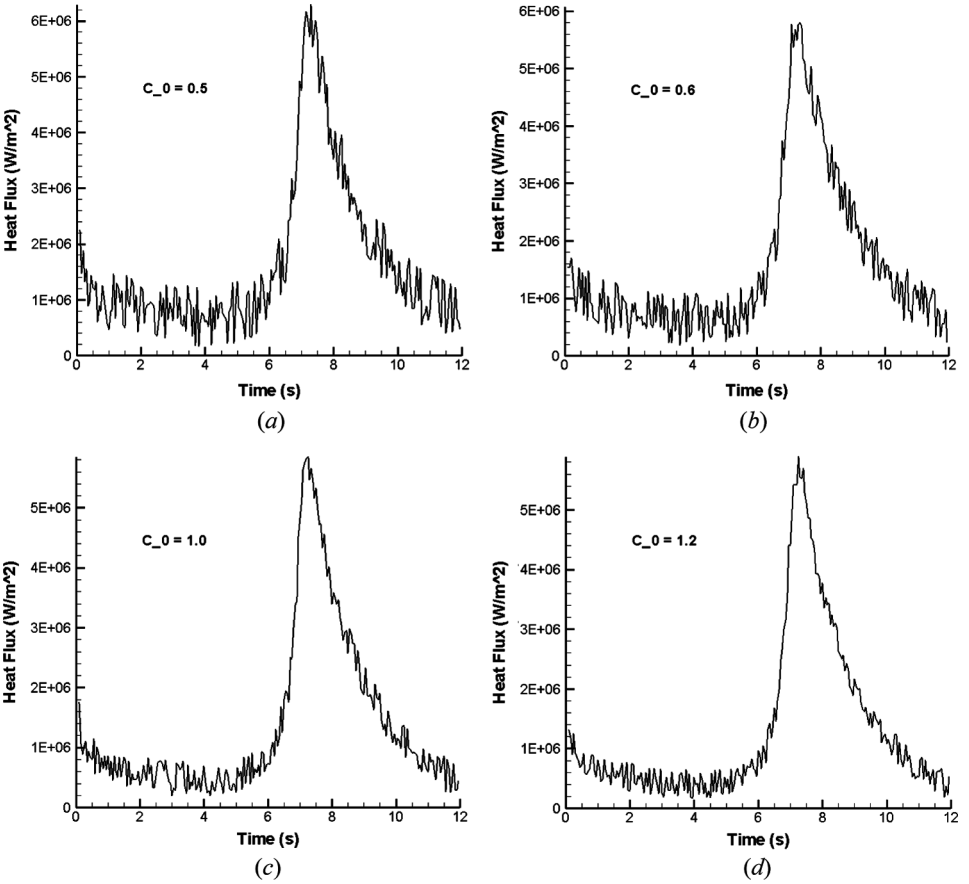


Figure 11. Effect of self-confidence parameter: (*a*) $c_0 = 0.5$; (*b*) $c_0 = 0.6$; (*c*) $c_0 = 1.0$; (*d*) $c_0 = 1.2$.

Table 6. Effect of self-confidence parameter: L_2 norm of the error (W/m^2)

C_0	0.5	0.6	0.7	1.0	1.2
$\alpha = 10^{-12}$	5.88e + 5	5.24e + 5	4.47e + 5	3.81e + 5	2.98e + 5
$\alpha = 10^{-10}$	4.85e + 5	4.30e + 5	3.19e + 5	2.36e + 5	2.10e + 5

Table 7. Effectiveness of PSO variants in dealing with noisy data

PSO variation	PSO	RPSO	CRPSO
L_2 norm of the error (W/m^2)	2.10e + 5	1.95e + 5	1.98e + 5

5. CONCLUSION

Particle swarm optimization is a new evolutionary-based global optimizer that has found popularity in many engineering applications. In this research, PSO was successfully applied to solve the inverse steady and transient heat conduction problems in one, two, and three dimensions. It was also found to mitigate the stability problem of classical inverse analysis approaches in dealing with smaller time steps. The efficiency of three variations of PSO was compared with that of an implementation of the GA. PSO was found to improve efficiency (i.e., reduce the required computational effort), especially in more complex test cases. A variation of the basic PSO, called CRPSO in this research, showed the best performance among the three versions. The effectiveness of PSO was also studied in the presence of noise. PSO proved to be effective in handling noisy data, especially when its performance parameters were tuned. Proper choice of the regularization parameter helped PSO deal with noisy data, similar to the way it helps in the classical function specification approaches. An increase in the self-confidence parameter was also found to be effective, as it increases the global search capabilities of the algorithm. RPSO was the most effective variation in dealing with noise, followed closely by CRPSO. The latter variation is recommended for inverse heat conduction problems, as it combines the efficiency and effectiveness required by these problems.

REFERENCES

1. J. V. Beck, B. Blackwell, and C. R. S. Clair, Jr., *Inverse Heat Conduction: Ill-Posed Problem*, Wiley-Interscience, New York, 1985.
2. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, Wiley, New York, 1977.
3. O. M. Alifanov, *Inverse Heat Transfer Problems*, Springer-Verlag, Berlin, 1995.
4. J. V. Beck, B. Blackwell, and A. Haji-Sheikh, Comparison of Some Inverse Heat Conduction Methods using Experimental Data, *Int. J. Heat Mass Transfer*, vol. 39, pp. 3649–3657, 1996.
5. G. Blanc, M. Raynaud, and T. H. Chau, A Guide for the Use of the Function Specification Method for 2D Inverse Heat Conduction Problems, *Rev. Gen. Therm.*, vol. 37, pp. 17–30, 1998.
6. N. Al-Khalidy, A General Space Marching Algorithm for the Solution of Two-Dimensional Boundary Inverse Heat Conduction Problems, *Numer. Heat Transfer, B*, vol. 34, pp. 339–360, 1998.
7. R. Abou Khachfe and Y. Jarny, Determination of Heat Sources and Heat Transfer Coefficient for Two-Dimensional Heat Flow–Numerical and Experimental Study, *Int. J. Heat Mass Transfer*, vol. 44, pp. 1309–1322, 2001.
8. C. H. Huang and S. P. Wang, A Three-Dimensional Inverse Heat Conduction Problem in Estimating Surface Heat Flux by Conjugate Gradient Method, *Int. J. Heat Mass Transfer*, vol. 42, pp. 3387–3403, 1999.
9. C. H. Huang, I. C. Yuan, and A. Herchang, A Three-Dimensional Inverse Problem in Imaging the Local Heat Transfer Coefficients for Plate Finned-Tube Heat Exchangers, *Int. J Heat Mass Transfer*, vol. 46, pp. 3629–3638, 2003.
10. E. Videcoq and D. Petit, Model Reduction for the Resolution of Multidimensional Inverse Heat Conduction Problems, *Int. J. Heat Mass Transfer*, vol. 44, pp. 1899–1911, 2001.

11. M. Raudensky, K. A. Woodbury, J. Kral, and T. Brezina, Genetic Algorithm in Solution of Inverse Heat Conduction Problems, *Numer. Heat Transfer B*, vol. 28, pp. 293–306, 1995.
12. M. Silieti, E. Divo, and A. J. Kassab, An Inverse Boundary Element Method/Genetic Algorithm Based Approach for Retrieval of Multi-Dimensional Heat Transfer Coefficients within Film Cooling Holes/Slots, *Inverse Prob. in Sci. Eng.*, vol. 13, pp. 79–98, 2005.
13. J. Krejsa, K. A. Woodbury, J. D. Ratliff, and M. Raudensky, Assessment of Strategies and Potential for Neural Networks in the IHCP, *Inverse Prob. Eng.*, vol. 7, pp. 197–213, 1999.
14. W. Aquino and J. C. Brigham, Self-Learning Finite Elements for Inverse Estimation of Thermal Constitutive Models, *Int. J. Heat Mass Transfer*, vol. 49, pp. 2466–2478, 2006.
15. S. Deng and Y. Hwang, Applying Neural Networks to the Solution of Forward and Inverse Heat Conduction Problems, *Int. J. Heat Mass Transfer*, vol. 49, pp. 4732–4750, 2006.
16. R. Hassan, B. Cohanim, O. de Weck, and G. Venter, A Comparison of Particle Swarm Optimization and the Genetic Algorithm, *Proc. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2005.
17. C. R. Mouser and S. A. Dunn, Comparing Genetic Algorithms and Particle Swarm Optimisation for an Inverse Problem Exercise, *Austr. & N. Z. Ind. Appl. Math. J.*, vol. 46, pp. 89–101, 2005.
18. J. E. Park, K. W. Childs, G. M. Ludtka, and W. Chu, Correction of Errors in Intrinsic Thermocouple Signals Recorded during Quenching, *National Heat Treat Conf.*, pp. 26–31, Minneapolis, MN, July, 1991.
19. F. Xu and M. S. Gadala, Investigation of Error Sources in Temperature Measurement using Thermocouples in Water Impingement Cooling, *Exp. Heat Transfer*, vol. 18, pp. 153–177, 2005.
20. F. Xu and M. S. Gadala, On the Application of Intrinsic Thermocouples in Temperature Measurements in Water Jet Cooling, *Numer. Heat Transfer A*, vol. 51, pp. 343–362, 2007.
21. K. Parsopoulos and M. Vrahatis, Particle Swarm Optimizer in Noisy and Continuously Changing Environments, in M. H. Hamza (ed.), *Artificial Intelligence and Soft Computing*, pp. 289–294, IASTED/ACTA Press, Anaheim, CA, 2001.
22. J. Pugh, A. Martinoli, and Y. Zhang, Particle Swarm Optimization for Unsupervised Robotic Learning, *Swarm Intelligence Symposium*, pp. 92–99, 2005.
23. R. Eberhart and J. Kennedy, A New Optimizer using Particle Swarm Theory, *Proc. Sixth Int. Symp. on Micro Machine and Human Science*, pp. 39–43, 1995.
24. M. Clerc, *Particle Swarm Optimization*, ISTE, London, 2006.
25. M. R. Alrasheed, C. W. de Silva, and M. S. Gadala, Evolutionary Optimization in the Design of a Heat Sink, in C. W. de Silva (ed.), *Mechatronic Systems: Devices, Design, Control, Operation and Monitoring*, Taylor & Francis, Boca Raton, FL, 2008.
26. F. O. Karray and C. W. de Silva, *Soft Computing and Intelligent System Design–Theory, Tools, and Applications*, Addison Wesley, New York, 2004.
27. S. K. S. Fan and J. M. Chang, A Modified Particle Swarm Optimizer using an Adaptive Dynamic Weight Scheme, in V. G. Duffy (ed.), *Digital Human Modeling*, pp. 56–65, Springer-verlay, Berlin, 2007.
28. J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, London, 2001.
29. M. R. Alrasheed, C. W. de Silva, and M. S. Gadala, A Modified Particle Swarm Optimization Scheme and Its Application in Electronic Heat Sink Design, *ASME/Pacific Rim Technical Conf. and Exhibition on Packaging and Integration of Electronic and Photonic Systems, MEMS, and NEMS*, 2007.

30. O. Urfalioglu, Robust Estimation of Camera Rotation, Translation and Focal Length at High Outlier Rates, *The First Canadian Conf. Computer and Robot Vision*, pp. 464–471, 2004.
31. K. H. Lee, S. W. Baek, and K. W. Kim, Inverse Radiation Analysis using Repulsive Particle Swarm Optimization Algorithm, *Int. J. Heat Mass Transfer*, vol. 51, pp. 2772–2783, 2008.
32. L. Gosselin, M. Tye-Gingras, and F. Mathieu-Potvin, Review of Utilization of Genetic Algorithms in Heat Transfer Problems, *Int. J. Heat Mass Transfer*, vol. 52, pp. 2169–2188, 2009.
33. L. Davis, *Handbook of Genetic Algorithms*, Thomson, New York, 1991.
34. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman, Boston, 1989.
35. K. H. Huebner, D. L. Dewhirst, D. E. Smith, and T. G. Byrom, *The Finite Element Method for Engineers*, 4th ed., Wiley-Interscience, New York, 2001.
36. F. Xu, Finite Element Simulation of Water Cooling Process of Steel Strips on Runout Table, Ph.D. thesis, University of British Columbia, Vancouver, BC, Canada, 2005.
37. F. Xu and M. S. Gadala, Heat Transfer Behavior in the Impingement Zone under Circular Water Jet, *Int. J. Heat Mass Transfer*, vol. 49, pp. 3785–3799, 2006.
38. M. S. Gadala and F. Xu, An FE-Based Sequential Inverse Algorithm for Heat Flux Calculation during Impingement Water Cooling, *Int. J. Numer. Meth. Heat Fluid Flow*, vol. 16, pp. 356–385, 2006.
39. G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*, 2d ed., Wiley, New York, 2005.