SOLVING l_1 NORM ERRORS–IN–VARIABLES PROBLEMS USING PARTICLE SWARM OPTIMIZATION

K.E. PARSOPOULOS
Department of Mathematics,
UPAIRC, University of Patras,
GR–261.10 Patras, Greece.
email: kostasp@math.upatras.gr

E.C. LASKARI
Department of Mathematics,
UPAIRC, University of Patras,
GR–261.10 Patras, Greece.
email: elena@master.math.upatras.gr

M.N. VRAHATIS
Department of Mathematics,
UPAIRC, University of Patras,
GR–261.10 Patras, Greece.
email: vrahatis@math.upatras.gr

ABSTRACT
In this paper we deal with the solution of l_1 norm data fitting problems, which have errors in all variables. These problems can be solved using the well–known Trust Region methods [14, 16]. Alternatively, we tackle these problems by applying the Particle Swarm Optimization (PSO) technique [3, 6, 7]. The ability to work within high dimensional search spaces as well as on non–differentiable objective functions, makes PSO a good choice for such problems and results in good solutions, better than the solutions obtained from the traditional Trust Region methods.

KEY WORDS
l_1 Norm Errors-in-Variables Problems, Data Fitting Models, Particle Swarm, Optimization, Genetic Algorithms

1. INTRODUCTION

Data fitting is a central problem in Approximation Theory and can be met in many engineering applications. A fundamental issue in such problems is the choice of the measure of quality of the fit which should be used. Of course, this choice depends on the underlying problem but, in general, the l_p norms are used. Given an m–dimensional vector, $x \in \mathbb{R}^m$, the l_p norm of x is defined as

$$\begin{aligned} \|x\|_p &= \Big(\sum_{i=1}^{m} |x_i|^p\Big)^{1/p}, 1 \leqslant p < \infty, \\ \|x\|_\infty &= \max_{1 \leqslant i \leqslant m} |x_i|. \end{aligned}$$

The two most commonly used norms are the l_1 and l_2 norms, which are defined as

$$\begin{aligned} \|x\|_1 &= \sum_{i=1}^{m} |x_i|, \\ \|x\|_2 &= \Big(\sum_{i=1}^{m} |x_i|^2\Big)^{1/2}. \end{aligned}$$

The l_1 norm plays an important role in data fitting, especially when there are large errors or “wild” points in the data. This is due to the assignment of smaller weights to these points than the weights assigned by other more popular measures, such as the l_2 norm (least squares) [15]. The case considered here is the one where all variables have errors and the explicit or implicit models, which are used for fitting the data, are in general nonlinear. In the case of implicit models, constraints appear and a common technique to solve them involves solving a sequence of linear subproblems, whose solutions are constrained to lie in a trust region [14, 15, 16]. Thus, Trust Region methods, such as the one presented in [14, 16], are used to solve the problem, exploiting the structure of the subproblems, which can be solved as equality bounded variable linear programming problems.

Recently, a new evolutionary optimization technique, called Particle Swarm, has been developed by Eberhart and Kennedy [3, 6]. This technique is inspired by the population dynamics of “birds flocks” and “fish schools” and it is able to work within disjoint search spaces and on non–differentiable objective functions, since it uses only function values and no derivatives, and results in very good solutions and fast convergence rates, without any loss of efficiency. Thus, it seems a good choice for solving l_1 norm optimization problems, where the objective function is not differentiable.

In the rest of the paper, the Errors–in–Variables problem is briefly presented in Section 2 and the Particle Swarm Optimizer is presented in Section 3. Experimental results for implicit models are presented in Section 4 and conclusions are derived in Section 5.

2. THE ERRORS – IN – VARIABLES PROBLEM

Suppose that we have a set of observations $y_i \in \mathbb{R}$, $i = 1, \ldots, n$, at points $x_i \in \mathbb{R}^t$, $i = 1, \ldots, n$, and a model for the underlying relationship

$$y = F(x, \alpha), \tag{1}$$

where $\alpha \in \mathbb{R}^p$ is a vector of free parameters. We are interested in finding the values of the free parameters that give the best fit. If it is assumed that both y_i and x_i contain significant errors, r_i and ε_i respectively, then the model can be written as

$$y_i + r_i = F(x_i + \varepsilon_i, \alpha), \qquad i = 1, 2, \ldots, n. \tag{2}$$

Thus, a proper vector α has to be chosen, in order to minimize the errors r_i and ε_i for $i = 1, 2, \ldots, n$.

Although the least squares norm (l_2) is widely used in data fitting modelling and it is well studied, there are some drawbacks, getting clear whenever "wild" points or large errors are introduced into the data set. This is due to the assignment of excessive weights to these points, which results in non-satisfactory estimators. The alternative, which is usually used in such cases, is the l_1 norm and to this end several algorithms have been developed [14, 16].

Although Eq. 1 is in explicit form, it can always be written in the more general implicit form

$$f(x, \alpha) = 0, \tag{3}$$

where $x \in \mathbb{R}^k$, $k = t + 1$, represents all the variables. The corresponding model for Eq. 3 is

$$f(x_i + \varepsilon_i, \alpha) = 0, \quad i = 1, 2, \ldots, n, \tag{4}$$

where ε_i represents now the vector of the errors for all the variables. Thus, the considered problem is

$$\textbf{minimize}_{\alpha,\varepsilon}\ \|\varepsilon\|_1 \tag{5}$$

$$\textbf{subject to}\ f(x_i + \varepsilon_i, \alpha) = 0,\ i = 1, \ldots, n, \tag{6}$$

which is a constrained problem, in contradiction to the case of the explicit model of Eq. 2, where the corresponding problem is unconstrained

$$\textbf{minimize}_{\alpha,\varepsilon}\ \{\|r\|_1 + \|\varepsilon\|_1\},$$

with $r_i = F(x_i + \varepsilon_i, \alpha) - y_i,\ i = 1, \ldots, n$.

There are several ways to solve the constrained minimization problem. The Kuhn–Tucker theory provides necessary conditions for the solution of such problems, but a subgradient has to be used [5]. Alternatively, this can be solved by using an l_1 penalty function [4, 14]. In this case, the problem under consideration is

$$\textbf{minimize}_{\alpha,\varepsilon}\ \{\|f\|_1 + \nu\|\varepsilon\|_1\}, \tag{7}$$

where $\nu > 0$.

In [14, 16], a Trust Region type algorithm which involves Simplex steps for solving the linear subproblems has been considered and it has been applied to various test problems. Although the solutions resulted from that algorithm are good, there is another approach to attack the problem. This approach is more straightforward, easy to implement as well as economic in function evaluations. It uses the PSO technique to solve the problem of Eq. 7. The main aspects of PSO are described in the next section.

3. THE PARTICLE SWARM OPTIMIZATION TECHNIQUE

The PSO technique is an Evolutionary Computation technique, but it differs from other well–known Evolutionary Computation algorithms, such as the Genetic Algorithms [3, 6, 7, 8]. Although a population is used for searching the search space, there are no operators inspired by the human DNA procedures applied on the population. Instead, in PSO, the population dynamics simulates a "bird flock's" behavior, where social sharing of information takes place and individuals can profit from the discoveries and previous experience of all the other companions during the search for food. Thus, each companion, called *particle*, in the population, which is called *swarm*, is assumed to "fly" over the search space in order to find promising regions of the landscape. For example, in the minimization case, such regions possess lower function values than other, visited previously. In this context, each particle is treated as a point in a D–dimensional space, which adjusts its own "flying" according to its flying experience as well as the flying experience of other particles (companions).

There have been many variants of the PSO technique proposed so far, after Eberhart and Kennedy introduced it [3, 6, 7]. In our experiments we used a new version of this algorithm, which is derived by adding a new inertia weight to the original PSO dynamics [2, 7]. This version is described in the following paragraphs.

First, let us define the notation adopted in this paper: the i-th particle of the swarm is represented by the D–dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the best particle in the swarm, i.e. the particle with the smallest function value, is denoted by index g. The best previous position (i.e. the position giving the best function value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the position change (velocity) of the i-th particle is $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

The particles are manipulated according to the following equations

$$\begin{aligned} v_{id} &= \chi\,\big(w v_{id} + c_1 r_1 (p_{id} - x_{id}) + \\ &\qquad + c_2 r_2 (p_{gd} - x_{id})\big), &(8)\\ x_{id} &= x_{id} + v_{id}, &(9) \end{aligned}$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$ and N is the size of the population; χ is a constriction factor (usually equal to 1) which is used to control and constrict velocities; w is the inertia weight; c_1 and c_2 are two positive constants; r_1 and r_2 are two random numbers within the range $[0, 1]$.

The first equation is used to calculate the i-th particle's new velocity by taking into consideration three terms: the particle's previous velocity, the distance between the particle's best previous and current position, and, finally, the distance between swarm's best experience (the position of the best particle in the swarm) and the i-th particle's current position. Then, following the second equation, the i-th particle flies toward a new position. In general, the performance of each particle is measured according to a predefined fitness function, which is problem–dependent.

The role of the inertia weight w is considered very important in the PSO convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. In this way, the parameter w regulates the trade–off between the global (wide–ranging) and the local (nearby) exploration abilities

x_1	0.1	0.9	1.8	2.6	3.3	4.4	5.2	6.1	6.5	7.4
x_2	5.9	5.4	4.4	4.6	3.5	3.7	2.8	2.8	2.4	1.5

Table 1. Data Set 1, used in Experiments 1 and 3.

x_1	0.05	0.11	0.15	0.31	0.46	0.52	0.70	0.74	0.82	0.98	1.17
x_2	0.956	0.89	0.832	0.717	0.571	0.539	0.378	0.370	0.306	0.242	0.104

Table 2. Data Set 2, used in Experiments 2, 4 and 5.

of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine–tuning the current search area. A suitable value for the inertia weight w usually provides balance between global and local exploration abilities and consequently a reduction on the number of iterations required to locate the optimum solution. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions. Thus, a time decreasing inertia weight value is used. The initial population can be generated either randomly or by using a Sobol sequence generator [13], which ensures that the D-dimensional vectors will be uniformly distributed within the search space.

From the above discussion, it is obvious that PSO, to some extent, resembles evolutionary programming. However, in PSO, instead of using genetic operators, each individual (particle) updates its own position based on its own search experience and other individuals (companions) experience and discoveries. Adding the velocity term to the current position, in order to generate the next position, resembles the mutation operation of Genetic Algorithms. Note that in PSO, however, the "mutation" operator is guided by the particle's own "flying" experience and benefits from the swarm's "flying" experience. In another words, PSO is considered as performing mutation with a "conscience", as pointed out by Eberhart and Shi [2].

The PSO technique has been proved very efficient in solving general Global Optimization problems and performing Neural Networks training [9, 10, 11, 12]. In the next section, results obtained by the application of this technique to data fitting modelling problems are exhibited and conclusions are derived in the final section of the paper.

4. EXPERIMENTAL RESULTS

The models that we consider in this section are (or assumed to be) implicit and are the same that appear in [14, 16] in order to obtain comparable results. All the models are simple and some of them are actually explicit, but treated as implicit. The initial approximation for the vector ε is usually set equal to zero and this is the approach we followed too. Concerning parameter ν, the fixed value 0.1 was used, although it is more proper for many problems to gradually decrease its value to force convergence of the penalty function to zero. Parameter α had a different initial value for each model and the desired accuracy for all unknowns was 10^{-3}.

The values of the PSO parameters were $c_1 = c_2 = 2$ (default values) and w was gradually decreased from 1.2 toward 0.1. The maximum allowable number of PSO iterations was 500 and the size of the swarm was fixed and equal to 150 for all experiments.

In all the tables, the found optimal values for α and for the l_1 norm of ε, denoted as α^* and $\|\varepsilon^*\|_1$ respectively, are presented.

Experiment 1 [14]. The first model considered is the *hyperbolic* model introduced by Britt and Luecke in [1] with

$$f = x_1^2 x_2^2 - \alpha.$$

The data used is given in Table 1 and the initial value of α was taken equal to 100. The results for both the Trust Region and the PSO methods are given in Table 3. PSO found the solution after 130 iterations.

	Trust Region	PSO
α^*	133.402	139.688
$\|\varepsilon^*\|_1$	7.255	3.446

Table 3. Results of Experiment 1.

Experiment 2 [14]. The second model considered has

$$f = \sum_{i=0}^{2} \alpha_i x_1^i - x_2,$$

and the data used is given in Table 2. Initially, $\alpha = (1, 1, 1)^\top$ was taken and PSO found the solution after 158 iterations. The results are exhibited in Table 4.

Experiment 3 [14]. This is another model with

$$f = \sum_{i=0}^{3} \alpha_i x_1^i - x_2,$$

	Trust Region	PSO
α^*	(1.001,-1.038,0.232)	(0.997,-1.001,0.201)
$\|\|\varepsilon^*\|\|_1$	0.112	0.001

Table 4. Results of Experiment 2.

and the data used is given in Table 1, after setting the first value of x_1 equal to 0.0. Initially, $\alpha = 0$ was taken, and PSO found the solution after 160 iterations. The results are exhibited in Table 5.

	Trust Region	PSO
α^*	(5.9,-0.839,0.142,-0.015)	(5.91,-0.624,0.057,-0.006)
$\|\|\varepsilon^*\|\|_1$	1.925	0.298

Table 5. Results of Experiment 3.

Experiment 4 [14]. Here we had the nonlinear model

$$f = \alpha_1 + \frac{\alpha_2}{x_1 + \alpha_3} - x_2,$$

and the data used is given in Table 2. The initial approximation was $\alpha = (-4, 5, -4)^\top$ and PSO found the solution after 282 iterations. The results are exhibited in Table 6.

	Trust Region	PSO
α^*	(-1.934,7.761,-2.638)	(0.503,-17.009,-482.805)
$\|\|\varepsilon^*\|\|_1$	0.109	0.165

Table 6. Results of Experiment 4.

Experiment 5 [14, 16]. This is a growth model with

$$f = \alpha_1 + \alpha_2 e^{-x_1} - x_2,$$

and the data used is given in Table 2. The initial approximation was $\alpha = (0, 0)^\top$ and PSO found the solution after 490 iterations. The results are exhibited in Table 7.

	Trust Region	PSO
α^*	(-0.225, 1.245)	(-0.227,1.247)
$\|\|\varepsilon^*\|\|_1$	0.169	0.005

Table 7. Results of Experiment 5.

Experiment 6 [16]. This is the *ordinary rational function* model with

$$f = \frac{\alpha_1}{x_1 - \alpha_2} - x_2,$$

and the data was generated according to the scheme

$$\begin{aligned} x_i &= 0.01 + 0.05\,(i-1), \\ y_i + r_i &= 1 + x_i + (x_i + \varepsilon_i)^2, i = 1, \ldots, 40, \end{aligned}$$

where r_i are uniformly distributed in (-0.15, 0.15), and ε_i are uniformly distributed in (-0.05, 0.05). The initial approximation was $\alpha = (1, 1)^\top$ and PSO found the solution after 211 iterations. The results are exhibited in Table 8.

	Trust Region	PSO
α^*	(0.978, 1.006)	(1.0017,1.0002)
$\|\|\varepsilon^*\|\|_1$	1.458	1.011

Table 8. Results of Experiment 6.

Experiment 7 [16]. This is the *simple decay curve*, which describes the decay of a radioactive substance,

$$f = \alpha_1 - \alpha_2\, e^{-\alpha_3 x_1} - x_2.$$

The data was generated according to the scheme

$$\begin{aligned} x_i &= 0.02\,(i-1), \\ y_i + r_i &= 3 - e^{-(x_i + \varepsilon_i)}, i = 1, \ldots, 40, \end{aligned}$$

where r_i are uniformly distributed in (-0.15, 0.15), and ε_i are uniformly distributed in (-0.05, 0.05). The initial approximation was $\alpha = (3, 1, 1)^\top$ and PSO found the solution after 315 iterations. The results are exhibited in Table 9.

	Trust Region	PSO
α^*	(2.528, 0.918, 4.396)	(2.709, 1.007, 0.975)
$\|\|\varepsilon^*\|\|_1$	2.564	2.231

Table 9. Results of Experiment 7.

Experiment 8 [16]. This is the *logistic curve*, which describes the decay of a radioactive substance,

$$f = \alpha_1 - \log\left(1 + e^{-(\alpha_2 + \alpha_3 x_1)}\right) - x_2.$$

The data was generated according to the scheme

$$\begin{aligned} x_i &= 0.02\,(i-1), \\ y_i + r_i &= 3 - \log\left(1 + e^{-(1 + x_i + \varepsilon_i)}\right), i = 1, \ldots, 40, \end{aligned}$$

where r_i are uniformly distributed in (-0.15, 0.15), and ε_i are uniformly distributed in (-0.05, 0.05). The initial approximation was again $\alpha = (3, 1, 1)^\top$ and PSO found the

	Trust Region	PSO
α^*	(2.812, 0.487, 16.033)	(2.722, 0.851, 1.164)
$\|\varepsilon^*\|_1$	2.348	0.471

Table 10. Results of Experiment 8.

solution after **336** iterations. The results are exhibited in Table 10.

As exhibited in all tables, PSO almost always outperformed the Trust Region algorithm. It was able to detect better solutions even in vicinities of the search space that were far away from the vicinity of solutions obtained by the Trust Region method. Furthermore, although the same initial vector for the parameter α, as in [14], was used, in further experiments, that we performed, we obtained the same good solutions starting from arbitrarily chosen initial points. The number of iterations performed by the PSO method to detect the optimum solutions was surprisingly small.

5. CONCLUSIONS

In this paper, we consider the ability of the Particle Swarm Optimizer to tackle data fitting models. We have performed several experiments in well known implicit as well as explicit models. The results obtained by our approach have been compared with the corresponding results presented in [14, 16], where a Trust Region method was used.

The results presented here are very promising and make clear that PSO can solve efficiently such problems. In many cases, the problem becomes easier due to the ability of the method to detect good solutions starting from several initial points, in contradiction to the Trust Region method whose behavior is heavily influenced by the starting point. Furthermore, the solutions are obtained after a quite small number of iterations. An important role is played by the penalty parameter ν. Usually, the convergence rates of both techniques and the quality of results depend on the value of ν.

Further work has to be done to fully investigate the performance of the algorithm in more complicated problems as well as to derive possible combinations of the PSO with other algorithms on this topic.

References

[1] H.I. Britt, R.H. Luecke, The estimation of parameters in nonlinear, implicit models, *Technometrics*, 15, 1973, 233–247.

[2] R.C. Eberhart, Y.H. Shi, Evolving Artificial Neural Networks, *Proceedings of the International Conference on Neural Networks and the Brain*, Beijing, P.R. China, 1998, PL5–PL13.

[3] R.C. Eberhart, P.K. Simpson, R.W. Dobbins, *Computational Intelligence PC Tools* (Boston: Academic Press Professional, 1996).

[4] R. Fletcher, *Practical methods of Optimization* (Chichester: Wiley, 1987).

[5] J.B. Hirriart–Urruty, On optimality conditions in non–differentiable programming, *Mathematical Programming*, 14, 1978, 73–86.

[6] J. Kennedy, R.C. Eberhart, Particle Swarm Optimization, *Proceedings of the IEEE International Conference on Neural Networks*, Piscataway, NJ, USA, 1995, 1942–1948.

[7] J. Kennedy, R.C. Eberhart, *Swarm Intelligence* (Morgan Kauffman, 2001).

[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (New York: Springer, 1996).

[9] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Objective function “stretching” to alleviate convergence to local minima, *Nonlinear Analysis, Theory, Methods and Applications*, in press.

[10] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, Stretching technique for obtaining global minimizers through Particle Swarm Optimization, *Proceedings of the Particle Swarm Optimization Workshop*, Indianapolis (IN), USA, 2001, 22–29.

[11] K.E. Parsopoulos, M.N. Vrahatis, Modification of the Particle Swarm Optimizer for locating all the global minima, in V. Kurkova, N. Steele, R. Neruda, M. Karny (Eds.), *Artificial Neural Networks and Genetic Algorithms*, (New York: Springer–Verlag, Computer Science series, 2001), 324–327.

[12] K.E. Parsopoulos, M.N. Vrahatis, Particle Swarm Optimizer in noisy and continuously changing environments, in M.H. Hamza (Ed.), *Artificial Intelligence and Soft Computing*, (IASTED/ACTA Press, 2001), 289–294.

[13] W.H. Press, W.T. Vetterling, S.A. Teukolsky, B.P. Flannery, *Numerical Recipes in Fortran 77* (Cambridge: Cambridge University Press, 1992).

[14] G.A. Watson, The use of the l_1 norm in Nonlinear Errors–in–Variables problems, in Van Huffel (Ed.), *Proceedings of Recent Advances on Total Least Squares Techniques & Errors–in–Variables Modeling*, (SIAM, 1997).

[15] G.A. Watson, Choice of norms for data fitting and function approximation, *Acta Numerica*, 1998, 337–377.

[16] G.A. Watson, K.F.C. Yiu, On the solution of the errors in variables problem using the l_1 norm, *BIT*, 31, 1991, 697–710.