

Article

## Interactive statistical computer program for multiple non-linear curves fitting using stochastic algorithms

Muhammad Tlas<sup>1</sup>, Bashar Abdul Ghani<sup>1</sup>, Jamal Asfahani<sup>2</sup>

<sup>1</sup>Scientific Services Department, Atomic Energy Commission of Syria, P. O. Box 6091, Syria

<sup>2</sup>Geology Department, Atomic Energy Commission of Syria, P. O. Box 6091, Syria

E-mail: pscientific31@aec.org.sy

Received 28 April 2021; Accepted 10 June 2021; Published 1 September 2021



### Abstract

An interactive computer program for multiple nonlinear curves fitting has been developed in this work. Several optimization algorithms have been implemented in this software for solving constrained and unconstrained nonlinear optimization models in order to evaluate and best-estimate the concerning and desired suggested mathematical model parameters. Two categories of algorithms have been used in this work. The first category is random and stochastic mathematical methods (nondeterministic methods) such as the simulated annealing and the adaptive simulated annealing. The second category is the direct search methods (deterministic methods) such as Hook–Jeeves pattern search, Fletcher and Reeves conjugate gradient and steepest descent method.

**Keywords** stochastic algorithms; direct search methods; gradient methods; steepest descent algorithm; curve fitting.

Computational Ecology and Software  
ISSN 2220-721X  
URL: <http://www.iaees.org/publications/journals/ces/online-version.asp>  
RSS: <http://www.iaees.org/publications/journals/ces/rss.xml>  
E-mail: [ces@iaees.org](mailto:ces@iaees.org)  
Editor-in-Chief: WenJun Zhang  
Publisher: International Academy of Ecology and Environmental Sciences

### 1 Introduction

Mathematical modeling phenomenon is as much as a cornerstone of 20<sup>th</sup> century science, as it is a collection of empirical, experimental and field data. Mathematical modeling is essential to go beyond current knowledge, to better understand new or complex phenomena. Many instances arise, in essentially all fields of science, when mathematical models of the real world become tested by fitting some parameters to empirical data. Since the real world is often nonlinear and stochastic, it is not surprising that often this process involves fitting statistical, nonlinear, non-convex functional forms to data. Physical methods of simulated annealing have been found to be extremely useful tools for this purpose in a wide variety of examples.

This work contributes to this methodology by presenting an improvement over previous algorithms. The sections of this work give a short outline of several algorithms used such as: the simulated annealing (SA), the adaptive (very fast) simulated annealing (ASA), Hooke and Jeeves pattern search (HJPS), Fletcher and Reeves conjugate gradient (FRCG), and finally the steepest decent (SD). These algorithms found to be extremely useful for multi-dimensional parameter-spaces. A computer program has been implemented and developed to

statistically find the best global fit of a nonlinear non-convex loss-function over a multi-dimensional parameters-space.

## 2 Summary of the Adaptive Simulated Annealing Algorithm

Simulated annealing techniques are implemented for finding global minimum (or maximum) of a target function in parameter space. The techniques are adopted from the physical annealing procedure where a liquid is cooled down in order to obtain a minimum energy formation. These techniques were developed due to the fact that, stochastic and non-linear systems are extremely difficult to be minimized. Stochastic methods such as adaptive simulated annealing (very fast simulated re-annealing) have been found to be extremely useful tools for a wide variety of minimization problems of large non linear systems.

Adaptive simulated annealing is a powerful stochastic optimization method applicable to a wide range of problems, especially for multi-modal, discrete, non-linear and non-differentiable target functions. The major advantage of adaptive simulated annealing over other methods is its ability to avoid becoming trapped at local minima. The algorithm employs a random search, which does not only accept changes that decrease the objective function, but also accepts some changes that increase it, at least temporarily.

We will now illustrate the adaptive simulated annealing random search algorithm for solving the following multi-variables unconstrained problem:

$$\begin{aligned} & \text{Minimize } \phi(v) \\ & \text{Subject to } v \in \mathbf{R}^n \end{aligned}$$

where the numerical function  $\phi(v)$  is called the objective (target or loss) function of the problem and  $v = (v_1, \dots, v_n) \in \mathbf{R}^n$  is the vector of model parameters (decision variables).

Using function minimization for illustrative purposes, the algorithm proceeds as follows:

*The algorithm*

*Initialization: /\*Definition of initial temperatures, radius of sphere, initial solution, iteration control parameter\*/. Let*

User-defined control parameters:  $\alpha_0 > 0, r > 0, t^0 \in \mathbf{R}_+^n$

A user-defined initial solution:  $v^0 \in \mathbf{R}^n$

A user-defined small positive real number close to zero:  $\varepsilon$

An initial number of iteration:  $i = 0$

*Main procedure: Repeat until ( $\alpha_i < \varepsilon$ )*

*Step 1: /\*Applied a random perturbation to each decision parameter\*/*

*for  $j = 1$  to  $n$  do*

*{*

*Generate a random number  $u$  between 0 and 1:  $u = \text{random}[0,1]$  /\*by the continuous uniform distribution\*/*

$$\text{Set } \theta = \text{sgn}(u - 0.5) t_j^i \left[ \left( 1 + \frac{1}{t_j^i} \right)^{2u-1} - 1 \right] \quad \text{/*new random generator*/}$$

$$\text{Set } \hat{v}_j = v_j^i + \theta \frac{r}{\sqrt{n}} \quad \text{/* } r \text{ is the radius of sphere centered at the point } v^i \text{ */}$$

}

Step 2: /\*Acceptance or rejection of the changes made on the decision parameters\*/

if  $\phi(\hat{v}) < \phi(v^i)$  then set  $v^{i+1} = \hat{v}$  and go to step 3

else calculate the probability  $p = \frac{1}{1 + e^{-\frac{\phi(\hat{v}) - \phi(v^i)}{\alpha_i}}}$  /\*Boltzmann distribution\*/

if  $p > \gamma = \text{random}[0,1]$  then set  $v^{i+1} = \hat{v}$

else set  $v^{i+1} = v^i$

Step 3: /\*Modification of temperatures and iteration control parameter\*/

for  $j=1$  to  $n$  do

{

$$\text{Set } t_j^{i+1} = \frac{t_j^0}{i+1}$$

}

Set  $\alpha_{i+1} = \frac{\alpha_0}{i+1}$ ,  $i = i + 1$  and go to step 1.

This algorithm has a good robustness and is easy to be inserted in a code. It does not require differentiability of the objective function with respect to the decision variables. Asfahani and Tlas (2007, 2011), Ingber and Rosen (1992), Ingber (1989, 1993, 1996), Sen and Stoffa (1996), Chen et al. (1998), Corana et al. (1987), Kirkpatrick et al. (1983), and Van Laarhoven and Aarts (1987) provide more details about the simulated annealing algorithms.

With this mathematical description, a chance will be given to the user to choose and define arbitrarily the control parameters (temperature  $\alpha_0 > 0$ , radii  $r > 0$ ) and the initial guess  $v^0 \in \mathbf{R}^n$ , where the convergence speed towards the optimal solution and the number of iterations can be augmented or reduced according to these choices.

### 3 Summary of the Simulated Annealing Algorithm

Initialization: /\*Definition of initial temperatures, radius of sphere, reduction parameter, initial solution \*/.

Let

User-defined control parameters:  $\alpha_0 > 0, r > 0, 0 < \beta < 1$

A user-defined initial solution:  $v^0 \in \mathbf{R}^n$

A user-defined small positive real number close to zero:  $\varepsilon$

An initial number of iteration:  $i = 0$

*Main procedure:* Repeat until  $(\alpha_i < \varepsilon)$

*Step 1: /\*Applied a random perturbation to each decision parameter\*/*

*for*  $j = 1$  *to*  $n$  *do*

{

Generate a random number  $\theta$  between -1 and 1:  $\theta = \text{random}[-1,1]$  */\*by the continuous uniform distribution\*/*

Set  $\hat{v}_j = v_j^i + \theta \frac{r}{\sqrt{n}}$  */\* r is the radius of sphere centered at the point  $v^i$  \*/*

}

*Step 2: /\*Acceptance or rejection of the changes made on the decision parameters\*/*

*if*  $\phi(\hat{v}) < \phi(v^i)$  *then* set  $v^{i+1} = \hat{v}$  *and go to step 3*

*else* calculate the probability  $p = \frac{1}{1 + e^{-\frac{\phi(\hat{v}) - \phi(v^i)}{\alpha_i}}}$  */\*Boltzmann distribution\*/*

*if*  $p > \gamma = \text{random}[0,1]$  *then* set  $v^{i+1} = \hat{v}$

*else* set  $v^{i+1} = v^i$

*Step 3: /\*Modification of temperatures \*/*

Set  $\alpha_{i+1} = \beta \alpha_i$ ,  $i = i + 1$  *and go to step 1.*

It should be noted that, there are others mathematical formulas can be used for the reduction control factor  $\alpha$  in both the simulated annealing algorithm and the adaptive simulated annealing method. These formulas are given as:

1.  $\alpha_{i+1} = \frac{\alpha_0}{i+1}$  ( $i = 0, 1, 2, \dots$ )
2.  $\alpha_{i+1} = \frac{\alpha_0}{\ln(i+2)}$  ( $i = 0, 1, 2, \dots$ )
3.  $\alpha_{i+1} = \alpha_0 e^{(c-1)(i+1)}$  ( $i = 0, 1, 2, \dots$ ),  $0 < c < 1$
4.  $\alpha_{i+1} = (1-c)\alpha_i$  ( $i = 0, 1, 2, \dots$ ),  $0 < c < 1$
5.  $\alpha_{i+1} = \frac{\alpha_0}{(i+1)^{\frac{1}{c}}}$  ( $i = 0, 1, 2, \dots$ ),  $c > 0$
6.  $\alpha_{i+1} = (1-c)^{i+1}$  ( $i = 0, 1, 2, \dots$ ),  $0 < c < 1$
7.  $\alpha_{i+1} = \frac{\alpha_0}{(i+1)^\alpha}$  ( $i = 0, 1, 2, \dots$ ),  $\alpha > 0$
8.  $\alpha_{i+1} = \alpha_0 e^{-c(i+1)^{\frac{1}{n}}}$  ( $i = 0, 1, 2, \dots$ ),  $c > 0$

Also, the same mathematical formulas can be used for another parameter  $t$  in both previous algorithms.

#### 4 Summary of Hooke and Jeeves Direct Search Algorithm

The Hooke and Jeeves direct search algorithm (1961) is one of the most widely known methods for optimizing a numerical function of several real variables on the real space  $\mathbf{R}^n$ .

The algorithm consists of two distinct phases. The first one is an exploratory search phase, which serves to establish a direction of improvement. The second one is a pattern move, which extracts the current solution vector to another point in the solution space. The algorithm for minimizing a numerical function proceeds as follows:

*Initialisation (Input):*  $\varepsilon > 0$  is the accuracy parameter;  $n$  linearly independent search directions  $d^1, \dots, d^n$ . It is possible to take  $d^j = h e^j$  ( $j = 1, \dots, n$ ) where  $h \in \mathbf{R}$  and  $e^j$  denote the  $j$ th unit vector, equal to the  $j$ th column in the unit matrix  $I$ ;  $0 < \alpha < 1$  damping factor;  $v \in \mathbf{R}^n$  is a given initial point.

Call the subroutine auxiliary procedure  $\hat{v} = \text{explore}(v, h)$

*Main procedure*

While ( $|h| \geq \varepsilon$ ) do

{ we always know two points  $v$  and  $\hat{v}$  }

if  $\phi(\hat{v}) < \phi(v)$  then

$z = \hat{v} + (\hat{v} - v)$

$v = \hat{v}$  (Pattern move or search)

else

$h = \alpha h$

$z = v$

end

```

 $\hat{v} = \text{explore}(z, h)$ 
end
Auxiliary procedure (exploratory search)  $\hat{v} = \text{explore}(v, h)$ 
begin
 $\hat{v} = v$ 
for  $j = 1$  to  $n$  do
 $\hat{\phi} = \min\{\phi(\hat{v} - he^j), \phi(\hat{v}), \phi(\hat{v} + he^j)\}$ 
 $\hat{v} = \text{Corresponding point}$ 
end
end

```

This algorithm has a good robustness and is easy to be put in a code. It does not require differentiability of the objective function with respect to the decision variables. Hooke and Jeeves (1961), Bazaraa and Shetty (1979), Nash (1990), Phillips et al. (1976), and Himmelblau (1972) provide more details about this algorithm.

### 5 Summary of Fletcher and Reeves Conjugate Gradient Method

The conjugate gradient method of Fletcher and Reeves creates search directions that are a linear combination of the steepest decent direction and previous search directions. Weighting factors are applied such that the search directions are conjugate. These factors are ratios of the present and past squared norms of the gradient.

**Initialization:** Choose the tolerance level  $\varepsilon > 0$  and the initial point  $v^1 \in \mathbf{R}^n$ .

Let  $y^1 = v^1$ ,  $d^1 = -\nabla\phi(y^1)$ ,  $k = j = 1$

*Main procedure*

**Step 1:** If  $\|\nabla\phi(y^j)\| < \varepsilon$ , stop.

else find

$$\lambda_j = \arg \min \phi(y^j + \lambda d^j)$$

$$ST \quad \lambda \geq 0$$

Put  $y^{j+1} = y^j + \lambda_j d^j$

If  $j < n$ , go to **step 2**, else go to **step 3**

**Step 2:** Let

$d^{j+1} = -\nabla\phi(y^{j+1}) + \alpha_j d^j$  Where

$$\alpha_j = \frac{\|\nabla\phi(y^{j+1})\|^2}{\|\nabla\phi(y^j)\|^2}$$

$j = j + 1$ , go to **step 1**

**Step 3:** Let

$$y^1 = v^{k+1} = y^{n+1}, d^1 = -\nabla \phi(y^1), j=1, k=k+1,$$

Go to **step 1**.

The concept of conjugacy is very important in unconstrained optimization problems. If the function to be minimized is quadratic, conjugate search directions guarantee that convergence will occur in at most  $n$  steps, where  $n$  is the number of parameters being adapted.

The conjugate gradient method works also very efficiently for non-quadratic problems, having very modest storage requirement.

## 6 Summary of Steepest Descent Method

**Initialization:** Choose the tolerance level  $\varepsilon > 0$  and the initial point  $v^1 \in \mathbf{R}^n$ , let  $k = 1$ .

*Main procedure*

**STEP 1:** IF  $\|\nabla \phi(v^k)\| < \varepsilon$ , STOP, ELSE GO TO STEP 2

**Step 2:** put  $d^k = -\nabla \phi(v^k)$ ,

$$\lambda_k = \arg \min_{\lambda \geq 0} \phi(v^k + \lambda d^k)$$

$$v^{k+1} = v^k + \lambda_k d^k$$

$k = k + 1$  and go to **step 1**.

The steepest descent method converges if the sequence of points generated is contained in a compact subset of  $\mathbf{R}^n$ . It usually works well during early stages of the optimization process, depending on  $v^1 \in \mathbf{R}^n$ .

However, as a stationary point is approached, the method usually starts zigzagging, taking small, nearly orthogonal steps. References of Bradely et al. (1977), Hillier et al. (1986), Phillips and Ravindra (1988), Bazaraa and Shetty (1979), and Himmelblau (1972) provide more details about these algorithms.

Now, it is noticed that, if some mathematical constraints exist on the decision variables as shown in the coming mathematical constrained optimization problem:

$$\begin{aligned} & \text{Minimize } f(v) \\ & \text{Subject to } g_i(v) \geq 0 \quad (i = 1, \dots, m) \\ & v \in \mathbf{R}^n \end{aligned}$$

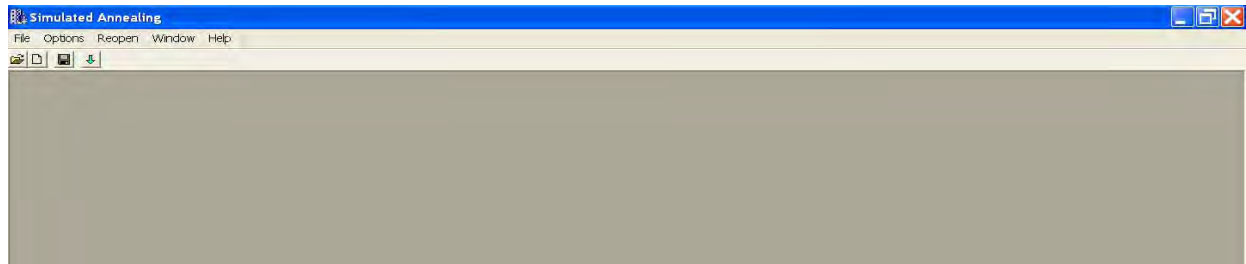
Then, in this case, the penalty function, defined below, as a new objective function of the problem, can be used:

$$\begin{aligned} & \text{Minimize } \phi(v) = f(v) - r \left( \sum_{i=1}^m \ln(g_i(v)) \right) \\ & \text{Subject to } v \in \mathbf{R}^n \end{aligned}$$

where  $r$  (penalty factor) is an arbitrary positive real number chosen to be close to zero.

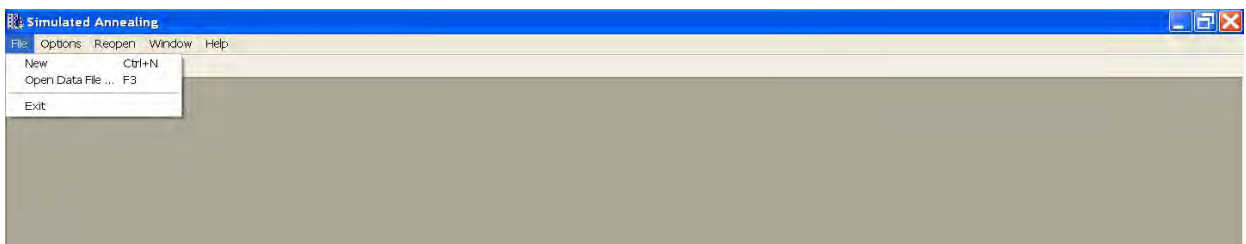
## 7 The Interactive Computer Code Algorithm

The interactive menu-driven computer code is developed using C++ programming language and Borland C++ Builder. The starting main interface screen contains two rows; the main menu is composed of five tabs entitled: (file, options, reopens window and help) and the second row is consisted of three tabs entitled (open, new file and save) as shown in Fig. 1.



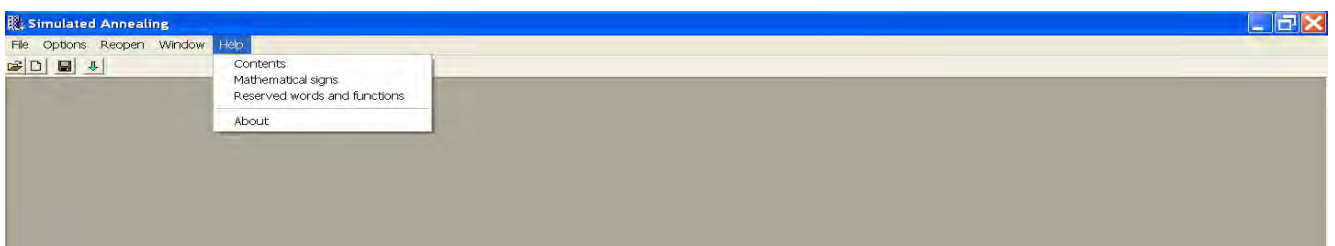
**Fig. 1** Main menu screen.

The tab "file" is a pop-up menu and consists of three commands (new, open data file, and exit) as seen in Fig. 2.



**Fig. 2** The file command.

The tab "help" consists of four commands entitled (mathematical signs, reserved words and functions, and help) as displayed in Fig. 3.



**Fig. 3** Help command.

For the rest of tabs, user is invited to deal with them by himself.

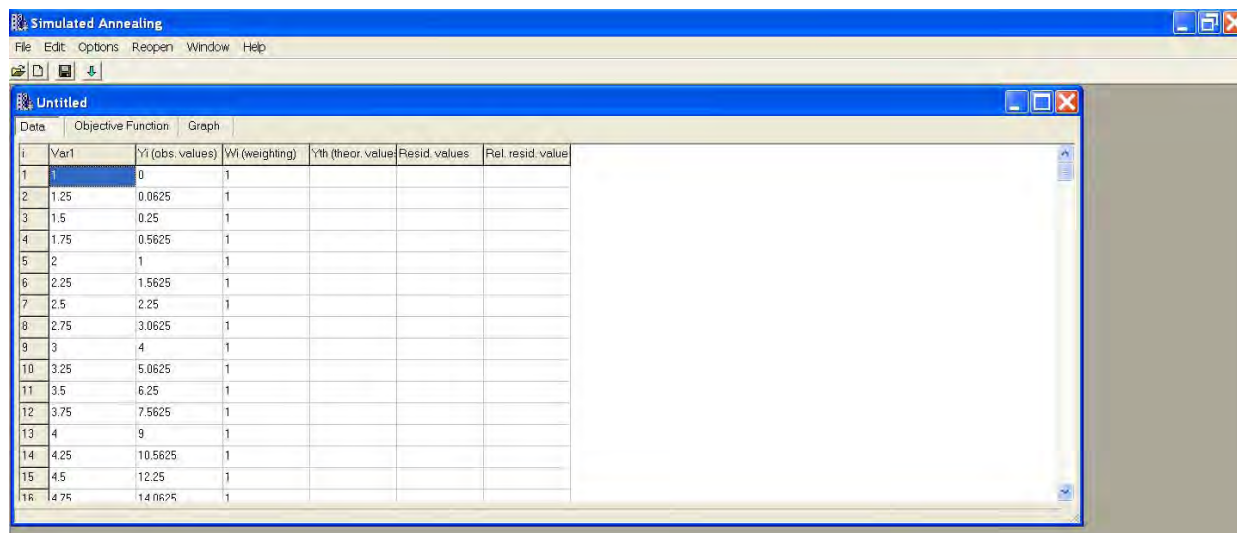


The proposed code algorithm is composed from the following three steps:

**Step 1** Copy the experimental big data from an Excel file or from another statistical program;

- Open the tab "file" from the main row and choose the command "new" to open an empty file
- Open the tab " edit" from the main row and choose the command " past data"

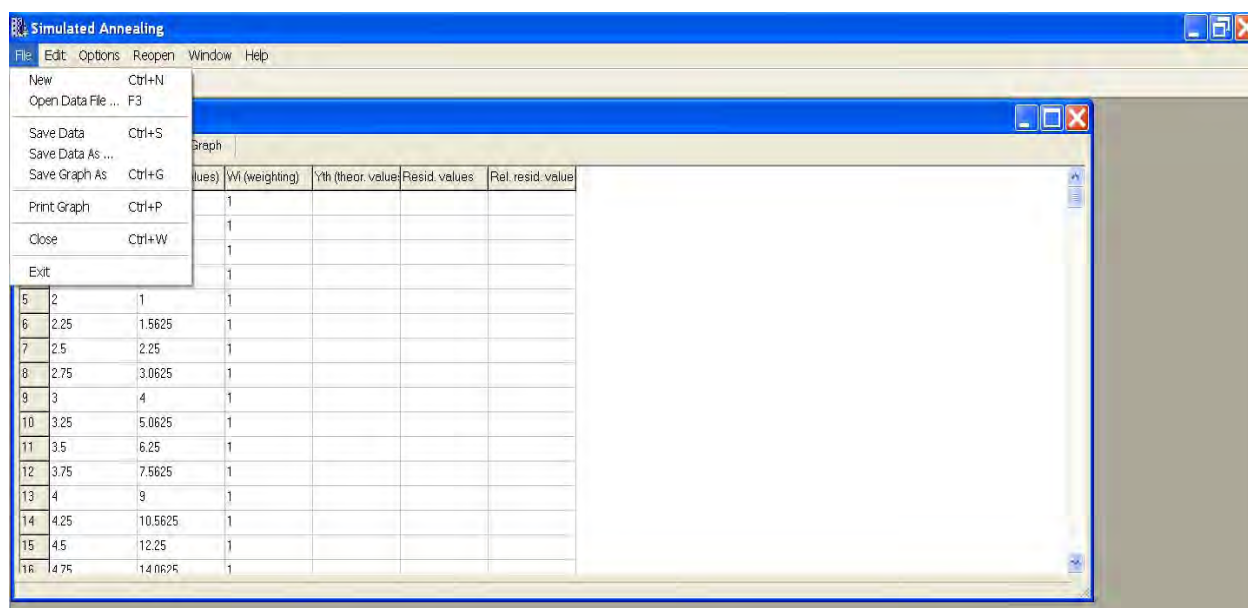
As described in Fig. 4.



| i  | Var1 | Y1 (obs. values) | W1 (weighting) | Yth (theor. value) | Resid. values | Rel. resid. value |
|----|------|------------------|----------------|--------------------|---------------|-------------------|
| 1  |      | 0                | 1              |                    |               |                   |
| 2  | 1.25 | 0.0625           | 1              |                    |               |                   |
| 3  | 1.5  | 0.25             | 1              |                    |               |                   |
| 4  | 1.75 | 0.5625           | 1              |                    |               |                   |
| 5  | 2    | 1                | 1              |                    |               |                   |
| 6  | 2.25 | 1.5625           | 1              |                    |               |                   |
| 7  | 2.5  | 2.25             | 1              |                    |               |                   |
| 8  | 2.75 | 3.0625           | 1              |                    |               |                   |
| 9  | 3    | 4                | 1              |                    |               |                   |
| 10 | 3.25 | 5.0625           | 1              |                    |               |                   |
| 11 | 3.5  | 6.25             | 1              |                    |               |                   |
| 12 | 3.75 | 7.5625           | 1              |                    |               |                   |
| 13 | 4    | 9                | 1              |                    |               |                   |
| 14 | 4.25 | 10.5625          | 1              |                    |               |                   |
| 15 | 4.5  | 12.25            | 1              |                    |               |                   |
| 16 | 4.75 | 14.0625          | 1              |                    |               |                   |

**Fig. 4** Input experimental data screen.

It is to notice that, always, there are possibilities to change data and weights manually also it can copy it to re-use aware in other programs by using from the tab "edit" the command "copy all data". Also it can save the data file with extension "owner name.sa" any ware on computer to re-use it later, as clarified in Fig 5.



| i  | Var1 | Y1 (obs. values) | W1 (weighting) | Yth (theor. value) | Resid. values | Rel. resid. value |
|----|------|------------------|----------------|--------------------|---------------|-------------------|
| 5  | 2    | 1                | 1              |                    |               |                   |
| 6  | 2.25 | 1.5625           | 1              |                    |               |                   |
| 7  | 2.5  | 2.25             | 1              |                    |               |                   |
| 8  | 2.75 | 3.0625           | 1              |                    |               |                   |
| 9  | 3    | 4                | 1              |                    |               |                   |
| 10 | 3.25 | 5.0625           | 1              |                    |               |                   |
| 11 | 3.5  | 6.25             | 1              |                    |               |                   |
| 12 | 3.75 | 7.5625           | 1              |                    |               |                   |
| 13 | 4    | 9                | 1              |                    |               |                   |
| 14 | 4.25 | 10.5625          | 1              |                    |               |                   |
| 15 | 4.5  | 12.25            | 1              |                    |               |                   |
| 16 | 4.75 | 14.0625          | 1              |                    |               |                   |

**Fig. 5** Saving data screen.

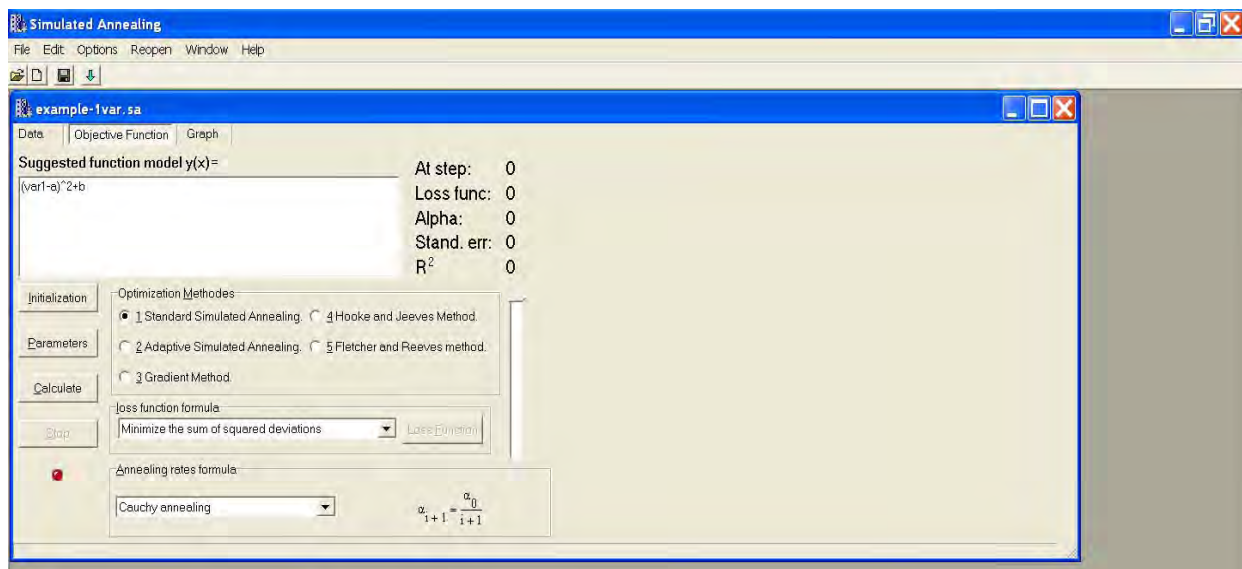
**Step 2** write the desired forward function in the field addressed "objective function" by using the following reserved signs, words and functions:

- The reserved mathematical signs are:

|              |                             |
|--------------|-----------------------------|
| () , {} , [] | Parenthesis                 |
| ^            | Power                       |
| + -          | Addition and Subtraction    |
| / *          | Division and multiplication |
| %            | Modula                      |

- The reserved words and functions in capital or small letters are:

acos, asin, atan, cos, sin, tan, cosh, sinh, tanh, ln or log, log10, fact, ceil, floor, round, sign, sqrt, abs, exp, int, pi=3.14, e=2.71, as depicted in Fig. 6.



**Fig. 6** Forward function screen.

**Step 3** Choose one of the following five algorithms from the field addressed "Optimization methods":

### 3-1 Standard simulated annealing method

- Choose the key "initialization", for setting the initial guess for all reserved parameters with respect to the following inequalities:  $\alpha > 0$  ,  $R > 0$  and  $0 < \beta < 1$  , as displayed in Fig. 7.

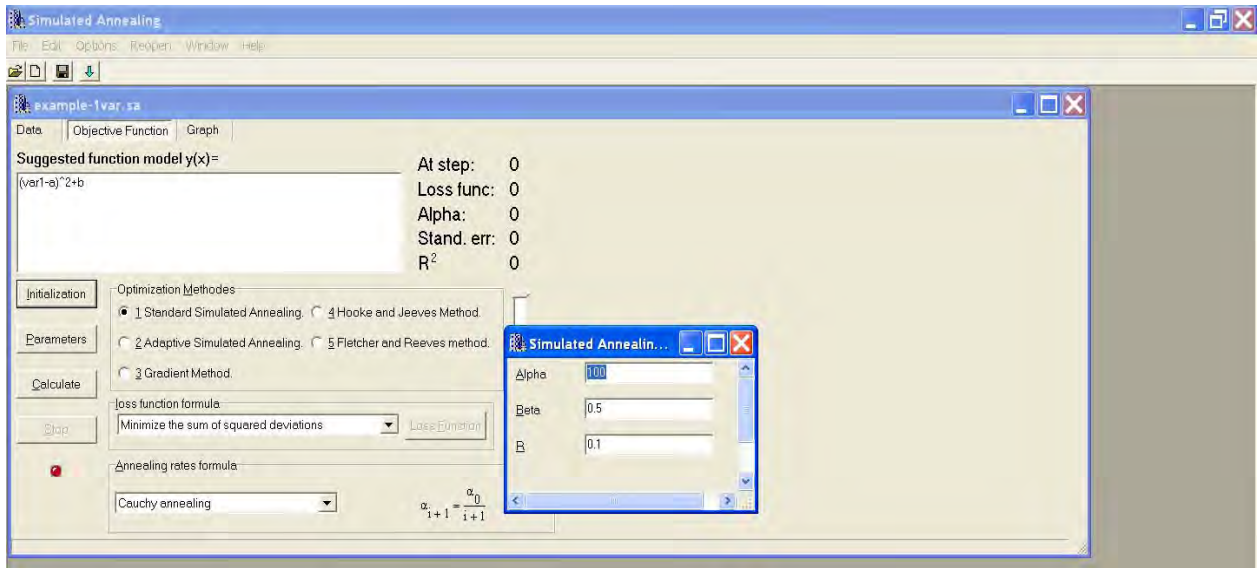


Fig. 7 Reserved parameters screen.

- b. Pressing the key "Parameters", will show the forward model parameters to be estimated. Set initial guess for all parameters arbitrary as represented in Fig. 8.

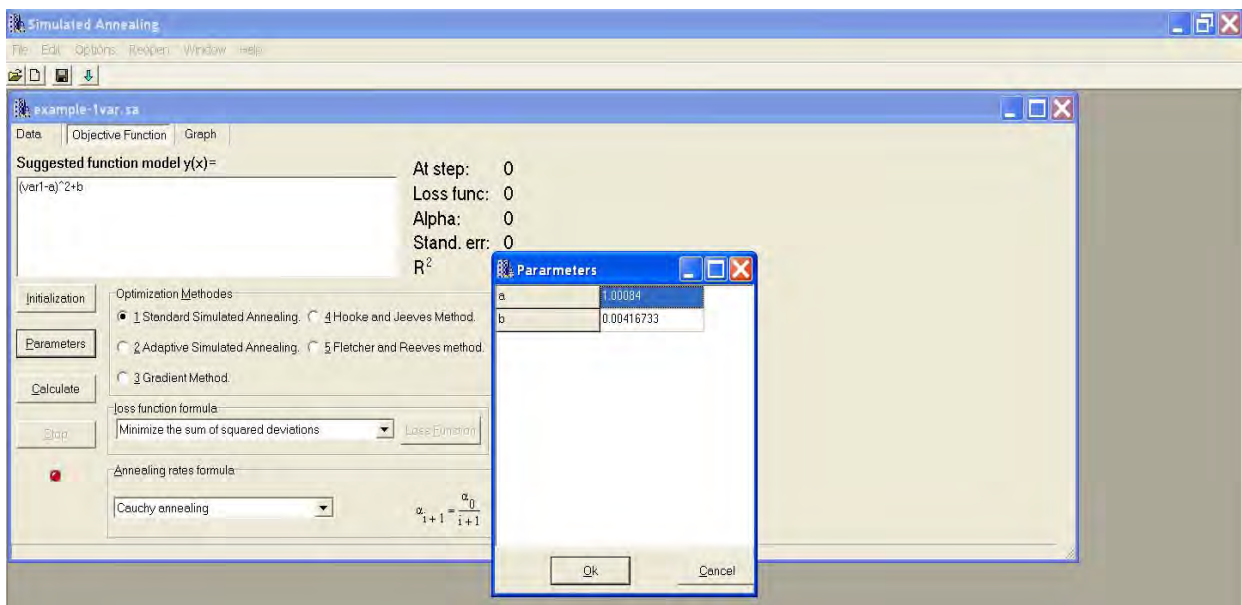


Fig. 8 Forward model parameters screen.

- c. Choose one of the following estimation methods from the field addressed "Loss function formula" as clarified in Fig. 9:
- Minimize the sum of squared deviations:

$$\text{Min} \sum_{i=1}^n (\text{Obs}_i - \text{Pred}_i)^2$$

- Minimize the sum of absolute deviation:

$$\text{Min} \sum_{i=1}^n |\text{Obs}_i - \text{Pred}_i|$$

- Minimize the sum of weighted squared deviations:

$$\text{Min} \sum_{i=1}^n w_i (\text{Obs}_i - \text{Pred}_i)^2$$

- Minimize the sum of weighted absolute deviations:

$$\text{Min} \sum_{i=1}^n w_i |\text{Obs}_i - \text{Pred}_i|$$

- Maximize the likelihood function:

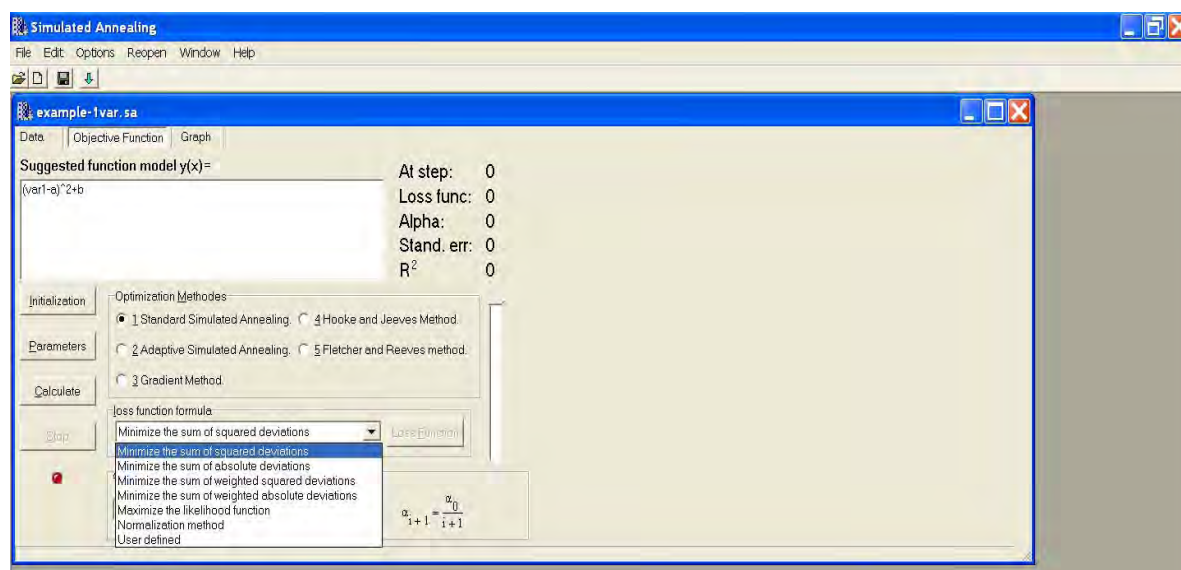
$$\text{Min} \left[ n \ln(\sqrt{2\pi} \sigma) + \frac{1}{2\sigma^2} \sum_{i=1}^n (\text{Obs}_i - \text{Pred}_i)^2 \right]$$

- Normalization method:

$$\text{Min} \left[ -\zeta + \ln \left( \frac{n}{\sqrt{2\pi} \sigma} \right) + \frac{1}{2n\sigma^2} \sum_{i=1}^n (\text{Obs}_i - \text{Pred}_i)^2 \right]^2$$

- User defined. There is a possibility to construct owner loss function by using the reserved mathematical term *obs \_ pred* defined as follows:

$$\text{obs\_pred}^q = \sum_{i=1}^n \text{weight}_i \times |\text{obs}_i - \text{pred}_i|^q \quad (q > 0)$$



**Fig. 9** Loss function formula.

d. Choose one of the following damping rate formulas from the field addressed "Annealing rate formulas", as shown in Fig.10.

- Cauchy annealing: 
$$\alpha_{i+1} = \frac{\alpha_0}{i+1}$$
- Boltzmann annealing: 
$$\alpha_{i+1} = \frac{\alpha_0}{\ln(i+2)}$$
- Exponential annealing 1: 
$$\alpha_{i+1} = \alpha_0 e^{-(\beta-1)(i+1)}$$
- Geometric annealing: 
$$\alpha_{i+1} = \beta \alpha_i$$
- Decreasing annealing 1: 
$$\alpha_{i+1} = \frac{\alpha_0}{(i+1)^{1/n}}$$
- Power annealing: 
$$\alpha_{i+1} = (1-\beta)^{i+1}$$
- Decreasing annealing 2: 
$$\alpha_{i+1} = \frac{\alpha_0}{(i+1)^\beta}$$
- Exponential annealing 2: 
$$\alpha_{i+1} = \alpha_0 e^{-\beta(i+1)^{1/n}}$$

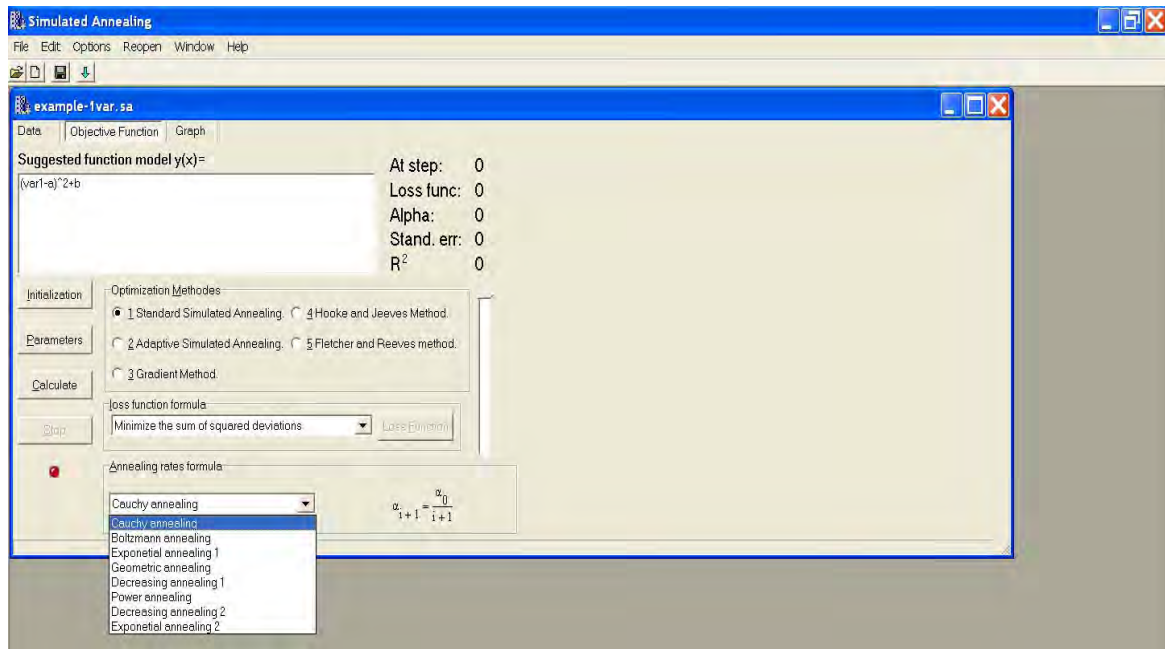


Fig. 10 Damping rate formulas.

- e. Choose "calculate", then the code starts to compute the forward model parameters
- f. Choose "stop" for stopping the execution any time or the code will stop automatically when the stopping criterions of algorithms are satisfied. The code can provide, at each step of execution the values of model parameters, loss function, R squared, and standard error.
- g. Choose "Graph" to draw the curve in two dimensions, as illustrated in Fig. 11.

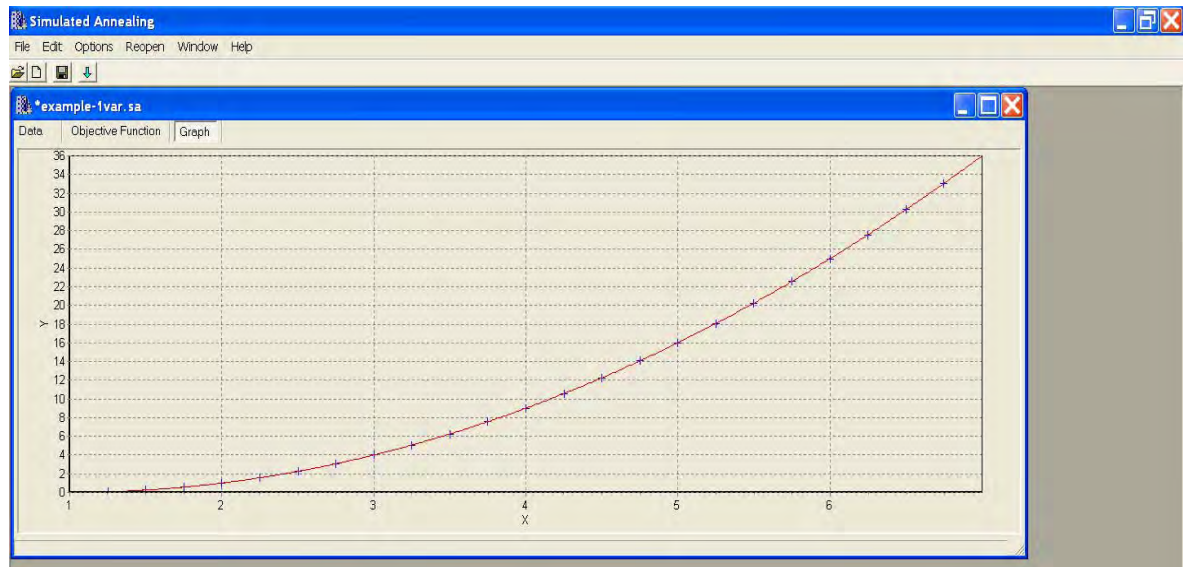
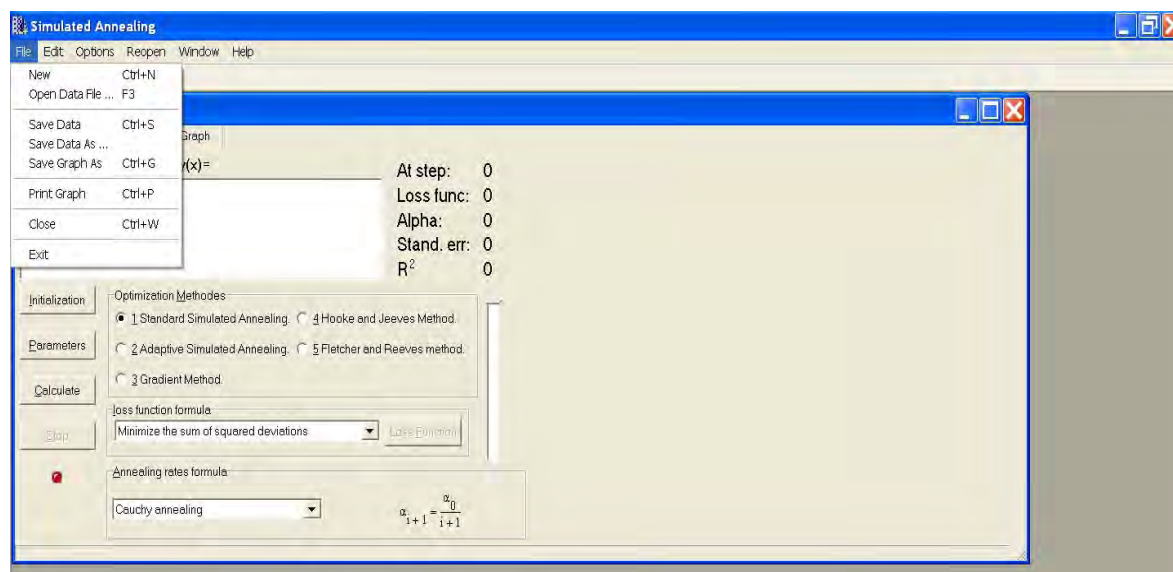


Fig. 11 Curve graph.

It should be pointed out that there are possibilities to save, copy and print all code outputs on computer any were to re-use or reopen it later with extension "owner name.sa", as represented in Fig. 12:



**Fig. 12** Save, print, and reopen code output.

Also, the code permits possibilities to treat, simultaneously, more than one experimental data file since it uses "multithreading" technique which aids users to open more than one window at a time.

### 3-2 Adaptive simulated annealing method

Repeat the same steps as explained in paragraph 3-1 with respect to the following inequalities:  $\text{Alpha} > 0$ ,  $R > 0$ ,  $0 < \text{Beta} < 1$ ,  $0 < c < 1$  and  $T_i > 0$  for all  $i$ .

### 3-3 Hooke and Jeeves method

Repeat the same steps as explained in 3-1 with respect to the following inequalities:  $\text{epsilon} < 10^{-3}$ ,  $\text{delta} > 0$  and  $0 < \text{gamma} < 1$ .

### 3-4 Steepest descent method

Repeat the same steps as explained in 3-1 with respect to the following inequalities:  $\text{epsilon} < 10^{-3}$  and  $H < 10^{-3}$ .

### 3-5 Fletcher and Reeves algorithm

Repeat the same steps as explained in 3-1 with respect to the following inequalities:  $\text{epsilon} < 10^{-3}$  and  $H < 10^{-3}$ .

## 8 Conclusions

A user-friendly interactive computer program for multiple nonlinear curves fitting has been proposed and developed in this work. The code algorithm is mainly based on famous mathematical optimization methods well-known in optimization theory such as: Standard simulated annealing, adaptive (very fast) simulated annealing, Hooke and Jeeves pattern search algorithm, Fletcher and Reeves conjugate gradient and steepest descent method.

The code is implemented and programmed using C++ programming language within Borland C++ Builder environment. "Software.rar" file contains examples of input data files, testing examples and the exe-file of this software which can be loaded directly when it is needed.

## Acknowledgment

Authors wish to thank Prof. I. Othman, the Director General of the Syrian Atomic Energy Commission for his valuable support and encouragement throughout this work.

## References

- Asfahani J, Tlas M. 2007. A robust nonlinear inversion for the interpretation of magnetic anomalies caused by faults, thin dikes and spheres like structure using stochastic algorithms. *Pure and Applied Geophysics*, 164: 2023-2042
- Asfahani J, Tlas M. 2012. Fair function minimization for direct interpretation of residual gravity anomaly profiles due to spheres and cylinders. *Pure and Applied Geophysics*, 169: 157-165
- Bazaraa MS, Shetty CM. 1979. *Nonlinear Programming: Theory and Algorithms*. John Wiley and Sons Inc, New York, NY, USA
- Bradely SP, Cax AC, Magnanti TL. 1977. *Applied Mathematical Programming*. Addison-Wesley Publishing Company, USA
- Chen S, Luk BL, Liu Y. 1998. Application of adaptive simulated annealing to blind channel identification with HOC fitting. *Electronics Letters*, 34: 234-235
- Corana A, Marchesi M, Martini C, Ridella S. 1987. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Transactions on Mathematical Software*, 13: 262-280
- Hillier FS, Lieberman GJ. 1986. *Introduction to Operation Research*. Holden-Day Inc, Australia
- Himmelblau DM. 1972. *Applied Nonlinear Programming*. McGraw-Hill Inc, New York, USA
- Hooke R, Jeeves TA. 1961. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8
- Ingber L. 1989. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, 12(8): 967-973
- Ingber L. 1993. Simulated annealing: practice versus theory. *Mathematical and Computer Modelling*, 18: 29-57
- Ingber L. 1996. Adaptive simulated annealing (ASA): lessons learned. *Journal of Control and Cybernetics*, 25: 33-54
- Ingber L, Rosen B. 1992. Genetic algorithms and very fast simulated re-annealing: A comparison. *Mathematical and Computer Modelling*, 16(11): 87-100
- Kirkpatrick S, Gelatt JCD, Vecchi MP. 1983. Optimization by simulated annealing. *Science*, 220: 671-680
- Nash JC. 1990. *Compact Numerical Method for Computers, Linear Algebra and Function Minimization*. Adam Hilger, UK
- Phillips DT, Ravindra A, Solber JJ. 1976. *Operations Research Principles and Practice*. John Wiley and Sons Inc, USA
- Sen M, Stoffa PL. 1996. Bayesian inference, Gibbs' sampler and uncertainty estimation in geophysical inversion. *Geophysical Prospecting*, 44: 313-350
- Van Laarhoven PJM, Aarts EHI. 1987. *Simulated Annealing: Theory and Applications*. D. Reidel, Dordrecht, Netherlands