Applied Mathematics and Computation
Manuscript Draft

Manuscript Number: AMC-D-15-01974

Title: A New Metropolis Optimization Method, the Cross-Section Particle Collision Algorithm.

Article Type: Full Length Article

Title: A New Metropolis Optimization Method, the Cross-Section Particle Collision Algorithm.

Authors: W.F. Sacco[a] and N. Henderson[b].

Affiliation:

[a] Instituto de Engenharia e Geociências, Universidade Federal do Oeste do Pará, Av. Vera Paz, s/n, Santarém, PA, 68135-110 Brazil.

[b] Depto. de Modelagem Computacional, Instituto Politécnico, Universidade do Estado do Rio de Janeiro, R. Bonfim, 25, Nova Friburgo, RJ, 28625-570 Brazil.

Number of Pages: 29

Number of Figures: 0

Number of Tables: 7

Please address all correspondence to:

Dr. Wagner F. Sacco

Instituto de Engenharia e Geociências

Universidade Federal do Oeste do Pará

Rua Vera Paz, s/n, Salé

Santarém, PA, 68035-110, Brazil

Tel: +55-93-98801-2391

Fax: +55-93-2101-4902

e-mails: wagner.sacco@ufopa.edu.br, wagner.sacco@pq.cnpq.br

**Highlights (for review)**

- We introduce a new Metropolis algorithm;
- This algorithm, CSPCA, is an enhancement of the Particle Collision Algorithm (PCA);
- Tests are performed using five real-world benchmarks;
- CSPCA outperforms not only PCA, but also differential evolution.

# A New Metropolis Optimization Method, the Cross-Section Particle Collision Algorithm.

Wagner F. Sacco [a,*], Nélio Henderson [b].

[a] *Instituto de Engenharia e Geociências, Universidade Federal do Oeste do Pará, Av. Vera Paz, s/n, Santarém, PA, 68135-110 Brazil.*

[b] *Instituto Politécnico, Universidade do Estado do Rio de Janeiro, R. Bonfim, 25, Nova Friburgo, RJ, 28625-570 Brazil.*

**Abstract**

In this work, we introduce a new Metropolis algorithm, which is an enhancement of the recent Particle Collision Algorithm (PCA), loosely inspired by the neutron interactions in a reactor. This novel method is called the Cross-Section Particle Collision Algorithm (CSPCA), as it incorporates the concept of cross-section from Reactor Physics, in the sense that points in the search space and their respective fitness-function values are analogous to the neutron cross sections which are used to express the likelihood of interaction between an incident neutron and a target nucleus. CSPCA is compared against the original PCA and the well-known differential evolution. These methods are applied to five real-world problems: two parameter estimation problems, a nonlinear system, the ten-atom Lennard-Jones potential, and an NP-hard combinatorial optimization problem. CSPCA performs better than its opponents, showing potential to be used in other practical optimization problems.

*Key words:* Metropolis Algorithms, Particle Collision Algorithm, Combinatorial Optimization, Parameter Estimation, Nonlinear systems, Lennard-Jones potential.

## 1 Introduction

The Particle Collision Algorithm (PCA) [51, 52, 48] is a Metropolis-based algorithm [38] that was introduced as an alternative to Simulated Annealing [28]. The main motivation behind the PCA was that in spite of being very powerful, the canonical simulated annealing is too sensitive to the choice of free parameters, such as, for example, the annealing schedule and initial temperature [8]. The PCA does not rely on user-supplied parameters other than the number of iterations for the local search phase to perform the optimality search, being thus more robust. This algorithm is loosely inspired by the physics of nuclear particle collision reactions [13], particularly scattering and absorption. Thus, a particle that hits a high-fitness "nucleus" is "absorbed" and explores the boundaries, which means that stochastic solutions are generated in the same region of the search space. On the other hand, a particle that hits a low-fitness region is scattered to another region of the search space. The PCA, on its canonical form or variants, has been successfuly applied to many real-world optimization problems, see [52, 54, 29, 1, 48, 10, 35, 4], among others.

In this article, we introduce an enhanced version of the PCA, which incorporates the concept of cross-sections [13], being thus called the Cross-Section Particle Collision Algorithm (CSPCA). The idea behind it is very simple. First, we generate solutions through the search space using the Sobol quasi-random generator [57], which produces a very uniform space covering [16]. These points and their respective fitness-function values are analogous to the neutron cross sections which are used to express the likelihood of interaction

* Corresponding author: wagner.sacco@ufopa.edu.br

between an incident neutron and a target nucleus. Thus, regions with better fitness values will be more likely to be visited, in the same way that a high value of cross-section is prone to attract more neutrons.

CSPCA is compared against an enhancement of the PCA which uses the Hooke-Jeeves local search algorithm [25, 5], HJPCA [48], and also to a state-of-the-art stochastic optimization algorithm: differential evolution (DE) [58], which outperformed the genetic agorithm and particle swarm optimization in extensive tests [60]. These comparisons are made applying the methods to five real-world problems: two parameter estimation problems, a nonlinear system, the ten-atom Lennard-Jones potential, and an NP-hard combinatorial optimization problem.

The remainder of the paper is described as follows. The description of the canonical PCA is presented in Section 2. The novel PCA is explained in section 3. The practical optimization problems are described in Section 4. The computational experiments and their discussions are in Section 5. Finally, the conclusions are made in Section 6.

## 2 The Particle Collision Algorithm

The PCA algorithm resembles in its structure that of simulated annealing: first an initial configuration is chosen; then there is a modification of the old configuration into a new one. The qualities (i.e., objective function or fitness values) of the two configurations are compared. A decision is then made on whether the new configuration is "acceptable". If it is, it serves as the old configuration for the next step (current). If it is not acceptable, the algorithm

proceeds with a new change of the old configuration (trial). PCA can also be considered a Metropolis algorithm, as a trial solution can be accepted with a certain probability. This acceptance may avoid the convergence to local optima.

A summary of PCA is given below. The algorithm's default is for maximization problems. For minimization problems, just multiply the objective-function by $-1$ and invert the ratio in $p_{scattering}$.

The "stochastic perturbation" in the beginning of the loop consists in random variations in each variable's values within their ranges. Note that in PCA the perturbation mechanism is different from simulated annealing, where generally only a few variables are perturbed at a time [7].

If the quality or fitness of the trial solution configuration is better than the fitness of the current one, then the "particle" is "absorbed", there is an exploitation of the boundaries searching for an even better solution. In this work, this exploitation is performed using the classical Hooke-Jeeves pattern search method, which performed better than the original scheme proposed in the canonical version of the PCA (see [48] for extensive comparisons). Note that as the Hooke-Jeeves method was conceived as a minimization algorithm, we had to change the signs of the objective-function values in function "Hooke-Jeeves" in order to fit the PCA.

Otherwise, if the quality of the trial is worse than the current's, the "particle" is "scattered". By scattering we mean that the new configuration receives random values between the lower and upper bounds of each variable. The scattering probability $p_{scattering}$ is inversely proportional to its fitness function value. Thus, a low-fitness particle will have a greater scattering probability.

This mechanism is essential to the success of the algorithm, as it introduces a purely random component, which is responsible to avoid getting stuck at local optima.

***Initialization Step***

Generate an initial random solution **current**.

***Main Step***

Do until termination criterion is reached:

(1) Generate a stochastic perturbation **trial** of the solution **current**:
**trial** $\leftarrow$ **current** + ((**Upper** − **current**) $*$ **rand**$(0,1)$) − ((**current** − **Lower**) $*$ $(1 - \textbf{rand}(0,1))$)

(2) If Quality(**trial**) > Quality(**current**), let
**current** $\leftarrow$ **trial** and go to *HookeJeeves*.
Otherwise, go to *Scattering*.

End-Do

***HookeJeeves***

(1) Apply the HJ method to the solution $-1 *$ **current**.

(2) If Quality(**trial**) > Quality(**current**), let
**current** $\leftarrow$ **trial**.

return

***Scattering***

(1) Calculate $p_{scattering} = 1 - (\text{Quality}(\textbf{trial})/\textbf{CurrentBest})$.

(2) If $p_{scattering} > \textbf{random}(0, 1)$, let **current** receive a random solution. Otherwise, go to *HookeJeeves*.

return

Regarding the *Scattering* routine, we have the attribution

$$p_{scattering} = 1 - \frac{\text{Quality}(\textbf{trial})}{\textbf{CurrentBest}}. \tag{1}$$

It is quite clear that **CurrentBest** may assume very small values, leading to a division by zero and to an infinite value for the fractional term. To remedy this situation, we created a routine called *Divide-Check*, which checks the occurrence of a division by zero. In case it occurs, $p_{scattering}$ receives the following value:

$$p_{scattering} = -signal[\text{Quality}(\textbf{trial})](MaxMach), \tag{2}$$

where $MaxMach$ is the magnitude of the largest number representable by the machine under use. Note that

$$signal[x] = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{3}$$

## 3 The Cross-Section Particle Collision Algorithm

Below, the pseudocode of CSPCA, followed by a detailed explanation of the algorithm. Note that at this time we are assuming a minimization problem.

***Initialization Step***

(1) Generate $NP$ points $\mathbf{x}_i$, $i = 1, 2, \ldots, NP$, using the Sobol sequence.

(2) Sort these points by increasing fitness order.

(3) Generate an initial random solution **current**.

***Main Step***

Do until termination criterion is reached:

(1) Generate a stochastic perturbation **trial** of the solution **current**:
**trial** $\leftarrow$ **current** $+ F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2})$, where $r_1$ and $r_2$ are distinct random numbers in the interval $[1, NP]$ and $F$ is a scaling factor.

(2) If Quality(**trial**) $<$ Quality(**current**), let
**current** $\leftarrow$ **trial** and go to *HookeJeeves*.
Otherwise, go to *Scattering*.

End-Do

***HookeJeeves***

(1) Apply the HJ method to the solution **current**.

(2) If Quality(**trial**) < Quality(**current**), let

**current** ← **trial**.

return

***Scattering***

(1) Select a point $\mathbf{x}_r$ by rank weighting.

(2) Let $p_{scattering} = P_{\mathbf{x}_r}$.

(3) If $p_{scattering} > aux\_rand = \mathbf{random}(0, 1)$, let

**trial** $\leftarrow \mathbf{x}_r + (\mathbf{U} - \mathbf{x}_r) * aux\_rand - (\mathbf{x}_r - \mathbf{L})(1 - aux\_rand)$, where $\mathbf{U}$ and $\mathbf{L}$ are the upper and lower bound vectors.

Otherwise, go to *HookeJeeves*.

return

Besides the generation of a random solution, the cross-section particle collision algorithm begins with the generation of $NP$ points in the search space that will characterize the space regions in the same way that the cross-sections do with the reactor. These points are generated using the Sobol quasi-random sequence, briefly described below.

The points in a quasi-random or low-discrepancy sequence are designed to maximally avoid each other. As Maaranen et al. [33] say

*while the points generated using pseudorandom numbers try to imitate random points, the points generated using quasi-random sequences try to imitate*

*points with a perfect uniform distribution.*

Discrepancy is a measure of deviation from uniformity of a given sequence of points, distributed within another set of points. Here, it is sufficient to consider the discrepancy of $S = \{q_1, q_2, \ldots, q_n\} \subset [0.1]^n$, an arbitrary $n$-dimensional finite sequence generated in $[0,1]^n$, the unit hypercube of $\Re^n$. Thus, we say that $D_N(S)$ is the discrepancy of $S$ in $[0,1]^n$ if

$$D_N(S) = \mathrm{Sup}_{\Im} |F_N(\Im) - M(\Im)|, \tag{4}$$

where the supremum is taken over all subintervals $\Im$ of the unit hypercube, $F_N(\Im)$ is the number of points in $S \cap \Im$ divided by $N$, and $M(\Im)$ is the measure of $\Im$ [18]. Note that the magnitude $|F_N(\Im) - M(\Im)|$ represents the relative size of the points fraction of $S$ in $\Im$. $S = \{q_1, q_2, \ldots, q_n\}$ is considered a sequence with low discrepancy in $[0,1]^n$ if $D_N(S)$ becomes small, for $N$ sufficiently large.

The Sobol sequence, which we use in this work, generates numbers between 0 and 1 directly as a binary fraction of length $w$ bits, from a set of $w$ special binary fractions, $V_i$, $i = 1, 2, \ldots, w$, so-called direction numbers [45]. For further details, see, for example, Niederreiter [42] or Gentle [18].

The points generated by Sobol are sorted by increasing fitness order, so that $\mathbf{x}_1$ is the point with best fitness.

Then comes the main step, where a trial solution is iteratively generated and tested in terms of fitness. The expression

$$\mathbf{trial} \leftarrow \mathbf{current} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) \tag{5}$$

was borrowed from differential evolution's mutation mechanism [58]. In the expression above, $F$ is the scaling factor, **current** is the so-called base-vector and the points $\mathbf{x}_{r_1}$ and $\mathbf{x}_{r_2}$ are randomly selected from the set generated by the Sobol sequence.

The exploitation phase, performed by the Hooke-Jeeves routine, remains identical to the PCA.

The scattering routine, by its turn, is completely new. In CSPCA, $p_{scattering} = P[\mathbf{x}_r]$, the selection probability of the individual $\mathbf{x}_r$. This individual is selected by a problem-independent approach called "rank weighting" [20], originally used in the genetic algorithm [24]. First, the individuals must be ranked by fitness, which we did in the "Initialization Step". Then, the selection probability of a point $r$ is given by the following equation [20]:

$$P_r = \frac{NP - r + 1}{\sum_{i=1}^{NP} i}, \tag{6}$$

where $NP$ is the total number of points. For each point $r$, besides $P_r$, we also calculate the cumulative probability, which is given by $\sum_{i=1}^{r} P_i$. A random number between zero and one is generated. Starting at the top of the list (i.e., the first individual in the ranking), the first point with a cumulative probability that is greater than the random number is selected [20]. Therefore, the selected individual is the $r$th individual $\mathbf{x}_r$.

Then, if $p_{scattering} > aux\_rand = \mathbf{random}(0, 1)$, then a new trial solution taking $\mathbf{x}_r$ as base is generated (see the pseudocode for the expression). Otherwise, the trial solution is used as a starting point for the Hooke-Jeeves method, using the Metropolis criterion.

## 4 The Practical Problems

### *4.1 Catalytic Reactor Model*

In this parameter estimation problem, it is assumed that there are measurement errors in all variables. In order to solve it,it is necessary to use the error–in–variables approach [59]. Using this model, the objective–function has the form [17]

$$\min_{\theta,\tilde{x}_i} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{(\tilde{x}_{ij} - x_{ij})^2}{\sigma_j^2} \tag{7}$$

subject to

$$f(\theta, \tilde{\mathbf{x}}_i) = \mathbf{0}, \quad i = 1, \ldots, m. \tag{8}$$

In the equations above, $\mathbf{x}_i = (x_{i1}, \ldots, x_{in})^T$ represents measurements of the variables from $i = 1, \ldots, m$ experiments, $\tilde{\mathbf{x}}_i = (\tilde{x}_{i1}, \ldots, \tilde{x}_{in})^T$ are the unknown actual values, and $\sigma_j$ is the standard deviation associated with the measurement of variable $j$ [17]. Therefore, the error–in–variables approach involves not only the parameters $\theta$, but also the true values $\tilde{x}_i$, increasing the dimensionality of the optimization problem.

This parameter estimation problem was introduced by Rod and Hančil [49], to model gas–phase catalytic hydrogenation of phenol on a palladium catalyst in pseudo–differential reactor [59]. Variables $x_1, x_2$ and $x_3$ represent the partial pressures of phenol and hydrogen, and the initial reaction rate [59]. The model is described by the following equation [59]:

$$x_3 = \frac{\theta_1 \theta_2^2 \theta_3 x_1 x_2^2}{(1 + \theta_1 x_1 + \theta_2 x_2)^3}, \tag{9}$$

where $\theta_1, \theta_2$, and $\theta_3$ are the parameters to be estimated. Standard deviations of 0.0075, 0.0075 and 2.5 are specified for $x_1, x_2$ and $x_3$, respectively [59]. Table ?? (Appendix A) presents the data and the fitted values. The optimal parameters of this 59–variable problem are equal to $\theta_1 = 7.39696$ atm$^{-1}$, $\theta_2 = 0.63782$ atm$^{-1}$, and $\theta_3 = 1769.71$ mol/kg h, corresponding to an objective value of 30.3072 [17]. For variables $\tilde{x}_i$, we use the same same search region as in [17]: $[x_{1i} - 3\sigma_1, x_{1i} + 3\sigma_1]$ and $[x_{2i} - 3\sigma_2, x_{2i} + 3\sigma_2]$, for $i = 1, \ldots, 28$. For the parameters to be estimated, we adopt a wider range than these authors, $\theta_1, \theta_2 \in [0, 10]$ and $\theta_3 \in [1000, 2000]$.

### *4.2 Fertilizer Experiment Parameter Estimation*

This parameter estimation problem was originally presented by Hartley [19]. Table ??, found in Appendix B, shows the results of an experiment where the variable $y$ represents the yields of wheat corresponding to six rates of application of fertilizer, $x$, on a coded scale. The idea is to fit these data to the exponential law of "diminishing returns" [19], given by [14]:

$$y = k_1 + k_2 \exp(k_3 x), \tag{10}$$

where $k_1, k_2$ and $k_3$ are the parameters to be estimated by a least–squares objective function. The optimal value of this function is given by $f(\mathbf{k}^*) = 13390.23$, where $\mathbf{k}^* = (523.3, -156.9, -0.1997)$ [19]. Such a high functional value is due to the low number of data points [14]. In this work, we define the search region for $k_1, k_2$ and $k_3$ as $[-10^5, 10^5] \times [-10^5, 10^5] \times [-10^2, 10^2]$.

### *4.3 Lennard–Jones Potential*

The optimization of Lennard–Jones clusters is a two–body problem used to simulate heavy atom rare gas clusters such as argon, xenon and krypton [15]. The Lennard–Jones (LJ) potential energy between two atoms with distance $r$ is given by [40]:

$$LJ(r) = \frac{1}{r^{12}} - \frac{2}{r^6}. \tag{11}$$

The total potential energy $V$ of a cluster of $N$ atoms is given by

$$V = \sum_{i<j} LJ(r_{ij}), \tag{12}$$

where $r_{ij}$ is the Euclidian distance between atoms $i$ and $j$, whose coordinates are given, respectively, by $\mathbf{X}_i = (x_i, y_i, z_i)$ and $\mathbf{X}_j = (x_j, y_j, z_j)$.

We define the search region as in [39]. To eliminate all translational and rotational degrees of freedom in the microcluster, we set $x_1 \equiv y_1 \equiv z_1 \equiv y_2 \equiv z_2 \equiv z_3 = 0$, reducing the problem dimensionality to $3N - 6$ [15]. Let $u_1 = x_2, u_2 = x_3$, and $u_3 = y_3$. Thus, the LJ problem with three atoms has dimension equal to 3. For more than three atoms, further coordinates $(u_i, u_{i+1}, u_{i+2})$ are added such that [39]:

$$u_i = \begin{cases} x\lfloor\frac{i+8}{3}\rfloor & \text{if} \quad \text{mod}\,(i+8,3) = 0 \\ y\lfloor\frac{i+8}{3}\rfloor & \text{if} \quad \text{mod}\,(i+8,3) = 1 \\ z\lfloor\frac{i+8}{3}\rfloor & \text{if} \quad \text{mod}\,(i+8,3) = 2 \end{cases} \quad , i = 4, \ldots, 3N-6. \tag{13}$$

The search region $\Omega$ is given by [39]:

$$\Omega = \bigcup_i \Omega_i, \tag{14}$$

where

$$\Omega_i = \begin{cases} [0, 4] & \text{if } i = 1 \text{ or } 2 \\ [0, \pi] & \text{if } i = 3 \\ \left[-4 - \frac{1}{4}\left\lfloor \frac{i-4}{3} \right\rfloor, 4 + \frac{1}{4}\left\lfloor \frac{i-4}{3} \right\rfloor\right] & \text{if } i = 4, \ldots, 3N-6 \end{cases} . \tag{15}$$

The Lennard–Jones potential energy minimization is an NP–hard problem, where the number of local minima of an $N$–atom microcluster grows as $\exp(N^2)$ [34]. Here, as in [11], we use use $N = 10$, which has $V = -28.422532$ as minimum [15].

### *4.3.1 Non–Adiabatic Stirred Tank Reactors*

This application, introduced by Kubíček et al. [30], is a model of two continuous non–adiabatic stirred tank recators, which yields in a system of two non–linear equations and two variables ($\phi_1$ and $\phi_2$) that represent the dimensionless temperature in two reactors:

$$\begin{cases} f_1 = (1-R)\left[\frac{D}{10(1+\beta_1)} - \phi_1\right] \exp\left(\frac{10\phi_1}{1+10\phi_1/\gamma}\right) - \phi_1 \\ f_2 = \phi_1 - (1+\beta_2)\phi_2 + (1-R)\left[\frac{D}{10} - \beta_1\phi_1 - (1+\beta_2) - \phi_2\right] \exp\left(\frac{10\phi_2}{1+10\phi_2/\gamma}\right) \end{cases}, \tag{16}$$

where $0 \leq \phi_1, \phi_2 \leq 1$, $D = 22, \beta_1 = \beta_2 = 2, \gamma = 1000$ and $R$ is the recycle ratio parameter, which takes values in the set $R = \{0.935, 0.940, \ldots, 0.995\}$.

Depending on the value of $R$, Eq. (16) may have from one to seven solutions. In this work, we employed $R = 0.935$, which has a single solution: $(0.724987, 0.245241)$ [15]. Despite having only two variables, it is an extremely challenging problem, being a benchmark in the literature (see, for example, [23, 21]).

### *4.4 The Turbine Balancing Problem*

The turbine balancing problem is a real challenge for optimization methods, as it is NP-hard [44]. It was originally proposed by Mosevich [41] as a combinatorial optimization problem [43], being also formulated as a quadratic assigment problem [32]. Since then, it has been attacked by other researchers, using both formulations and different kinds of turbines [56, 3, 44, 9].

In this work, we solve the case presented by Mosevich [41]. The problem consists in balancing the runners of a Francis hydraulic turbine. Sinclair [56] gives a precise description of the problem to be solved:

> *A hydraulic turbine runner consists essentially of a cylinder with blades attached to its circumference. The turbine rotates as water flows across the blades. During the manufacturing process the individual blades must be welded into place, equally spaced around the cylinder. The problem encountered during this phase is the static balancing of the completed runner. Because of the complexity of the manufacturing process, the final weights of the blades may differ substantially. The result is an unbalanced runner. Since the runner can rotate at very high revolutions during use, it is crucial that the unbalance be as small as possible, otherwise the bearings on which*

*the runners rotate will wear out very quickly.*

We must add that, according to Mosevich [41], the variations in final weight mentioned above can be as great as $\pm 5\%$.

Let us formulate the problem, following [41]. The runner is modeled as $n$ equally-spaced weights on a circle of zero mass and radius $r$ equal to the common distance from the blade centers-of-mass to the runner axis. The blade positions are labeled counterclockwise, starting at position $1 = (r, 0)$ in an $x-y$ coordinate system, receiving indexes $k = 1, 2, \ldots, n$. Let $P_t$ be a configuration of blades where $P_t(j) = k$ assigns blade $k$ to position $j$. First, we define the following variables:

$M^k$ = mass of blade $k$;

$M_j^k$ = mass of blade $k$ when in position $j$;

$\theta_j = (2\pi/n)(j-1)$ = angle between position $j$ and position $1, j = 1, \ldots, n$;

$M$ = total mass of blades = $\sum_{k=1}^{n} M^k$.

Then, each permutation $P_t$ determines a center of mass $(\bar{x}, \bar{y})$ given by:

$$\bar{x}(P_t) = \frac{1}{M} \sum_{j=1}^{n} M_j^k r \cos\theta_j, \tag{17}$$

$$\bar{y}(P_t) = \frac{1}{M} \sum_{j=1}^{n} M_j^k r \sin\theta_j, \tag{18}$$

Finally, Eq. (17) and Eq.(18) define deviation $\bar{D}$,

$$\bar{D}(P_t) = \sqrt{[\bar{x}(P_t)]^2 + [\bar{y}(P_t)]^2]}, \tag{19}$$

which is the objective function to be minimized. $\bar{D}(P_t) = 0$ means that a perfect static balance has been reached [41].

As suggested by Mosevich [41], we scale the problem making $r = 1$. We use

$n = 14$ blades, as a typical runner has between 14 and 18 blades [41], and this value of $n$ is one of the most difficult to optimize [32]. Regarding the values of $M_k$, we follow [32], generating $n$ numbers according to a normal distribution with a mean of 100 and a standard deviation of 5/3, so that most $M_k$s fall within $\pm 5\%$ of the mean. We generated these numbers using a Gaussian Random Number Generator available at the Random.org website [47].

## 5 Results and Discussion

### *5.1 Implementation and Setup*

Our tests were performed on an Intel® Core™ i7 PC with 12 Gb RAM running Ubuntu 14.04 LTS. Our algorithms were implemented in C++ and compiled with GNU g++ version 4.8.2. For the stochastic part of this algorithm, we used the pseudorandom number generating algorithm developed by Matsumoto and Nishimura [37], the Mersenne Twister, which is available for download at one of its creator's website [36]. The Sobol sequence generator used in CSPCA was downloaded from Dr. Frances Kuo's website [31]. The code, freely available there, is the one presented by Joe and Kuo [26] with some enhancements [27]. It can generate sequences up to 21,201 dimensions [31].

Regarding the turbine balancing problem, as the algorithms used here were conceived for continuous optimization algorithm [58, 52], first, we need to adapt them for combinatorial optimization. In order to do so, we employ a representation technique named random keys [6]. This mechanism, originally

designed for the genetic algorithm, allows us to treat discrete problems as if they were continuous. The solution is translated into a discrete sequence only in the moment of the objective function evaluation. Let us show how it works with a simplified example: a six-component combinatorial optimization problem. DE works with six continuous variables, all in the range $[0, 1]$. Let us suppose we have a solution $\mathbf{S}_1$, given by

$$\mathbf{S}_1 = (0.18, 0.73, 0.42, 0.87, 0.01, 0.23). \tag{20}$$

Each one of these variables receive an integer index, in subscripts, corresponding to their order of appearance:

$$\mathbf{S}_1 = (0.18_1, 0.73_2, 0.42_3, 0.87_4, 0.01_5, 0.23_6). \tag{21}$$

Then, these real numbers (the so-called random keys) are sorted:

$$\mathbf{S}_{1_{\text{sorted}}} = (0.01_5, 0.18_1, 0.23_6, 0.42_3, 0.73_2, 0.87_4). \tag{22}$$

The subscripts represent a valid sequence $\mathbf{T}_1$:

$$\mathbf{T}_1 = (5, 1, 6, 3, 2, 4). \tag{23}$$

Note that, even in an extreme case with repeated real numbers, a valid sequence is produced:

$$\mathbf{S}_2 = (0.93, 0.27, 0.93, 0.45, 0.11, 0.93), \tag{24}$$

$$\mathbf{S}_2 = (0.93_1, 0.27_2, 0.93_3, 0.45_4, 0.11_5, 0.93_6), \tag{25}$$

$$\mathbf{S}_{2_{\text{sorted}}} = (0.11_5, 0.27_2, 0.45_4, 0.93_1, 0.93_3, 0.93_6), \tag{26}$$

$$\mathbf{T_2} = (5, 2, 4, 1, 3, 6). \tag{27}$$

For DE, we used $NP = 100$ (population size), $CR = 0.9$ (crossover rate), and $F = 0.5$ (scaling factor). These values have been widely employed in the literature [60, 2, 46, 53].

For the new method, we used $NP = 50d$, where $d$ is the problem's number of dimensions, and $F = 0.5$. The Hooke-Jeeves routine inside the PCA and CSPCA was set up with $\Delta = 10^{-3}$, $\varepsilon = 10^{-6}$ and $\alpha = 0.8$.

As the problems used here have known global minima, the algorithms were run using the same termination criterion as, for example, in Siarry et al. [55], Hirsch et al. [22], and Rios-Coelho et al. [48], which is ideal for an algorithm's performance assessment:

$$|f(\mathbf{x}^*) - f(\mathbf{x})| \leq \varepsilon_1 |f(\mathbf{x}^*)| + \varepsilon_2, \tag{28}$$

where $f(\mathbf{x}^*)$ is the global optimum, $f(\mathbf{x})$ is the current best, coefficient $\varepsilon_1 = 10^{-4}$ corresponds to the relative error and $\varepsilon_2 = 10^{-6}$ corresponds to the absolute error [55].

As a safeguard, we set a maximum number of fitness function evaluations equal to $10,000,000$ for all methods tested herein as a stopping criterion, in case the condition given by Eq. (28) is not achieved.

We performed one-hundred executions for each algorithm, using one-hundred previously selected random seeds in all of them.

*5.2 Results*

The results obtained in our experiments are displayed in Tables 1-5, in terms of success rate (SR), fitness value and number of fitness evaluations (NFE). For the fitness values, we determined the average (Avg.), median (Mdn.), standard deviation (Std. Dev.), maximum (Max.) and minimum (Min.) of all runs, successful or unsuccessful. For NFE, by its turn, we only computed these values for successful executions.

Table 1 shows the results for the catalytic reactor model problem. As one can see, CSPCA and PCA obtained a success rate of 100/100, while differential evolution achieved 7/100. In terms of computational effort (NFE), CSPCA was better than PCA.

Analyzing Table 2, the reader will note that, once more, CSPCA and PCA outperformed DE in terms of success rate. Comparing CSPCA against PCA, we can see that PCA was more accurate for the fitness values, but CSPCA demanded less fitness evaluations.

In Table 3, we can see that CSPCA was the most effective, including computational effort. It is worth mentioning that the Lennard-Jones potential problem is a great challenge for a general-purpose technique like CSPCA. This problem requires specialized techniques, like basin-hopping [61]. Therefore, CSPCA's results are quite surprising.

For the tank reactor problem (Table 4), PCA was clearly the best. CSPCA does not obtain 100/100 success and demands a high machine effort.

Before commenting the results displayed in Table 5, we must remind the reader

that general-purpose optimization methods like those tested here are not as effective as, for example, Ant-Colony Optimization [12], for combinatorial optimization problems. This explains the relatively low SRs obtained by DE, PCA, and CSPCA. However, CSPCA obtains an acceptable success rate. On the other hand, it requires more function evaluations than the other methods.

Overall, except for the non-adiabatic stirred tank reactor, CSPCA obtained the best performance. We must add that PCA, the second best, had obtained an excellent performance in a previous work [48], outperforming two state-of-the art algorithms.

## 6 Conclusions

In this work, we introduce a new Metropolis algorithm, the Cross-Section Particle Collision Algorithm, which is an enhancement of the Particle Collision Algorithm. Our initial tests were performed applying the CSPCA to five real-world optimization benchmarks. CSPCA outperformed not only PCA, but also the state-of-the-art stochastic optimization method differential evolution.

The results obtained in this work are encouraging, giving us a good prospect to attack other practical optimization problems in the future.

As further research, we are also planning to conceive a populational version of the CSPCA, similar to the PCA variants introduced by Sacco et al. [50] and by Da Luz et al. [10].

## Acknowledgements

W.F.S. and N.H. gratefully acknowledge the financial support provided by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico, Ministry of Science, Technology and Innovation, Brazil). The research by N.H. has been carried out within the framework of project PROCIENCIA-UERJ financed by FAPERJ.

## Appendix A - Data and fitted variables for the Catalytic Reactor Model

## Appendix B - Data for the Fertilizer Experiment problem

## References

[1] A. Abuhamdah and M. Ayob. Hybridization multi-neighbourhood particle collision algorithm and great deluge for solving course timetabling problems. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 108–114. IEEE, 2009.

[2] M. M. Ali and A. Törn. Population set-based global optimization algorithms: some modifications and numerical studies. *Computers and Operations Research*, 31(10):1703–1725, 2004.

[3] S. V. Amiouny, J. J. Bartholdi III, and J. H. Vande Vate. Heuristics for balancing turbine fans. *Operations Research*, 48(4):591–602, 2000.

[4] J. Anochi and H. Campos Velho. Optimization of feedforward neural network by multiple particle collision algorithm. In *FOCI 2014: 2014 IEEE*

*Symposium on Foundations of Computational Intelligence, Proceedings*, pages 128–134. IEEE, 2015.

[5] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms.* John Wiley and Sons, 2013.

[6] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[7] S. P. Brooks and B. J. Morgan. Optimization using simulated annealing. *The Statistician*, 44(2):241–257, 1995.

[8] J. N. Carter. Genetic algorithms for incore fuel management and other recent developments in optimisation. *Advances in Nuclear Science and Technology*, 25:113–154, 1997.

[9] W. Choi and R. H. Storer. Heuristic algorithms for a turbine-blade-balancing problem. *Computers and Operations Research*, 31(8):1245–1258, 2004.

[10] E. Da Luz, J. Becceneri, and H. Campos Velho. Multiple particle collision algorithm applied to radiative transference and pollutant localization inverse problems. In *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 347–351. IEEE, 2011.

[11] K. Deep and M. Arya. Minimization of lennard-jones potential using parallel particle swarm optimization algorithm. *Communications in Computer and Information Science*, 94 CCIS(PART 1):131–140, 2010.

[12] M. Dorigo and T. Stützle. *Ant colony Optimization.* The MIT Press, 2004.

[13] J. J. Duderstadt and L. J. Hamilton. *Nuclear reactor analysis.* John Wiley and Sons, Inc., New York, 1976.

[14] P. Englezos and N. Kalogerakis. *Applied parameter estimation for chem-*

*ical engineers.* CRC Press, 2001.

[15] C. A. Floudas, C. Adjiman, W. Esposito, Z. Gümüs, and et al. *Handbook of test problems in local and global optimization.* Dordrecht: Springer–Science+Business Media, B.V., 1999.

[16] S. Galanti and A. Jung. Low-discrepancy sequences: Monte carlo simulation of option prices. *The Journal of Derivatives*, 5(1):63–83, 1997.

[17] C. Gau and M. Stadtherr. Deterministic global optimization for errors-in-variables parameter estimation. *AIChE Journal*, 48:1192–1197, 2002.

[18] J. E. Gentle. *Random number generation and Monte Carlo methods.* Springer, 2003.

[19] H. O. Hartley. The modified gauss-newton method for the fitting of non-linear regression functions by least squares. *Technometrics*, 3(2):269–280, 1961.

[20] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms.* Wiley Interscience, 2004.

[21] N. Henderson, W. F. Sacco, N. E. Barufatti, and M. Ali. Calculation of critical points of thermodynamic mixtures with differential evolution algorithms. *Industrial and Engineering Chemistry Research*, 49(4):1872–1882, 2010.

[22] M. J. Hirsch, C. Meneses, P. M. Pardalos, and M. G. Resende. Global optimization by continuous grasp. *Optimization Letters*, 1(2):201–212, 2007.

[23] M. J. Hirsch, P. M. Pardalos, and M. G. Resende. Solving systems of non-linear equations with continuous grasp. *Nonlinear Analysis: Real World Applications*, 10(4):2000–2006, 2009.

[24] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelli-*

*gence.* University of Michigan Press, 1975.

[25] R. Hooke and T. A. Jeeves. "direct search"solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

[26] S. Joe and F. Y. Kuo. Remark on algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 29(1):49–57, 2003.

[27] S. Joe and F. Y. Kuo. Constructing sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654, 2008.

[28] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[29] D. Knupp, A. Silva Neto, and W. F. Sacco. Radiative properties estimation with the luus-jaakola and the particle collision algorithm. *CMES - Computer Modeling in Engineering and Sciences*, 54(2):121–145, 2009.

[30] M. Kubíček, H. Hofmann, V. Hlavaček, and J. Sinkule. Multiplicity and stability in a sequence of two nonadiabatic nonisothermal cstr. *Chemical Engineering Science*, 35(4):987–996, 1980.

[31] F. Y. Kuo. Sobol sequence generator. Website, 2010. `http://web.maths.unsw.edu.au/~fkuo/sobol/index.html` (retrieved 04-28-2015).

[32] G. Laporte and H. Mercure. Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35(3):378–381, 1988.

[33] H. Maaranen, K. Miettinen, and M. M. Makela. Quasi-random initial population for genetic algorithms. *Computers and Mathematics with Applications*, 47(12):1885–1895, 2004.

[34] C. Maranas and C. Floudas. A global optimization approach for lennard-jones microclusters. *The Journal of Chemical Physics*, 97(10):7667–7678,

1992.

[35] Y. Martínez González, J. Martínez Rodríguez, A. Da Silva Neto, and P. Gomes Watts Rodrigues. Estimation of open channels hydraulic parameters with the stochastic particle collision algorithm. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 36(1):69–77, 2014.

[36] M. Matsumoto. Mersenne twister home page. Website, 2011. `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html` (retrieved 04-13-2015).

[37] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[38] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21:1087–1092, 1953.

[39] N. Moloi. Microcluster minimization using differential evolution with local search. *Johannesburg: Faculty of Science, University of the Witwatersrand*, 2001.

[40] N. Moloi and M. Ali. An iterative global optimization algorithm for potential energy minimization. *Computational Optimization and Applications*, 30(2):119–132, 2005.

[41] J. Mosevich. Balancing hydraulic turbine runners – a discrete combinatorial optimization problem. *European Journal of Operational Research*, 26(2):202–204, 1986.

[42] H. Niederreiter. *Quasi-Monte Carlo Methods*. John Wiley and Sons, Ltd, 1992.

[43] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity.* Dover Publications, 1998.

[44] L. S. Pitsoulis, P. M. Pardalos, and D. W. Hearn. Approximate solutions to the turbine balancing problem. *European Journal of Operational Research*, 130(1):147–155, 2001.

[45] W. H. Press, S. T. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes: The art of scientific computing - 3rd edition.* Cambridge University Press, 2007.

[46] R. Rahnamayan, H. Tizhoosh, and M. Salama. Opposition-based differential evolution. *IEEE Transactions on Evolutionary Computation*, 12(1):64–79, 2008.

[47] Random.org. Gaussian random number generator. Website, 2015. `http://www.random.org/gaussian-distributions/` (retrieved 03-06-2015).

[48] A. Rios-Coelho, W. F. Sacco, and N. Henderson. A metropolis algorithm combined with hooke-jeeves local search method applied to global optimization. *Applied Mathematics and Computation*, 217(2):843–853, 2010.

[49] V. Rod and V. Hancil. Iterative estimation of model parameters when measurements of all variables are subject to error. *Computers and Chemical Engineering*, 4(2):33–38, 1980.

[50] W. F. Sacco, H. Alves Filho, and C. R. de Oliveira. A populational particle collision algorithm applied to a nuclear reactor core design optimization. In *Joint International Topical Meeting on Mathematics and Computations and Supercomputing in Nuclear Applications, M&C+SNA 2007.* American Nuclear Society, 2007.

[51] W. F. Sacco and C. R. de Oliveira. June 2005. a new stochastic optimization algorithm based on particle collisions. In *2005 ANS Annual Meeting. Transactions of the American Nuclear Society*, volume 92.

[52] W. F. Sacco, C. R. de Oliveira, and C. M. Pereira. Two stochastic optimization algorithms applied to nuclear reactor core design. *Progress in Nuclear Energy*, 48(6):525–539, 2006.

[53] W. F. Sacco, N. Henderson, and A. Rios-Coelho. Topographical clearing differential evolution: A new method to solve multimodal optimization problems. *Progress in Nuclear Energy*, 71:269–278, 2014.

[54] W. F. Sacco, C. M. Lapa, C. M. Pereira, and H. Alves Filho. A metropolis algorithm applied to a nuclear power plant auxiliary feedwater system surveillance tests policy optimization. *Progress in Nuclear Energy*, 50(1):15–21, 2008.

[55] P. Siarry, G. Berthiau, F. Durdin, and J. Haussy. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. *ACM Transactions on Mathematical Software (TOMS)*, 23(2):209–228, 1997.

[56] M. Sinclair. Comparison of the performance of modern heuristics for combinatorial optimization on real data. *Computers and Operations Research*, 20(7):687–695, 1993.

[57] I. M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[58] R. Storn and K. Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[59] I.-B. Tjoa and L. Biegler. Reduced successive quadratic programming strategy for errors-in-variables estimation. *Computers and chemical engineering*, 16(6):523–533, 1992.

[60] J. Vesterstrom and R. Thomsen. A comparative study of differential

evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1980–1987. IEEE, 2004.

[61] D. J. Wales and J. P. Doye. Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

Table 1 – Results for the catalytic reactor model problem.

| | | DE | PCA | CSPCA |
|---|---|---|---|---|
| SR | | 7/100 | 100/100 | 100/100 |
| Fitness | Avg. | 30.32855 | 30.30733 | 30.30743 |
| | Mdn. | 30.32633 | 30.30721 | 30.30729 |
| | Std. Dev. | 0.01473 | 0.00037 | 0.00031 |
| | Max | 30.36771 | 30.31007 | 30.30872 |
| | Min. | 30.31023 | 30.30721 | 30.30721 |
| NFE | Avg. | 7,886,875.6 | 623,922.6 | 228,609.5 |
| | Mdn. | 8,285,824 | 224,004 | 121,110 |
| | Std. Dev. | 1,997,453.3 | 879,789.8 | 300,435.3 |
| | Max | 9,725,345 | 4,410,590 | 2,026,002 |
| | Min. | 3,867,260 | 25,578 | 40,058 |

Table 2 – Results for the fertilizer experiment parameter estimation.

| | | DE | PCA | CSPCA |
|---|---|---|---|---|
| SR | | 33/100 | 100/100 | 100/100 |
| Fitness | Avg. | 15,478.877 | 13,390.093 | 13,390.180 |
| | Mdn. | 14,064.753 | 13,390.093 | 13,390.109 |
| | Std. Dev. | 2,541.967 | 3.007E-05 | 0.195 |
| | Max | 21,665.324 | 13,390.093 | 13,391.286 |
| | Min. | 13,390.212 | 13,390.093 | 13,390.093 |
| NFE | Avg. | 17,767.5 | 9,834.4 | 5,579.3 |
| | Mdn. | 14,334 | 5,871 | 2,920 |
| | Std. Dev. | 8,696.0 | 10,859.8 | 7,412.6 |
| | Max | 45,308 | 59,522 | 38,487 |
| | Min. | 9,836 | 1,304 | 739 |

Table 3 – Results for the ten-atom Lennard-Jones potential.

| | | DE | PCA | CSPCA |
|---|---|---|---|---|
| SR | | 6/100 | 98/100 | 100/100 |
| Fitness | Avg. | -26.68799 | -28.41089 | -28.42253 |
| | Mdn. | -26.61738 | -28.42253 | -28.42253 |
| | Std. Dev. | 1.09813 | 0.09131 | 0.00001 |
| | Max | -22.25121 | -27.55586 | -28.42243 |
| | Min. | -28.41967 | -28.42253 | -28.42253 |
| NFE | Avg. | 2,998,805.5 | 2,425,235.5 | 1,530,147.8 |
| | Mdn. | 614,588.5 | 1,869,307 | 1,112,692 |
| | Std. Dev. | 3,880,899.7 | 2,160,039.1 | 1,476,123.2 |
| | Max | 9,141,852 | 9,945,778 | 6,961,748 |
| | Min. | 396,531 | 76,438 | 15,895 |

Table 4 – Results for the non-adiabatic stirred tank reactor.

| | | DE | PCA | CSPCA |
|---|---|---|---|---|
| SR | | 3/100 | 100/100 | 96/100 |
| Fitness | Avg. | 2.808E-03 | 2.240E-10 | 5.338E-06 |
| | Mdn. | 3.422E-03 | 2.240E-10 | 2.410E-10 |
| | Std. Dev. | 1.277E-03 | 3.100E-12 | 2.615E-05 |
| | Max | 3.422E-03 | 2.310E-10 | 1.334E-04 |
| | Min. | 1.572E-07 | 2.190E-10 | 2.230E-10 |
| NFE | Avg. | 3,839.7 | 270,571.6 | 2,286,373.0 |
| | Mdn. | 4,099 | 202,810 | 1,647,770 |
| | Std. Dev. | 496.7 | 258,436.9 | 2,361,400.0 |
| | Max | 4,153 | 1,672,438 | 9,501,450 |
| | Min. | 3,267 | 10,658 | 1,824 |

Table 5 – Results for the turbine balancing problem.

| | | DE | PCA | CSPCA |
|---|---|---|---|---|
| SR | | 10/100 | 13/100 | 21/100 |
| Fitness | Avg. | 3.572E-06 | 2.334E-06 | 1.755E-06 |
| | Mdn. | 2.662E-06 | 2.075E-06 | 1.775E-06 |
| | Std. Dev. | 2.926E-06 | 1.199E-06 | 7.622E-07 |
| | Max | 1.513E-05 | 5.948E-06 | 4.203E-06 |
| | Min. | 4.631E-07 | 3.940E-07 | 3.890E-07 |
| NFE | Avg. | 3,296,290.0 | 4,022,271.1 | 4,630,253.0 |
| | Mdn. | 3,436,095 | 2,744,754 | 4,000,397 |
| | Std. Dev. | 2,853,008.3 | 2,707,734.5 | 3,216,176.0 |
| | Max | 8,280,554 | 9,180,650 | 9,416,848 |
| | Min. | 87,310 | 427,012 | 7,318 |

**Table A1 (Appendix A)**

Table A.1

Data and fitted values for the Catalytic Reactor Model [49].

| $x_1$ | | $x_2$ | | $x_3$ | |
|---|---|---|---|---|---|
| Data | Fitted | Data | Fitted | Data | Fitted |
| 0.015 | 0.018 | 0.235 | 0.236 | 6.25 | 2.54 |
| 0.030 | 0.031 | 0.220 | 0.220 | 4.90 | 3.11 |
| 0.045 | 0.045 | 0.205 | 0.205 | 2.90 | 3.21 |
| 0.100 | 0.100 | 0.150 | 0.150 | 1.75 | 1.94 |
| 0.180 | 0.180 | 0.070 | 0.070 | 0.30 | 0.35 |
| 0.015 | 0.023 | 0.485 | 0.486 | 12.30 | 8.88 |
| 0.030 | 0.034 | 0.470 | 0.471 | 14.00 | 10.83 |
| 0.045 | 0.038 | 0.455 | 0.453 | 5.00 | 10.77 |
| 0.045 | 0.047 | 0.455 | 0.456 | 14.20 | 11.82 |
| 0.100 | 0.100 | 0.400 | 0.400 | 10.81 | 10.73 |
| 0.143 | 0.143 | 0.357 | 0.357 | 7.81 | 8.12 |
| 0.167 | 0.167 | 0.333 | 0.333 | 6.41 | 6.72 |
| 0.250 | 0.250 | 0.250 | 0.250 | 3.90 | 3.06 |
| 0.333 | 0.333 | 0.167 | 0.167 | 3.60 | 1.09 |
| 0.030 | 0.023 | 0.720 | 0.720 | 13.00 | 14.74 |
| 0.045 | 0.044 | 0.705 | 0.705 | 20.00 | 20.72 |
| 0.100 | 0.100 | 0.650 | 0.649 | 19.81 | 22.45 |
| 0.180 | 0.180 | 0.570 | 0.570 | 15.10 | 15.88 |
| 0.240 | 0.241 | 0.510 | 0.510 | 8.90 | 11.10 |
| 0.300 | 0.300 | 0.450 | 0.450 | 7.50 | 7.51 |
| 0.360 | 0.360 | 0.390 | 0.389 | 2.00 | 4.85 |
| 0.026 | 0.015 | 0.974 | 0.974 | 13.00 | 14.56 |
| 0.050 | 0.048 | 0.950 | 0.950 | 30.00 | 30.67 |
| 0.100 | 0.100 | 0.900 | 0.901 | 37.50 | 34.89 |
| 0.250 | 0.248 | 0.750 | 0.752 | 25.00 | 20.52 |
| 0.150 | 0.150 | 0.850 | 0.850 | 31.50 | 30.98 |
| 0.333 | 0.334 | 0.667 | 0.666 | 10.00 | 13.34 |
| 0.500 | 0.500 | 0.500 | 0.500 | 4.00 | 5.26 |

**Table B1 (Appendix B)**

Table B.1

Data for the Fertilizer Experiment Parameter Estimation Problem [19].

| x | y |
|---|---|
| -5 | 127 |
| -3 | 151 |
| -1 | 379 |
| 1 | 421 |
| 3 | 460 |
| 5 | 426 |