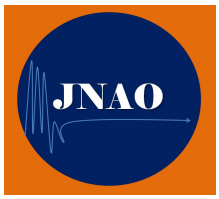
Journal of Nonlinear Analysis and Optimization
Volume 12(1) (2021)
http://www.math.sci.nu.ac.th
ISSN : 1906-9685



J. Nonlinear Anal. Optim.

A PRACTICAL APPROACH TO OPTIMIZATION

WACHIRAPONG JIRAKITPUWAPAT[1], POOM KUMAM[*,2,3], KONRAWUT KHAMMAHAWONG[2,4] AND SOMPONG DHOMPONGSA[2,3]

[1] National Security and Dual-Use Technology Center, National Science and Technology Development Agency (NSTDA), Phathum Thani, 12120, Thailand

[2] Center of Excellence in Theoretical and Computational Science (TaCS-CoE), Science Laboratory Building, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thrung Khru, Bangkok 10140, Thailand

[3] KMUTTFixed Point Research Laboratory, KMUTT-Fixed Point Theory and Applications Research Group, SCL 802 Fixed Point Laboratory, Department of Mathematics, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thrung Khru, Bangkok 10140, Thailand

[4] NCAO Research Center, Fixed Point Theory and Applications Research Group, Center of Excellence in Theoretical and Computational Science (TaCS-CoE), Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha Uthit Rd., Bang Mod, Thung Khru, Bangkok 10140, Thailand

ABSTRACT. We present a new approach for finding a minimal value of an arbitrary function assuming only its continuity. The process avoids verifying Lagrange- or KKT-conditions. The method enables us to obtain a Brouwer fixed point (of a continuous function mapping from a cube into itself).

KEYWORDS: convex algorithm, optimization, particle swarm optimization, pattern-search.
AMS Subject Classification: 46N10, 49M37, 65K05.

[*] *Corresponding author.*
Email address : poom.kumam@mail.kmutt.ac.th (P. Kumam).
Article history : Received 18 November 2020; Accepted 14 March 2021.

1. Introduction

For a given set of continuous functions $f, g_1, g_2, \ldots, g_m, h_1, h_2, \ldots, h_n : \mathcal{C} = \prod_{i=1}^{p}[c_i, d_i] \to \mathbb{R}$, a minimization problem of the form

$$\begin{aligned} &\min_{x\in\mathcal{C}} f(x) \\ \text{subject to } &g_i(x) = 0 \ (i = 1, 2, \ldots, m) \\ &h_i(x) \leq 0 \ (i = 1, 2, \ldots, n). \end{aligned} \tag{1.1}$$

is well known. For the Problem (1.1), f is called the objective function and the equalities (described by g_i) and the inequalities (described by h_i) are called the constraints. We call the set $\mathcal{A} = \{x \in \mathcal{C} : g_i(x) = 0 \ (i = 1, 2, \ldots, m)$ and $h_i(x) \leq 0 \ (i = 1, 2, \ldots, n)\}$ the feasible set of Problem (1.1). If $\mathcal{A}$ is not empty, it is compact since it is a zero set of the continuous function F defined below. Consequently, Problem (1.1) always has a solution if $\mathcal{A}$ is not empty. The subject is well understood for convex optimization with Lagrange multipliers and Karush-Kuhn-Tucker conditions are its familiar main tools. It is the purpose of this article to introduce an alternative method in minimizing a function without using the tools mentioned above. The method can be considered as a complement to the "penalty method". It transforms the constrained Problem (1.1) of f into an unconstrained one of a deformation f_t of f. "It also serves as a toolkit using for approximating a result by applying any existing software. We choose to work on some well-known software to find a decreasing sequence $\{f_t(x_n)\}$, namely, particle swarm optimization (PSO), particle-search algorithm, and convex optimization. By testing the method over many kinds of objective functions f, we believe the method is quite practical. It is found that a problem may work well under one software but not under some others. Moreover, the method can be performed to obtain a Brouwer fixed point and applied to a vector optimization.

In computational science, particle swarm optimization (PSO) [12, 13, 14] is the computational method that optimization problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. A basic variant of the PSO algorithm works by having a population (swarm) of candidate solutions (particles). These particles are moved around in the search-space according to a simple formula. The movements of the particles are guided by their own best known position in the search-space. The entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

Pattern search algorithm is a family of numerical optimization methods. It finds a sequence of points that approach an optimal point. The value of the objective function either decreases or remains the same from each point in the sequence to the next [1, 2, 8].

Convex optimization is a subfield of mathematical optimization that studies the problem of minimizing convex functions over convex sets. Convex algorithm is a mathematical method of solving convex optimization [4, 5, 7]. The key to the algorithmic success in minimizing convex functions is that these functions exhibit a local to global phenomenon. This local to global phenomenon is that local minimal of convex functions are in fact global minimal.

2. Methodology

Put $G_i = |g_i| \ \ (i = 1, 2, \ldots, m)$, $H_i = |h_i| + h_i \ \ (i = 1, 2, \ldots, n)$, and $F = \sum_{i=1}^{m} G_i + \sum_{i=1}^{n} H_i$. Clearly, F is continuous and $F(x) = 0$ if and only if x satisfies the constraints of Problem (1.1) (i.e., it lies in the feasible set $\mathcal{A}$). For large numbers K and M, set for $t \in (0, 1)$, $f_t = (1 - t)(f - K) + tMF$.
Since we are going to work on the deformed function f_t for t sufficiently close to 1, we therefore take any existing software available. We select 3 softwares, namely Particle Swarm Optimization, Pattern-Search, and Convex Algorithm. We let K to be large to be certained that the graph of $f - K$ totally lies under the graph of F. As for large M, we try to make it easy for a software to find a decreasing sequence $\{f_t(x_n)\}$. The parameter t getting close to 1 is to making the iteration point x_n being closer to or lying in the feasible set $\mathcal{A}$.

Proposition 2.1. *For any $t \in (0, 1)$ with $f_t > 0$ outside $\mathcal{A}$, x is a minimizer of Problem (1.1) if and only if x is a minimizer of f_t.*

Proof. This is straightforward since $f_t = (1 - t)(f - K)$ on $\mathcal{A}$. □

By the term "minimizer" it is meant to be a minimal element, i.e., a local minimizer.

Algorithm 1 Example code (PAO our Algorithm)

Input Set up problem 1.1
Parameter K, M, t
Output x

$G_i = |g_i| \ \ (i = 1, 2, \ldots, m)$
$H_i = |h_i| + h_i \ \ (i = 1, 2, \ldots, n)$
$F = \sum_{i=1}^{m} G_i + \sum_{i=1}^{n} H_i$
$f_t = (1 - t)(f - K) + tMF$
$x = \arg\min_{x \in C} f_t(x)$

3. Applications

3.1. **Brouwer Fixed Points.** The Brouwer fixed theorem says that any continuous mapping $T = (f_1, \ldots, f_d) : \prod_{i=1}^{d} [a_i, b_i] \to \prod_{i=1}^{d} [a_i, b_i]$ always has a fixed point. See [3, 6, 10, 9] for some new proofs. To find a fixed point of T, set in Problem (1.1), $f(x_1, x_2, \ldots, x_d) = 1$ and $g_i(x_1, x_2, \ldots, x_d) = f_i(x_1, x_2, \ldots, x_d) - x_i$ $(i = 1, 2, \ldots, d)$. (See Example 4.6 and 4.7.)

3.2. **Vector Optimization.** Given continuous mappings $f_1, f_2, \ldots, f_k, g_1, g_2, \ldots, g_m, h_1, h_2, \ldots, h_n : \mathcal{C} = \prod_{i=1}^{p} [c_i, d_i] \longrightarrow \mathbb{R}$. We need to solve

$$\begin{aligned} &\min_{x \in \mathcal{C}} (f_1(x), f_2(x), \ldots, f_k(x)) \text{ (with respect to an order)} \\ \text{subject to } &g_i(x) = 0 \ (i = 1, 2, \ldots, m) \\ &h_i(x) \leq 0 \ (i = 1, 2, \ldots, n). \end{aligned} \tag{3.1}$$

We consider the problem of the forms:

(1) $\min\limits_{x\in\mathcal{C}} \sum\limits_{i=1}^{k} f_i(x)$. Set $f = \sum_{i=1}^{k} f_i$ for the objective function in Problem (1.1). (See Example 4.8.)

(2) Finding $x^* = (x_1^*, x_2^*, \ldots, x_p^*) \in \mathcal{C}$ such that $f_i(x^*) \leq c_i$, where $c_i \leq t_i$ for some thresholds $t_i \quad (i = 1, 2, \ldots, k)$. To comply with Problem (1.1), we set $f = 1$ as an objective function and additionally define $h_i = f_i - c_i$ $(i = n+1, n+2, \ldots, n+k)$. (See Example 4.9.)

In practice, if we only want to find a point x^* with $f(x^*) \leq c$ for some assigned number c, Problem (1.1) can read as

$$\begin{aligned} &\min_{x\in\mathcal{C}} 1 \\ \text{subject to } &g_i(x) = 0 \ (i = 1, 2, \ldots, m) \\ &h_i(x) \leq 0 \ (i = 1, 2, \ldots, n) \\ &f(x) - c \leq 0. \end{aligned} \tag{3.2}$$

3.3. **Quantiles.** For a distribution function $f : \mathbb{R} \longrightarrow [0, 1]$, a quantile at $\alpha \in [0, 1]$ is defined as $f^{-1}(\alpha) = \inf\{x \in \mathbb{R} : f(x) \geq \alpha\}$. There does not exist a method to extend the concept to multivalued case. We are given a continuous function $f : \prod_{i=1}^{p}[c_i, d_i] \longrightarrow \mathbb{R}$ and $c \in \mathbb{R}$. We need to find x^* giving $f(x^*) = c$. We set Problem (1.1) as:

$$\begin{aligned} &\min_{x\in\mathcal{C}} 0 \\ \text{subject to } &f(x) = c. \end{aligned}$$

See Example 4.10.

3.4. **Non-emptiness of the feasible set.** To see if the feasible set is non-empty, we set $f_t = (1 - k)(-K) + tMF$ and find a minimizer x_t and see if $x_t \in A$, i.e., $F(x_t) = 0$. Thus, any kind of problems on non-emptiness of sets defined by sets of equations and inequalities can be verified by our method. Consequently, assumptions on non-emptiness in many theorems can be worked out. For examples, non-emptiness of fixed points of mappings assumed in various results.

4. Numerical Examples

We choose $\mathcal{C} = [-10, 10]^p, K = 100, M = 10000$ and $t = 0.95$. We experiment on nine Examples, and record results in three Tables. The Tables display approximate minimizers and constraint validation.

Example 4.1. [11]

$$\begin{aligned} \min_{x\in\mathcal{C}} \quad & x_1^2 + x_1x_2 + x_2^2 - 5x_2 \\ \text{subject to} \quad & x_1 + x_2 = 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Example 4.2. [11]

$$\begin{aligned} \min_{x\in\mathcal{C}} \quad & -(x_1 - 3)^6 - (x_2 - 4)^6 \\ \text{subject to} \quad & x_1^2 + x_2^2 \leq 25 \\ & x_1 + x_2 \geq 7 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Example 4.3. [11][Geometric Programming]

$$\begin{array}{ll} \min\limits_{x\in\mathcal{C}} & \frac{1}{x_1x_2x_3} + x_1x_2 \\ \text{subject to} & 0.5x_1x_3 + 0.25x_1x_2 \leq 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{array}$$

Example 4.4. [11]

$$\begin{array}{ll} \min\limits_{x\in\mathcal{C}} & \frac{1}{x_1x_2x_3} + x_1x_2 + x_3^7 \\ \text{subject to} & 0.5x_1x_3 + 0.25x_1x_2 \leq 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{array}$$

Example 4.5.

$$\begin{array}{ll} \min\limits_{x\in\mathcal{C}} & 4x_1 + 10x_2 + 15x_3 \\ \text{subject to} & x_1 + 2x_2 + 3x_3 = 3 \\ & 3x_1 + x_2 + 2x_3 = 7.5 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{array}$$

Example 4.6.

$$\begin{array}{ll} \min\limits_{x\in\mathcal{C}} & 1 \\ \text{subject to} & 0.5(\cos(x_1 + x_2 - x_3^4x_5))x_4 - x_1 = 0 \\ & 0.1(|x_1x_2 + x_3 - x_5| + x_4^2) - x_2 = 0 \\ & (x_1 + x_3x_4 - (x_2 + x_5)^2)/30 - x_3 = 0 \\ & (x_1 - x_2^2 + x_3 - x_5^2)/12 - x_4 = 0 \\ & (x_1 + x_2 - (x_3 + x_5 + x_4)^2)/40 - x_5 = 0 \end{array}$$

Example 4.7.

$$\begin{array}{ll} \min\limits_{x\in\mathcal{C}} & 1 \\ \text{subject to} & 0.001((x_1 + 3)^2 + (x_2 - 2)^4 + x_3^2 + x_4^2 + x_5) - x_1 = 0 \\ & 0.01(x_1 + (x_2 + 5)^2 + x_3 + x_4 + (x_5 + 2)) - x_2 = 0 \\ & 0.001(x_1^4 + (x_4 - 3)^2 + (x_5 + 2)^2) - x_3 = 0 \\ & 0.001((x_3 - 3)^4 + x_5^2 + x_1^4) - 1 - x_4 = 0 \\ & 0.01(x_1^2 + x_2 + x_3 - (x_5 - 1)^2) - x_5 = 0 \end{array}$$

Example 4.8. [11]

$$\begin{array}{ll} \min\limits_{x\in\mathcal{C}} & (x_1^2 - 5x_1 + 7x_2) + (-x_1^2 - x_2^2) + (x_1 - 1)^2 + (x_2 - 5)^2 \\ \text{subject to} & 3x_1 + 4x_2 = 6 \\ & x_1 + x_2 = 2 \\ & 2x_1 + 3x_2 \leq 6 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{array}$$

Example 4.9. [11]

$$\begin{aligned} \min_{x\in\mathcal{C}} \quad & 1 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 1 \\ & x_1^2 \leq 1 \\ & \sqrt{x_1^2 + x_2^2} - x_1^3 \leq 2 \\ & -x_1^3 + 0.5(-x_2 - x_2^3 + |x_2^3 - x_2|) \leq 0, \quad x_1, x_2 \in \mathbb{R}. \end{aligned}$$

Example 4.10. Find $f(x_1, \ldots, x_5) = \sin((x_1^2 + x_2^3 - x_3x_4^2) + \cos(x_5^3 - x_1 + x_3^2x_4^3)) + x_1x_2 + x_3^2x_4 + (1-x_2)^2 + (1-x_3)^2 + (1-x_4)^2$ restricted to $x_1^2 + 2x_2^2 - x_5 - 2x_3^2x_4 = 0$, $(1-x_3)^2 - x_2^2x_4^2 + \cos(x_4x_5^3) \leq 2$. Find $x \in A$ for which $f(x) = 19$.

TABLE 1. Particle Swarm Optimization

Example	PSO				
	initial point	value	x	$\max_{x\in\mathcal{C}} \lvert g_i(x)\rvert$	$\max_{x\in\mathcal{C}} h_j(x)$
4.1	-	-4	$(0, 1)$	0	0
4.2	-	-2	$(4, 3)$	–	0
4.3	-	0.6325	$(10, 0.0316, 10)$	-	-0.0316
4.4	-	2.4397	$(0.1141, 9.9975, 0.7715)$	-	-0.1141
4.5	-	12.6	$(2.4, 0.3, 0)$	0	0
4.6	-	1	$(-1.977\times10^{-11}, 1.02\times10^{-12}, -1.067\times 10^{-12}, -2.719\times10^{-11}, 6.04\times10^{-12})$	2.546×10^{-11}	–
4.7	-	1	$(0.018, 0.291, 0.019, -0.921, -0.007)$	1.766×10^{-12}	–
4.8	-	16	$(2, 0)$	0	0
4.9	-	1	$(0.7312, 1.0271)$	–	-0.4654
4.10	-	0	$(-4.03, -1.27, 1.93, 1.77, 6.28)$	0	9.82

TABLE 2. Pattern-Search Optimization

Example	Pattern-Search				
	initial point	value	x	$\max_{x\in\mathcal{C}} \lvert g_i(x)\rvert$	$\max_{x\in\mathcal{C}} h_j(x)$
4.1	$(1,1)$	-4	$(0,1)$	0	0
4.2	$(1,1)$	-94.3669	$(4.6094, 1.9374)$	–	0.4532
4.3	$(1,1,1)$	0.6325	$(0.6325, 0.5, 10)$	–	-0.5
4.4	$(1,1,1)$	2.4397	$(1.1385, 1, 0.7715, 1)$	–	-0.2762
4.5	$(1,1,1)$	12.6429	$(2.3571, 0, 0.2143)$	1.5259×10^{-5}	0
4.6	$(0,0,0,0,0)$	1	$(0,0,0,0,0)$	0	–
4.7	$(0,0,0,0,0)$	1	$(0.019, 0.029, 1.93, -0.92, -0.007)$	3.978×10^{-6}	–
4.8	(1,1)	18.7777	$(0.6667, 1)$	1.5259^{-5}	-0.6667
4.9	(1,1)	1	$(0,1)$	–	-1
4.10	$(0,0,0,0,0)$	0	$(-2, -2.01, -0.88, 2.13, 8.88)$	0	20.16

TABLE 3. Convex Algorithm

Example	Convex Algorithm				
	initial point	value	x	$\max_{x\in\mathcal{C}} \lvert g_i(x)\rvert$	$\max_{x\in\mathcal{C}} h_j(x)$
4.1	$(1,1)$	-3.9694	$(0.0076, 0.9924)$	7.3×10^{-9}	-0.0076
4.2	$(1,1)$	-1.2957	$(3.9302, 3.0698)$	–	-7.97×10^{-13}
4.3	$(1,1,1)$	0.6325	$(0.5623, 0.5623, 10)$	–	-0.5623
4.4	$(1,1,1)$	2.4397	$(1.0670, 1.0670, 0.7715)$	–	-0.3038
4.5	$(1,1,1)$	12.6392	$(2.3608, 0.0253, 0.1962)$	0.3167×10^{-7}	-0.0253
4.6	$(0,0,0,0,0)$	1	$(-1.520\times10^{-10}, 1.283\times10^{-11}, 1.265\times 10^{-10}, 2.282\times10^{-10}, -3.341\times10^{-11})$	2.661×10^{-10}	–
4.7	$(0,0,0,0,0)$	1	$(0.018, 0.291, 0.019, -0.921, -0.007)$	8.413×10^{-9}	–
4.8	(1,1)	16	$(2,0)$	0	0
4.9	(1,1)	1	$(-0.0888, 0.8020)$	–	-0.8013
4.10	$(0,0,0,0,0)$	0	$(-0.73, -0.53, -0.25, -0.95, 1.21)$	0	4.47

5. Discussion

In this paper, we transform a constrained optimization to an unconstrained one. Under our approach, the given objective function f (subjected to some constraints) is replaced by a deformed function f_t (without constraints) for some t. We chose to use some software packages to approximate a minimizer of f_t. We observe that all outcomes approximately satisfy corresponding constraints. Of course, we may obtain different minimizers from different software. It is challenging to construct a new algorithm for finding a global minimizer even for some special cases.

6. Appendix

In this appendix, we give the MATLAB GUI for finding a minimizer by using POA method. The MATLAB GUI of POA method is given in Figures 1, 2 and 3.

Acknowledgements

The authors acknowledge the financial support provided by the Center of Excellence in Theoretical and Computational Science (TaCS-CoE), KMUTT. The second author was supported by the "Petchra Pra Jom Klao Ph.D. Research Scholarship from King Mongkut's University of Technology Thonburi" (No. Grand 10/2560).

References

1. Mark A Abramson, *Pattern search algorithms for mixed variable general constrained optimization problems*, Tech. report, 2002.
2. Charles Audet and J. E. Dennis, Jr., *Analysis of generalized pattern searches*, SIAM J. Optim. **13** (2002), no. 3, 889–903 (2003). MR 1972220
3. L. E. J. Brouwer, *Über Jordansche Mannigfaltigkeiten*, Math. Ann. **71** (1912), no. 4, 598. MR 1511679
4. Richard H. Byrd, Jean Charles Gilbert, and Jorge Nocedal, *A trust region method based on interior point techniques for nonlinear programming*, Math. Program. **89** (2000), no. 1, Ser. A, 149–185. MR 1795061
5. Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal, *An interior point algorithm for large-scale nonlinear programming*, SIAM J. Optim. **9** (1999), no. 4, 877–900, Dedicated to John E. Dennis, Jr., on his 60th birthday. MR 1724768
6. N. Chuensupantharat, P. Kumam, and S. Dhompongsa, *A graphical proof of the Brouwer fixed point theorem*, Thai J. Math. **15** (2017), no. 3, 607–610. MR 3745100
7. Thomas F. Coleman and Yuying Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM J. Optim. **6** (1996), no. 2, 418–445. MR 1387333
8. A. R. Conn, Nick Gould, and Ph. L. Toint, *A globally convergent lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds*, Mathematics of Computation **66** (1997), no. 217, 261–289.
9. S. Dhompongsa and J. Nantadilok, *A simple proof of the Brouwer fixed point theorem*, Thai J. Math. **13** (2015), no. 3, 519–525. MR 3446192
10. Sompong Dhompongsa and Poom Kumam, *An elementary proof of the Brouwer fixed point theorem*, Thai J. Math. **17** (2019), no. 2, 539–542. MR 4009295
11. Joel Franklin, *Fixed-point theorems*, Methods of Mathematical Economics, Springer New York, 1980, pp. 224–292.
12. J. Kennedy and R. Eberhart, *Particle swarm optimization*, Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
13. Efrén Mezura-Montes and Carlos A. Coello Coello, *Constraint-handling in nature-inspired numerical optimization: Past, present and future*, Swarm and Evolutionary Computation **1** (2011), no. 4, 173–194.
14. Magnus Erik Hvass Pedersen, *Good parameters for particle swarm optimization*, Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001 (2010), 1551–3203.

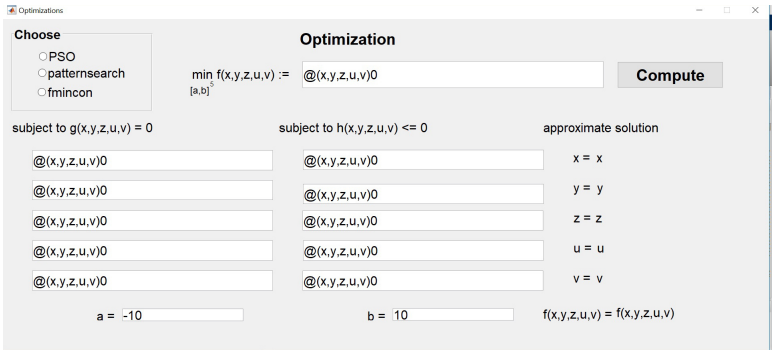


FIGURE 1. MATLAB GUI for PAO method.

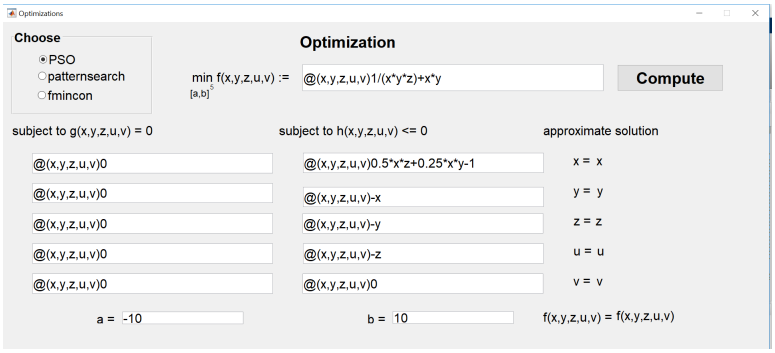


FIGURE 2. In put data.

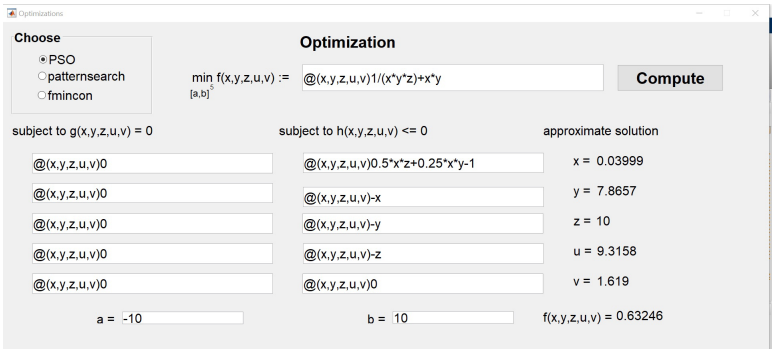


FIGURE 3. Result.