A new heuristic approach for non-convex optimization problems

R. Toscano¹, P. Lyonnet

Université de Lyon
Laboratoire de Tribologie et de Dynamique des Systèmes CNRS UMR5513 ECL/ENISE
58 rue Jean Parot 42023 Saint-Etienne cedex 2

Abstract. In this work a new optimization method, called the heuristic Kalman algorithm (HKA), is presented. This new algorithm is proposed as an alternative approach for solving continuous, non-convex optimization problems. The principle of HKA is to explicitly consider the optimization problem as a measurement process designed to give an estimate of the optimum. A specific procedure, based on the Kalman estimator, was developed to improve the quality of the estimate obtained through the measurement process. The main advantage of HKA, compared to other metaheuristics, lies in the small number of parameters that need to be set by the user. Further, it is shown that HKA converges almost surely to a near-optimal solution. The efficiency of HKA was evaluated in detail using several non-convex test problems, both in the unconstrained and constrained cases. The results were then compared to those obtained via other metaheuristics. The numerical experiments show that HKA is a promising approach for solving non-convex optimization problems, particularly in terms of computation time and success ratio.

Keywords: heuristic Kalman algorithm, non-convex optimization, metaheuristic, objective function, almost sure convergence.

1 Introduction

In all areas of engineering, physical and social sciences, problems involving the optimization of some objective function are encountered. Usually, the problem that needs to be solved can be formulated precisely, but is often difficult or impossible to solve either analytically or through conventional numerical procedures. This is the case when the problem is non-convex and thus inherently nonlinear and multimodal. In fact, it is now well-established that the convexity of optimization problems determines whether they can be efficiently solved or not [24]. Today, very efficient algorithms for solving convex problems exist [2], but the problem of non-convex optimization remains largely open, despite an enormous amount of effort that has been devoted to its resolution.

¹E-mail address: toscano@enise.fr, Tel.: +33 477 43 84 84; Fax: +33 477 43 84 99
Evolutionary approaches as well as other-population based methods handle non-convex optimization problems well [18, 10, 9, 29, 11, 28, 14, 19]. This is the reason behind the success and the broad diffusion of evolutionary optimization methods such as the well-known genetic algorithm (GA) [16, 13]. The main characteristic of these kinds of approaches, also called metaheuristics, is the use of a stochastic mechanism to find a solution. From a general point of view, the use of a stochastic search procedure appears essential for finding a promising solution.

Following this kind of approach, we propose a new optimization method, called the heuristic Kalman algorithm (HKA). Our approach falls in the category of the so-called “population-based stochastic optimization techniques”. However, its search heuristic is entirely different from other known stochastic algorithms. Indeed, HKA explicitly considers the optimization problem as a measurement process designed to give an estimate of the optimum. A specific procedure, based on the Kalman estimator, was developed to improve the quality of the estimate obtained through the measurement process. The main advantage of HKA compared to other metaheuristics, lies in the small number of parameters that need to be set by the user (only three). This property makes the algorithm easy to use for non-specialists.

The efficiency of HKA was evaluated in detail using several non-convex test problems, both in unconstrained and constrained cases. The results were then compared to those obtained via other metaheuristics. The numerical experiments show that the HKA has promising potential for solving non-convex optimization problems, notably in terms of the computation time and success ratio.

The paper is organized as follows. In section 2, HKA is presented and its convergence properties are given. In Section 3, various numerical experiments are conducted to evaluate the efficiency of HKA in solving non-convex optimization problems. Finally, section 4 concludes the paper.
Consider the general system presented in figure 1, which produces an output in response to a given input. In addition, this system has some tuning parameters that can modify its behavior. By behavior, we mean the relationship existing between the inputs and the outputs.

Figure 1: An optimization problem.

The problem is how to tune these parameters so that the system behaves well. Usually, the desired behavior can be formulated via an objective function that depends on the tuning parameters $J(q)$, which needs to be maximized or minimized with respect to $q$.

More formally, the problem to be solved can be formulated as follows: find the optimal tuning parameters $q_{opt}$, solution of the following problem:

$$
\left\{ \begin{array}{l}
q_{opt} = \arg \min_{q \in D} J(q) \\
D = \{ q : g_i(q) \leq 0, \ i = 1, \cdots, n_q \}
\end{array} \right. \tag{1}
$$

where $J : \mathbb{R}^{n_q} \rightarrow \mathbb{R}$ is a function for which the minimum ensures that the system behaves as desired and $g_i$ are some constraints on the parameters $q$. The objective is to find the tuning vector of parameters $q_{opt}$ that belong to the set of admissible solutions $D$ that minimize the cost function $J$. Unfortunately, there are several obstacles in solving this kind of problem. The main obstacle is that most optimization problems are NP-hard [12]. Therefore, the known theoretical methods cannot be applied except possibly for some small sized problems. Another difficulty is that the cost function may be not differentiable and/or multimodal. Therefore, methods that require derivatives of the cost function cannot be
used. Another obstacle is when the cost function cannot be expressed in an analytic form; in this case, the cost function can be only evaluated through simulations.

In these situations, heuristic approaches seem to be the only way to solve optimization problems. By a heuristic approach, we mean a computational method employing experimentations, evaluations and trial-and-error procedures to obtain an approximate solution for computationally difficult problems. This type of approach was used to develop HKA and is described in the next section.

2.1 Principle of the algorithm

The main idea of HKA is to generate, via experiments (measurements), a new point which is hopefully closer to the optimum than the preceding point. This process of estimation is repeated until no further improvements can be made. More precisely, we adopt the principle depicted in figure 2. The proposed procedure is iterative, and we denote by \( k \) the \( k^{th} \) iteration of the algorithm. We have a random generator of a probability density function (pdf) \( f(q) \), which produces, at each iteration, a collection of \( N \) vectors that are distributed around a given mean vector \( m_k \) with a given variance-covariance matrix \( \Sigma_k \). This collection can be written as follows:

\[
q(k) = \{ q_{1,k}, q_{2,k}, \cdots, q_{N,k} \}, \tag{2}
\]

where \( q_{i,k}^l \) is the \( i^{th} \) vector generated at iteration \( k \): \( q_{i,k} = [q_{i,1,k}^1, \cdots, q_{i,n,k}^n]^T \), and \( q_{i,k}^l \) is the \( l^{th} \) component of \( q_{i,k} \) (\( l = 1, \cdots, n \)). This random generator is applied to the cost function \( J \).

Without loss of generality, we assume that the vectors are ordered by their increasing cost function, i.e.

\[
J(q_{1,k}^1) < J(q_{2,k}^2) < \cdots < J(q_{N,k}^N). \tag{3}
\]

The principle of the algorithm is to modify the mean vector and the variance-covariance matrix of the random generator until the minimum of the cost function is reached.
More precisely, let $N_{\xi}$ be the number of best samples under consideration, such that $J(q_{k}^{N_{\xi}}) < J(q_{i}^{k})$ for all $i > N_{\xi}$. Note that the best samples are those of sequence (2) which have the smallest cost function. The objective is then to generate, from the best samples, a new random distribution which approaches, on average, the minimum of the cost function $J$ with decreasing variance.

To this end, a measurement procedure followed by an optimal estimator of the parameters of the random generator is introduced. The measurement process consists in computing the average of the candidates that are the most representative of the optimum. For the $k^{th}$ iteration, the measurement, denoted $\xi_{k}$, is defined as follows:

$$\xi_{k} = \frac{1}{N_{\xi}} \sum_{i=1}^{N_{\xi}} q_{i}^{k},$$  \hspace{1cm} (4)

where $N_{\xi}$ is the number of considered candidates. We can consider that this measurement gives a perturbed knowledge about the optimum, i.e.

$$\xi_{k} = q_{\text{opt}} + v_{k},$$  \hspace{1cm} (5)

where $v_{k}$ is an unknown disturbance, centered on $q_{\text{opt}}$, and acting on the measurement process. The components of $v_{k}$ are assumed to be independent and follow a centered normal distribution. Since the covariance between two independent variables is zero, all non-diagonal elements of variance-covariance matrix are zero. Thus, the variance-covariance matrix of $v_{k}$ is given by:
\[ V_k = \frac{1}{N_\xi} \begin{bmatrix} \sum_{i=1}^{N_\xi} (q_{i,1,k} - \xi_{1,k})^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{i=1}^{N_\xi} (q_{n,k} - \xi_{n,k})^2 \end{bmatrix}. \] (6)

In other words, the diagonal of \( V_k \) (denoted \( \text{vec}^d(V_k) \)) represents the variance vector. Note that this variance vector can be used to measure our ignorance about \( q_{opt} \). In these conditions, the Kalman filter can be used to make a so-called “a posteriori” estimate of the optimum, i.e. it accounts for both the measurement and the confidence placed in it. As seen, this confidence can be quantified by (6). Roughly speaking, a Kalman filter is an optimal recursive data-processing algorithm [21]. Optimality must be understood as the best estimate that can be made based on the model used for the measurement process as well as the data used to compute this estimate.

Our objective is to design an optimal estimator which combines a prior estimation of \( q_{opt} \) and the measurement of \( \xi_k \), so that the resulting posterior estimate is optimal in a sense which will be defined below. In the Kalman framework, this kind of estimator takes the following form:

\[ \hat{q}_k^+ = L_k' \hat{q}_k^- + L_k \xi_k, \] (7)

where \( \hat{q}_k^- \) represents the prior estimation, i.e. before measurement, \( \hat{q}_k^+ \) is the posterior estimation i.e. after measurement, \( L_k' \) and \( L_k \) are unknown matrices which have to be determined to ensure optimal estimation. Here optimality is reached when the expectation of the posterior estimation error is zero and its variance is minimal. This can be expressed as follows:

\[ \left\{ \begin{array}{l} (L_k', L_k) = \arg \min_{L_k', L_k} \mathbb{E}[\tilde{q}_k^+ \tilde{q}_k^+^T] \\ \mathbb{E}[\tilde{q}_k^+] = 0 \end{array} \right., \] (8)

where \( \mathbb{E} \) is the expectation operator and \( \tilde{q}_k^+ \) represents the posterior estimation error at
iteration $k$. We define the posterior estimation error and its variance-covariance matrix as

$$q_k^+ = q_{opt} - \hat{q}_k^+, \quad P_k^+ = \mathbb{E}[\tilde{q}_k^+ \tilde{q}_k^{+T}].$$ \hspace{1cm} (9)

Similarly, we define the prior estimation error and its variance-covariance matrix as

$$q_k^- = q_{opt} - \hat{q}_k^-, \quad P_k^- = \mathbb{E}[\tilde{q}_k^- \tilde{q}_k^{-T}].$$ \hspace{1cm} (10)

Under the assumption that $\mathbb{E}[\tilde{q}_k^-] = 0$, it can be easily established that the satisfaction of the condition $\mathbb{E}[\tilde{q}_k^+] = 0$ requires

$$L'_k = I - L_k,$$ \hspace{1cm} (11)

where $I$ is the identity matrix. Then, putting this expression into equation (7) gives:

$$\hat{q}_k^+ = \hat{q}_k^- + L_k(\xi_k - \hat{q}_k^-).$$ \hspace{1cm} (12)

The objective is now to determine $L_k$ in such a way that the variance of the posterior estimation error is minimized. Noting that $\text{trace}(P_k^+) = \mathbb{E}[\tilde{q}_k^+ \tilde{q}_k^{+T}]$, the minimization of the variance of $q_k^+$ is accomplished by minimizing the trace of $P_k^+$ with respect to $L_k$. Standard calculus, similar to the one used for the derivation of the Kalman filter, yields [21]

$$L_k = P_k^- (P_k^- + V_k)^{-1}, \quad P_k^+ = (I - L_k)P_k^-.$$

Finally, for the $k^{th}$ iteration, the HKA algorithm utilizes the relations (4), (6), (12), (13) and the next iteration is initialized with $m_k = q_k^+, \Sigma_k = P_k^+$ and $k = k + 1$. Practical considerations for efficient use of HKA are given in the next section.

### 2.2 Some practical considerations

The expression used for computing $P_k^+$ (see 13) generally leads to a decrease in the variance of the Gaussian distribution that is too fast, which results in a premature convergence of the algorithm. This difficulty can be tackled by introducing a slowdown factor in $S_k^+$ (the posterior standard deviation vector of the Gaussian generator) and adjusting it according
to the dispersion of the best candidates considered for the estimation of $q_{\text{opt}}$. This can be done as follows:

$$S_k^+ = S_k^- + a_k(W_k - S_k^-),$$

with:

$$a_k = \frac{\alpha \min(1, \left(\frac{1}{n_q} \sum_{i=1}^{n_q} \sqrt{v_{i,k}}\right)^2)}{\min(1, \left(\frac{1}{n_q} \sum_{i=1}^{n_q} \sqrt{v_{i,k}}\right)^2) + \max_{1 \leq i \leq n_q} (w_{i,k})},$$

$$S_k^- = \left(\text{vec}^d(P_k^-)\right)^{1/2}, W_k = \left(\text{vec}^d(P_k^+)^{1/2}ight.$$}

(14)

where $S_k^-$ and $S_k^+$ are, respectively, the prior and the posterior standard deviation vectors, $a_k$ is the slowdown factor, $\alpha \in (0, 1]$ the slowdown coefficient, and $v_{i,k}$ represents the $i^{th}$ component of the variance vector $\text{vec}^d(V_k)$ defined in (6), $w_{i,k}$ is the $i^{th}$ component of the vector $W_k$, and $\text{vec}^d(.)$ is the diagonal vector of the matrix $(.)$.

All the matrices used in our formulation (i.e. $P_k^+, P_k^-, L_k, \Sigma_k$) are diagonals. Consequently, to save computation time, we must use a vectorial form for computing the various quantities of interest. The vectorial form of (12) and (13) are given by

$$\hat{q}_k^+ = \hat{q}_k^- + \text{vec}^d(L_k) \otimes (\xi_k - \hat{q}_k^-)$$

$$\text{vec}^d(L_k) = \text{vec}^d(P_k^-) // (\text{vec}^d(P_k^-) + \text{vec}^d(V_k))$$

$$\text{vec}^d(P_k^+) = \text{vec}^d(P_k^-) - \text{vec}^d(L_k) \otimes \text{vec}^d(P_k^-),$$

(15)

where the symbol $\otimes$ stands for a componentwise product and $//$ represents a componentwise divide. At each iteration $k$, HKA utilizes the relations (4), (6), (14), (15) and the next iteration is initialized with $m_k = q_k^+, \Sigma_k = \text{diag}(S_k^+)^2$ and $k = k + 1$, where the notation $\text{diag}(.)$ represents a diagonal matrix with the vector $(.)$ on the diagonal. The algorithm used for the minimization of the objective function $J(q)$ is presented hereafter.

**Heuristic Kalman Algorithm**

1. Initialization. Choose $N, N_\xi$ and $\alpha$. Set $k = 0$, $\hat{q}_k^- = m_0$, $P_k^- = \Sigma_0$, $m_k = \hat{q}_k^-$, $\Sigma_k = P_k^-$.  

2. Random generator $\mathcal{N}(m_k, \Sigma_k)$. Generate a sequence of $N$ vectors $\mathbf{q}(k) = \{q_1^k, q_2^k, \cdots, q_N^k\}$ according to a Gaussian distribution parametrized by $m_k$ and $\Sigma_k$.  

8
3. Measurement process. Using relations (4) and (6), compute $\xi_k$ and $\text{vec}^d(V_k)$.

4. Optimal estimation. Using relation (15), compute $\text{vec}^d(L_k), \hat{q}_k^+$ and $\text{vec}^d(P_k^+)$. Using relation (14), compute $S_k^+$.

5. Initialization of the next step. Set $\hat{q}_k^- = \hat{q}_k^+, \ P_k^- = \text{diag}(S_k^+)^2, \ m_k = \hat{q}_k^-, \ \Sigma_k = P_k^-, \ k = k + 1$.

6. Termination test. If the Stopping rule (see paragraph 2.2.2) is not satisfied, go to step 2, otherwise stop.

The practical implementation of this algorithm requires

- properly initializing the Gaussian distribution, i.e. $m_0$ and $\Sigma_0$.
- selecting the user-defined parameters, namely $N$, $N_\xi$ and $\alpha$.
- introducing a stopping rule.

These various aspects are considered in sections 2.2.1 and 2.2.2.

2.2.1 Initialization and parameter settings

The initial parameters of the Gaussian generator are selected to cover the entire search space. To this end, the following rule can be used:

$$m_0 = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_{n_q} \end{bmatrix}, \Sigma_0 = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_{n_q}^2 \end{bmatrix}, \text{ with: } \begin{cases} 
\mu_i = \bar{x}_i + \bar{x}_i \\
\sigma_i = \frac{\bar{x}_i - \bar{x}_i}{6}, \quad i = 1, \ldots, n_q,
\end{cases} \quad (16)$$

where $\bar{x_i}$ (respectively, $\bar{x_i}$) is the $i^{th}$ upper bound (respectively, lower bound) of the hyperbox search domain. With this rule, 99% of the samples are generated in the interval $\mu_i \pm 3\sigma_i, \ i = 1, \ldots, n_x$. The three following parameters must be set: the number of points
$N$, the number of best candidates $N_{\xi}$, and the coefficient $\alpha$. To facilitate this task, Table 1 summarizes the influence of these parameters on the number of function evaluations (and thus on the CPU time) and on the average error.

Table 1: Effect of HKA parameters ($\nearrow$: increase, $\searrow$: decrease).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$N$ $\nearrow$</th>
<th>$N_{\xi}$ $\nearrow$</th>
<th>$\alpha$ $\nearrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of function evaluations</td>
<td>$\nearrow$</td>
<td>$\nearrow$</td>
<td>$\searrow$</td>
</tr>
<tr>
<td>Average error</td>
<td>$\searrow$</td>
<td>$\searrow$</td>
<td>$\nearrow$</td>
</tr>
</tbody>
</table>

### 2.2.2 Stopping rule

The algorithm stops when a given number of iterations, MaxIter, is reached (MaxIter = 300 in all our experiments) or a given accuracy indicator is obtained. The latter accounts for the dispersion of the $N_{\xi}$ best points. To this end, we consider that no significant improvement can be made when the $N_{\xi}$ best points are in a ball of a given radius $\rho$ ($\rho = 0.005$ in all our experiments). More precisely, the algorithm stops when $\max_{2 \leq i \leq N_{\xi}} \| q_1 - q_i \|_2 \leq \rho$, where $q_i$ ($i = 1, \ldots, N_{\xi}$) are the $N_{\xi}$ best candidates.

In conclusion, the search procedure HKA is articulated around three main components, the pdf function $f_k(q)$ (parametrized by $m_k$ and $\Sigma_k$), the measurement process, and the Kalman estimator. Sampling from the pdf $f_k(q)$ at iteration $k$, creates a collection of vectors $\mathbf{q}(k)$. This collection is then used by the measurement process to provide information on the optimum. Via the Kalman estimator, this information is then combined with the pdf $f_k(q)$ in order to produce a new pdf $f_{k+1}(q)$ which will be used in the next iteration. Since the estimation process is carried out to minimize the variance of the posterior estimation error, it follows that $\Sigma_k$ tends toward zero after a sufficient number of iterations. The resulting $m_k$ can then be considered as a reliable estimate of the optimum. More precisely, an interesting property of this algorithm is its almost sure convergence to a near-optimal solution. This property is established in the next section.
2.3 Convergence properties

Let \( J(q) : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( \mathcal{D} \subset \mathbb{R}^n \) be the cost function and the set of admissible solutions, respectively. A vector \( q_{\text{opt}} \in \mathbb{R}^n \) is a global minimizer on \( \mathcal{D} \), if:

\[
q_{\text{opt}} \in \mathcal{D}, \quad \text{and} \quad J(q_{\text{opt}}) \leq J(q), \quad \forall q \in \mathcal{D}.
\]  

(17)

By definition, a set of vectors \( q \in \mathcal{D} \) that belong to a neighborhood of the global minimum is a set of approximate solutions. Given \( \varepsilon > 0 \), the set of \( \varepsilon \)-approximate solutions of \( J(q_{\text{opt}}) \) is defined as:

\[
\mathcal{E} = \{ q \in \mathcal{D} : |J(q_{\text{opt}}) - J(q)| \leq \varepsilon \}
\]

(18)

Thus any value \( \hat{J}_{\text{opt}} = J(q) \) with \( q \in \mathcal{E} \) is an estimate of the optimum \( J(q_{\text{opt}}) \) with a precision level \( \varepsilon \), and every \( q \in \mathcal{E} \) is a near-optimal solution.

In what follows, \( f_k(q) \) represents the probability density function (pdf) of the random generator at iteration \( k \). We denote by \( \eta(k) = |q(k) \cap \mathcal{E}| \) the cardinal number of the finite discrete set \( q(k) \cap \mathcal{E} \), i.e. the number of elements at iteration \( k \). The proposition given hereafter gives the sufficient conditions for convergence of HKA to a near-optimal solution.

**Proposition.** If the following two conditions are satisfied:

1. \( \mathcal{E} \) is a convex set,

2. \( f_k(q) > 0 \) for all \( q \in \mathcal{D} \) and \( k > 0 \),

then the HKA converges almost surely to a near-optimal solution.

**Proof.** To show the almost sure convergence, it is sufficient to show that the probability of drawing at least \( N_\xi \) samples in \( \mathcal{E} \) tends to unity as the number of iterations increases. For iteration \( k \), the probability of drawing at least \( N_\xi \) samples in \( \mathcal{E} \) is given by the binomial law

\[
\Pr \{ \eta(k) \geq N_\xi \} = z_k = \sum_{i=N_\xi}^{N} \frac{N!}{i!(N - i)!} p_k^i (1 - p_k)^{N-i},
\]

(19)
where $p_k$ is the probability that $x \in D$ belongs to $\mathcal{E}$. This probability is given by

$$p_k = \int_E f_k(q) dq. \quad (20)$$

Condition 2 implies that $p_k > 0$ for all $k$, and thus $z_k$ is also a non-zero probability for all $k$. Therefore, a worst-case probability $z_{wc}$ exists, which can be defined as follows:

$$0 < z_{wc} \leq z_k \quad \forall k. \quad (21)$$

Since $\mathcal{E}$ is convex, $\Pr \{ \eta(k) \geq N_\xi \}$ is also the probability that $\xi_k$ belongs to $\mathcal{E}$. Consider the sequence $\{\xi^{best}_k\}$ defined as

$$\xi^{best}_{k+1} = \xi_{k+1} \text{ if } J(\xi_{k+1}) < J(\xi_k)$$
$$\xi^{best}_{k+1} = \xi^{best}_k \text{ if } J(\xi_{k+1}) \geq J(\xi_k). \quad (22)$$

The almost sure convergence means that $\lim_{k \to \infty} \Pr \{ \xi^{best}_k \in \mathcal{E} \} = 1$. Then, we must show that the sequence $\{\xi^{best}_k\}$ obtained by HKA converges with a probability of one to $\mathcal{E}$. To this end, consider the random variable $y_i, i = 2, \cdots, k$, defined as

$$y_i = 1, \text{ if } J(\xi^{best}_i) \leq J(\xi^{best}_{i-1}) - \varepsilon$$
$$y_i = 0, \text{ if } J(\xi^{best}_i) > J(\xi^{best}_{i-1}) - \varepsilon \quad (23)$$

In other words, $y_i = 1$ means a success in our attempt to reduce the cost function $J$ by at least $\varepsilon$. Using the value of $J$ for the initial measurement (i.e. $\xi_1$), we introduce the following variable:

$$N_s = \left\lfloor \frac{J(\xi_1) - J(q_{opt})}{\varepsilon} \right\rfloor, \quad (24)$$

where $[v]$ is the largest integer smaller than $v$. Therefore, a sufficient condition so that $\xi^{best}_k$ belongs to $\mathcal{E}$ is

$$\sum_{i=2}^{k} y_i \geq N_s + 1. \quad (25)$$

The probability $\Pr \{ \xi^{best}_k \notin \mathcal{E} \}$ is less than or equal to the probability that the number of successful steps does not exceed $N_s$:

$$\Pr \{ \xi^{best}_k \notin \mathcal{E} \} \leq \Pr \left\{ \sum_{i=2}^{k} y_i \leq N_s \right\}. \quad (26)$$
The latter probability increases with a decrease in the probability of successful steps. From (21), we know that the probability of any successful step is greater than or equal to the worst case probability $z_{wc}$. Consequently, we can write:

$$\Pr \left\{ \sum_{i=2}^{k} y_i \leq N_s \right\} \leq \sum_{i=0}^{N_s} \frac{k!}{i!(k-i)!} z_{wc}^i (1 - z_{wc})^{k-i},$$

this latter expression can be upper-bounded as follows (see [20]):

$$\sum_{i=0}^{N_s} \frac{k!}{i!(k-i)!} z_{wc}^i (1 - z_{wc})^{k-i} \leq \frac{N_s + 1}{N_s!} k^{N_s} (1 - z_{wc})^k.$$

Therefore, from (26), we have

$$\Pr \{ \xi_k^{best} \notin \mathcal{E} \} \leq \frac{N_s + 1}{N_s!} k^{N_s} (1 - z_{wc})^k.$$

Since $z_{wc} > 0$, it is clear that

$$\lim_{k \to \infty} k^{N_s} (1 - z_{wc})^k = 0,$$

thus

$$\lim_{k \to \infty} \Pr \{ \xi_k^{best} \in \mathcal{E} \} = 1.$$ This shows the almost sure convergence of HKA.

3 Numerical experiments

In this section, the ability of the presented method to solve a wide range of non-convex optimization problems is tested on various numerical examples, both in the unconstrained and constrained cases. In all cases, our results were compared to those obtained via other metaheuristics. We did not program the corresponding algorithms, but only used the available published results. Details on these metaheuristics are given in the cited literature. We considered the unconstrained and constrained cases separately, because specific methods have been proposed to handle constraints (in particular the notion of co-evolution or the introduction of an augmented Lagrangian). The same HKA algorithm was used in both cases. The constraints were handled merely by introducing an augmented cost function using penalty functions. The various experiments were performed using a 1.2 Ghz Celeron personal computer.
3.1 Unconstrained case

HKA was compared to other metaheuristics such as ACO\textsubscript{R}, CGA, ECTS, ESA and INTEROPT, which are listed in Tab. 2. The efficiency of HKA was tested using a set of well-known test functions (RC, B2, DJ, S\textsubscript{4,5}, S\textsubscript{4,7}, S\textsubscript{4,10}, and H\textsubscript{6,4}), which are listed in the Appendix. For these experiments, we performed each test 100 times and we compared our results with those that have been previously published. In all these experiments, the following parameters were used: number of points $N = 25$, number of best candidates $N_\xi = 5$, slowdown coefficient $\alpha = 0.9$. The experimental results are presented in Tab. 3. For each test function, we give the success ratio for 100 runs and the corresponding average number of function evaluations. It can be seen that some results are not available for ECTS, ESA and INTEROPT (as indicated by the symbol “-“). As shown in Tab. 3, the best results were obtained for ACO\textsubscript{R}, CGA, and HKA. The number of evaluations produced by HKA was slightly greater than those produced by CGA and ACO\textsubscript{R}, but its success ratio was better. Tab. 4 presents the results on average error. These results were

Table 2: List of the methods used in our comparisons.

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ant colony optimization for continuous domains (ACO\textsubscript{R})</td>
<td>K. Socha and M. Dorigo [26]</td>
</tr>
<tr>
<td>Continuous Genetic Algorithm (CGA)</td>
<td>R. Chelouah and P. Siarry [4]</td>
</tr>
<tr>
<td>Enhanced Continuous Tabu Search (ECTS)</td>
<td>R. Chelouah and P. Siarry [3]</td>
</tr>
<tr>
<td>Enhanced Simulated Annealing (ESA)</td>
<td>P. Siarry and al. [25]</td>
</tr>
<tr>
<td>INTEROPT</td>
<td>G. L. Bilbro and W.E. Snyder [1]</td>
</tr>
</tbody>
</table>

| Table 3: Comparison of HKA with ACO\textsubscript{R}, CGA, ECTS, ESA and INTEROPT. |
|---------------------------------------------|---------------------------------------------|
|                                      | Success Ratio (%) | Average number of function evaluations |
| ACO\textsubscript{R} | CGA | ECTS | ESA | INTEROPT | HKA | ACO\textsubscript{R} | CGA | ECTS | ESA | INTEROPT | HKA |
|---------------------------------------------|---------------------------------------------|
| RC                                         | 100 | 100 | 100 | 100 | 100 | 857 | 620 | 245 | 4172 | 625 |
| B2                                         | 100 | 100 |    |    |    | 559 | 430 |    |      | 1275 |
| DJ                                         | 100 | 100 |    |    |    | 392 | 750 |    |      | 600  |
| S\textsubscript{4,5}                       | 57  | 76  | 75  | 54  | 40  | 793 | 610 | 825 | 3700 | 675  |
| S\textsubscript{4,7}                       | 79  | 83  | 80  | 54  | 60  | 748 | 680 | 910 | 1223 | 2426 |
| S\textsubscript{4,10}                      | 81  | 81  | 75  | 50  | 50  | 715 | 650 | 898 | 1189 | 3463 |
| H\textsubscript{6,4}                       | 100 | 100 | 100 | 100 | 97  | 722 | 976 | 1520| 2638 | 17262|
| Mean Values                                | 89  | 92  | 86  | 65  | 70  | 684 | 674 | 880 | 1547 | 6205 |
| Mean Values                                |     |     |     |     |     | 745 |     |     |      |      |
not available for ACOR and INTEROPT. Compared to CGA, ECTS and ESA, HKA led to a lower average error.

<table>
<thead>
<tr>
<th>Table 4: Average error.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average Error</strong></td>
</tr>
<tr>
<td>ACO\textsubscript{R}</td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>RC</td>
</tr>
<tr>
<td>B2</td>
</tr>
<tr>
<td>DJ</td>
</tr>
<tr>
<td>S\textsubscript{45}</td>
</tr>
<tr>
<td>S\textsubscript{47}</td>
</tr>
<tr>
<td>S\textsubscript{4,10}</td>
</tr>
<tr>
<td>H\textsubscript{6,4}</td>
</tr>
<tr>
<td>Mean Values</td>
</tr>
</tbody>
</table>

3.2 Constrained case

HKA was compared to other metaheuristics specifically designed for solving constrained problems. Two examples were considered, the welded beam design problem and the robust PID design problem. In both experiments, HKA handled constraints via a new objective function which includes penalty functions:

\[ J_{\text{new}}(x) = J(x) + w \sum_{i=1}^{N_c} \max(g_i(x), 0), \]  

where \( N_C \) is the number of constraints, \( g_i(x) \) is the \( i^{th} \) inequality constraint of the form \( g_i(x) \leq 0 \), and \( w \) is a weighting factor, which was set to 100 in both experiments.

The welded beam design problem. A welded beam is designed for minimum cost subject to constraints on shear stress \( \tau(x) \), bending stress in the beam \( \sigma(x) \), buckling load on the bar \( P_c \), end deflection of the beam \( \delta(x) \), and side constraints [15]. There are four design variables as shown in Fig. 3: \( h(x_1), l(x_2), t(x_3) \) and \( b(x_4) \), \( x = [x_1 \ x_2 \ x_3 \ x_4]^T \).
Figure 3: Welded beam design problem.

The problem can be mathematically formulated as follows:

Minimize \[ J(x) = (1 + c_1)x_1^2 x_2 + c_2 x_3 x_4 (L + x_2) \]

Subject to:

\[ g_1(x) = \tau(x) - \tau_{\text{max}} \leq 0 \]
\[ g_2(x) = \sigma(x) - \sigma_{\text{max}} \leq 0 \]
\[ g_3(x) = x_1 - x_4 \leq 0 \]
\[ g_4(x) = c_1 x_1 + c_2 x_3 x_4 (L + x_2) - 5 \leq 0 \]
\[ g_5(x) = h_{\text{min}} - x_1 \leq 0 \]
\[ g_6(x) = \delta(x) - \delta_{\text{max}} \leq 0 \]
\[ g_7(x) = P - P_c(x) \leq 0 \]

where

\[ \tau(x) = \sqrt{\tau_1^2 + 2 \tau_1 \tau_2 \frac{x_2}{2R} + \tau_2^2}, \quad \tau_1 = \frac{P}{\sqrt{2 x_1 x_2}}, \quad \tau_2 = \frac{MR}{T} \]
\[ M = P \left( L + \frac{x_2}{2} \right), \quad R = \sqrt{\frac{x_2}{4} + \left( \frac{x_1 + x_3}{2} \right)^2}, \quad I = 2 \left\{ \sqrt{2 x_1 x_2} \left[ \frac{x_2^2}{12} + \left( \frac{x_1 + x_3}{2} \right)^2 \right] \right\} \]
\[ \sigma(x) = \frac{6PL}{x_4 x_3^2}, \quad \delta(x) = \frac{4PL^3}{Ex_3^2 x_4}, \quad P_c(x) = \frac{4.013E \sqrt{x_3^2 x_4^2 / 36}}{L^2} \left( 1 - \frac{x_3}{2L \sqrt{4G}} \right) \]

\[ (32) \]

\[ (33) \]
and
\[ c_1 = 0.10471, \quad c_2 = 0.04811, \quad P = 6 \times 10^3, \quad L = 14, \quad E = 3 \times 10^7 \]
\[ G = 1.2 \times 10^7, \quad h_{\text{min}} = 0.125, \quad \delta_{\text{max}} = 0.25, \quad \tau_{\text{max}} = 1.36 \times 10^4, \quad \sigma_{\text{max}} = 3 \times 10^4. \]

The ranges of design variables are

\[ 0.1 \leq x_1 \leq 2, \quad 0.1 \leq x_2 \leq 10, \quad 0.1 \leq x_3 \leq 10, \quad 0.1 \leq x_4 \leq 2. \]

This problem has been solved using a genetic algorithm (GA) with binary representation and a traditional penalty function [8]. It has also been solved using geometric programming (GP) [23]. Recently, this problem was also solved using a GA-based co-evolution model [5] as well as a multi-objective genetic algorithm (MGA) [6, 7]. Even more recently, this problem was solved using a co-evolutionary particle swarm optimization (CPSO), with a better solution than those previously obtained [15].

In this experiment, we performed the minimization problem 30 times and we compared our results with those obtained via the methods listed in TAB. 5. The following parameters were used: number of points \( N = 50 \), number of best candidates \( N_\xi = 5 \), slowdown coefficient \( \alpha = 0.3 \).

<table>
<thead>
<tr>
<th>Method</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometric Programming (GP)</td>
<td>K.M. Ragsdell and D.T. Phillips [23]</td>
</tr>
<tr>
<td>Genetic Algorithm and Penalty function (GAP)</td>
<td>K. Deb [8]</td>
</tr>
<tr>
<td>Co-Evolutionary Genetic Algorithm (CEGA)</td>
<td>C.A.C. Coello [5, 6]</td>
</tr>
<tr>
<td>Multi-objective Genetic Algorithm (MGA)</td>
<td>C.A.C. Coello and E.M. Montes [7]</td>
</tr>
<tr>
<td>Co-evolutionary Particle Swarm Optimization (CPSO)</td>
<td>Q. He and L. Wang [15]</td>
</tr>
</tbody>
</table>

The best solutions obtained by the above-mentioned approaches are listed in TAB. 6 and the statistical results are shown in TAB. 7.
Table 6: Comparison of the best solution found using different methods.

<table>
<thead>
<tr>
<th></th>
<th>GP</th>
<th>GAP</th>
<th>CEGA</th>
<th>MGA</th>
<th>CPSO</th>
<th>HKA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.245500</td>
<td>0.248900</td>
<td>0.208800</td>
<td>0.205986</td>
<td>0.202369</td>
<td>0.205624</td>
</tr>
<tr>
<td>$x_2$</td>
<td>6.196000</td>
<td>6.173000</td>
<td>3.420500</td>
<td>3.471328</td>
<td>3.544214</td>
<td>3.473825</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.245500</td>
<td>0.253300</td>
<td>0.210000</td>
<td>0.206480</td>
<td>0.205723</td>
<td>0.205738</td>
</tr>
<tr>
<td>$g_1$</td>
<td>-5743.826517</td>
<td>-5758.603777</td>
<td>-0.337812</td>
<td>-0.074092</td>
<td>-12.839796</td>
<td>-5.621131</td>
</tr>
<tr>
<td>$g_2$</td>
<td>-4.715097</td>
<td>-255.576901</td>
<td>-353.902604</td>
<td>-0.266227</td>
<td>-1.247467</td>
<td>-14.103308</td>
</tr>
<tr>
<td>$g_3$</td>
<td>0.000000</td>
<td>-0.004400</td>
<td>-0.001200</td>
<td>-0.000495</td>
<td>-0.001498</td>
<td>-0.000115</td>
</tr>
<tr>
<td>$g_4$</td>
<td>-3.020289</td>
<td>-2.982866</td>
<td>-3.411865</td>
<td>-3.430043</td>
<td>-3.429347</td>
<td>-3.432290</td>
</tr>
<tr>
<td>$g_5$</td>
<td>-0.120500</td>
<td>-0.123900</td>
<td>-0.083800</td>
<td>-0.080986</td>
<td>-0.079381</td>
<td>-0.080624</td>
</tr>
<tr>
<td>$g_6$</td>
<td>-0.234208</td>
<td>-0.234160</td>
<td>-0.235649</td>
<td>-0.235514</td>
<td>-0.235536</td>
<td>-0.235550</td>
</tr>
<tr>
<td>$g_7$</td>
<td>-3604.275002</td>
<td>-4465.270928</td>
<td>-363.232384</td>
<td>-58.666440</td>
<td>-11.681355</td>
<td>-1.595159</td>
</tr>
<tr>
<td>$J(x)$</td>
<td>2.385937</td>
<td>2.433116</td>
<td>1.748309</td>
<td>1.728226</td>
<td>1.728024</td>
<td>1.725539</td>
</tr>
</tbody>
</table>

Table 7: Statistical results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std Dev</th>
<th>Average number of function evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>2.385937</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GAP</td>
<td>2.433116</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CEGA</td>
<td>1.748309</td>
<td>1.771973</td>
<td>1.785835</td>
<td>0.011220</td>
<td>900000</td>
</tr>
<tr>
<td>MGA</td>
<td>1.728226</td>
<td>1.792654</td>
<td>1.993408</td>
<td>0.074713</td>
<td>80000</td>
</tr>
<tr>
<td>CPSO</td>
<td>1.728024</td>
<td>1.748831</td>
<td>1.782143</td>
<td>0.012926</td>
<td>200000</td>
</tr>
<tr>
<td>HKA</td>
<td>1.725539</td>
<td>1.725824</td>
<td>1.726287</td>
<td>0.000172</td>
<td>186000</td>
</tr>
</tbody>
</table>

As shown in Tab. 6, the best feasible solution found by HKA was better than the best solutions found by other techniques. As shown in Tab. 7, the average searching quality of HKA was also significantly better than those of other methods, and even the worst solution found by HKA was better than the best solution found via CPSO method. In addition the standard deviations of the results obtained by HKA were very small. Furthermore, the number of function evaluations was significantly lower than those obtained by the other methods.

Robust PID controller tuning. In a wide range of engineering applications, the problem of designing a robust Proportional-Integral-Derivative (PID) controller remains an open issue. This is mainly due to the fact that the underlying optimization problem is non-convex and thus suffers from computational intractability and conservatism. To overcome
these difficulties, Kim et al. [17] suggested solving this problem by augmented Lagrangian Particle Swarm Optimization (ALPSO). Here we show that HKA is able to solve the problem of designing a robust PID controller. The obtained results were then compared to those of ALPSO. The problem can be mathematically formulated as follows

\[
\text{Minimize} \quad J(x) = \arg \max_{\lambda_i(x)} \{ \text{Re}(\lambda_i(x)), \forall i \}, \quad x = [x_1 \ x_2 \ x_3 \ x_4]^T
\]

\[
\text{Subject to:} \quad g_1(x) = \sup_{\omega \geq 0} \{ |W_S(s)|_{s=j\omega} |S(s,x)|_{s=j\omega} \} - 1 \leq 0
\]

\[
\text{Subject to:} \quad g_2(x) = \sup_{\omega \geq 0} \{ |W_T(s)|_{s=j\omega} |T(s,x)|_{s=j\omega} \} - 1 \leq 0
\]

\[x_i \leq x_i \leq \bar{x}_i, \quad i = 1, \cdots, 4\]  \hspace{1cm} (35)

where \(x = [x_1 \ x_2 \ x_3 \ x_4]^T\) is the vector of decision variables, \(x_i\) and \(\bar{x}_i\) are the bounds of the hyperbox search domain, \(s\) is the Laplace variable, \(\omega\) is the frequency (rad/s), \(j\) is the unit imaginary number, \(S(s,x)\) is the sensitivity function defined as \(S(s,x) = 1/(1 + L(s,x))\), \(T(s,x)\) is the closed-loop system defined as \(T(s,x) = L(s,x)/(1 + L(s,x))\), \(L(s,x)\) is the open-loop transfer function defined as \(L(s,x) = P(s)K(s,x)\) where \(P(s)\) is the transfer function of the system to be controlled, and \(K(s,x)\) the transfer function of the PID controller

\[
K(s,x) = 10^{x_1} \left(1 + \frac{1}{10^{x_2} s} + \frac{10^{x_3} s}{1 + 10^{(x_3-x_4)} s}\right) \hspace{1cm} (36)
\]

In the objective function, \(\lambda_i(x)\) denotes the \(i^{th}\) pole of the closed-loop system. The frequency-dependent weighting functions \(W_S(s)\) and \(W_T(s)\) are set so as to meet the performance specifications of the closed-loop system.

As in [17], we solved this optimization problem for the magnetic levitation system described in [27]. The process model is defined as

\[
P(s) = \frac{7.147}{(s - 22.55)(s + 20.9)(s + 13.99)} \hspace{1cm} (37)
\]

The frequency-dependent weighting functions \(W_S(s)\) and \(W_T(s)\) are respectively given as

\[
W_S(s) = \frac{5}{s + 0.1}, \quad W_T(s) = \frac{43.867(s + 0.066)(s + 31.4)(s + 88)}{(s + 10^4)^2} \hspace{1cm} (38)
\]
The search space is

\[ 2 \leq x_1 \leq 4, \quad -1 \leq x_2 \leq 1, \quad -1 \leq x_3 \leq 1, \quad 1 \leq x_4 \leq 3. \]

In this test, we performed minimization 30 times and we compared our results with those presented in [17]. The following parameters were used: number of points \( N = 50 \), number of best candidates \( N_{\xi} = 5 \), slowdown coefficient \( \alpha = 0.4 \).

The best solutions obtained via ALPSO and HKA are listed in Tab. 8 and the statistical results are shown in Tab. 9 (the statistical results were not available for ALPSO).

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( g_1(x) )</th>
<th>( g_2(x) )</th>
<th>( J(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPSO</td>
<td>3.2548</td>
<td>-0.8424</td>
<td>-0.7501</td>
<td>2.3137</td>
<td>6.1e-3</td>
<td>-4.0e-4</td>
</tr>
<tr>
<td>HKA</td>
<td>3.2542</td>
<td>-0.8634</td>
<td>-0.7493</td>
<td>2.3139</td>
<td>-9.6e-4</td>
<td>-1.2e-3</td>
</tr>
</tbody>
</table>

Table 8: Comparison of the best solutions found via ALPSO and HKA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std Dev</th>
<th>CPU time</th>
<th>Average number of function evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPSO</td>
<td>-1.7197</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>687 s</td>
<td>25000</td>
</tr>
<tr>
<td>HKA</td>
<td>-1.7106</td>
<td>-1.7023</td>
<td>-1.6891</td>
<td>0.0048</td>
<td>266 s</td>
<td>5427</td>
</tr>
</tbody>
</table>

Table 9: Statistical results.

The higher absolute value of the objective function obtained using ALPSO is due to the violation of constraint \( g_1(x) \); this is not the case for our solution in which all constraints are satisfied. As shown in Tab. 9, HKA required a smaller number of function evaluations and had a lower associated CPU time compared to ALPSO.
If, as in ALPSO, a small violation of the constraint $g_1(x)$ is tolerated, we obtained the results listed in Tab. 10 and Tab. 11.

Table 10: Comparison of the best solutions found via ALPSO and HKA.

<table>
<thead>
<tr>
<th></th>
<th>ALPSO</th>
<th>HKA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>3.2548</td>
<td>3.2556</td>
</tr>
<tr>
<td>$x_2$</td>
<td>-0.8424</td>
<td>-0.8354</td>
</tr>
<tr>
<td>$x_3$</td>
<td>-0.7501</td>
<td>-0.7539</td>
</tr>
<tr>
<td>$x_4$</td>
<td>2.3137</td>
<td>2.3127</td>
</tr>
<tr>
<td>$g_1(x)$</td>
<td>6.1e-3</td>
<td>4.9e-3</td>
</tr>
<tr>
<td>$g_2(x)$</td>
<td>-4.0e-4</td>
<td>-2.8e-3</td>
</tr>
<tr>
<td>$J(x)$</td>
<td>-1.7197</td>
<td>-1.7435</td>
</tr>
</tbody>
</table>

Table 11: Statistical results.

<table>
<thead>
<tr>
<th>Method</th>
<th>Best</th>
<th>Mean</th>
<th>Worst</th>
<th>Std Dev</th>
<th>CPU time</th>
<th>Average number of function evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPSO</td>
<td>-1.7197</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>687 s</td>
<td>25000</td>
</tr>
<tr>
<td>HKA</td>
<td>-1.7435</td>
<td>-1.7381</td>
<td>-1.7323</td>
<td>0.0030</td>
<td>248 s</td>
<td>5072</td>
</tr>
</tbody>
</table>

The best solution found by HKA was better than the solution found by ALPSO with, in addition, a smaller violation constraint (Tab. 10). Tab. 11 shows that the worst solution found by HKA was better than the solution found via ALPSO; in addition, HKA had a smaller number of function evaluations (and thus a lower corresponding CPU time) compared to ALPSO.

4 Conclusion

In this paper, a new optimization algorithm, called the heuristic Kalman algorithm (HKA), was presented. The main characteristic of HKA is to explore the search space via a Gaussian pdf. This exploration is directed by appropriately adjusting the pdf parameters so that they converge to a near-optimal solution with low variance. To this end, a measurement process followed by a Kalman estimator is introduced. The role of the Kalman estimator is to combine the prior pdf function with the measurement process to give a new pdf function.
for the exploration of the search space. We demonstrated that under not too restrictive conditions, the HKA algorithm converges “almost surely” to a near-optimal solution.

We tested the performance of HKA using several non-convex test problems, both in the unconstrained and constrained cases. The results obtained show that HKA can be considered as a good alternative for solving difficult non-convex problems quickly and with a high probability of success.

### Appendix

**Test functions RC, B2, DJ, S_{4,5}, S_{4,7}, S_{4,10}, and H_{6,4}**

- **Branin’s function** (RC) (2 variables)

  \[ J(x) = \left( x_2 - \left( \frac{5.1}{4\pi^2} \right) x_1^2 + \left( \frac{5}{\pi} \right) x_1 - 6 \right)^2 + 10 \left( 1 - \left( \frac{1}{8\pi} \right) \right) \cos(x_1) + 10 \]

  search domain: \(-5 \leq x_i \leq 10, i = 1, 2\); 3 global minima: \(x_{opt} = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)\); \(J(x_{opt}) = 0.397887\).

- **Bohachecksky’s function** (B2) (2 variables)

  \[ J(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7; \]

  search domain: \(-100 \leq x_i \leq 100, i = 1, 2\); global minimum: \(x_{opt} = (0, 0)\), \(J(x_{opt}) = 0\).

- **De Jong’s function** (DJ) (3 variables); \(J(x) = x_1^2 + x_2^2 + x_3^2\); search domain: \(-5 \leq x_i \leq 5, i = 1, 3\); global minimum: \(x_{opt} = (0, 0, 0)\), \(J(x_{opt}) = 0\).

- **Shekel’s functions** (S_{4,5}, S_{4,7}, S_{4,10}) (4 variables); search domain: \(0 \leq x_i \leq 9, i = 1, 4\); 3 functions were considered S_{4,5}, S_{4,7} and S_{4,10}. For a complete definition of these functions see [26].

- **Hartmann’s functions** (H_{6,4}) (6 variables); search domain: \(0 \leq x_i \leq 1, i = 1, 6\); global minimum: \(J(x_{opt}) = -3.322368\). For a complete definition of this function, see [26].
References


