**6th World Congresses of Structural and Multidisciplinary Optimization**
Rio de Janeiro, 30 May - 03 June 2005, Brazil

# Constrained Particle Swarm Optimization of Mechanical Systems

**Kai Sedlaczek and Peter Eberhard**

Institute B of Mechanics, University of Stuttgart
Pfaffenwaldring 9, 70569 Stuttgart, Germany
[sedlaczek,eberhard]@mechb.uni-stuttgart.de
www.mechb.uni-stuttgart.de

**1. Abstract**
Using Particle Swarm Optimization (PSO) for solving nonlinear, multimodal and non-differentiable optimization problems has gained increasing attention in recent years. Based on the social behavior and interaction of swarms like bird flocks, the method of Particle Swarm Optimization was introduced in 1995. While the behavior of the PSO algorithm is partially similar to other well known stochastic optimization methods such as Evolutionary Strategies or Simulated Annealing, it convinces by its simple structure characterized by only a few lines of computer code.

However, engineering optimization tasks often require optimization methods capable of handling problem immanent equality and inequality constraints. This work utilizes the simple structure of the basic PSO technique and combines this method with an extended non-stationary penalty function approach, called Augmented Lagrange Multiplier Method, where ill-conditioning is a far less harmful problem and the correct solution can be obtained even for finite penalty factors. We describe the basic PSO algorithm, its relation to Evolutionary Strategies and the resulting method for constrained problems including results from benchmark tests. The applicability of the Augmented Lagrange Particle Swarm Optimization is shown by a demanding mechanical engineering example, where we optimize the stiffness of an industrial hexapod robot with parallel kinematics.
**2. Keywords:** Particle Swarm Optimization, Nonlinear Constraints, Augmented Lagrange Multiplier Method, Multibody Systems, Parallel Kinematics, Optimization.

## 3. Introduction

The general nonlinear optimization problem is given by the nonlinear objective function $f$, which is to be minimized with respect to the design variables $\boldsymbol{x}$ and the nonlinear equality and inequality constraints. This can be formulated by

$$\underset{\boldsymbol{x}}{\text{minimize}} \;\; f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{D} \cap \mathbb{F}, \; \mathbb{D} \subseteq \mathbb{R}^n, \tag{1}$$

subject to the nonlinear equality and inequality constraints

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}, \quad \boldsymbol{g} : \mathbb{R}^n \rightarrow \mathbb{R}^{m_e}, \tag{2}$$

$$\boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}, \quad \boldsymbol{h} : \mathbb{R}^n \rightarrow \mathbb{R}^{m_i}, \tag{3}$$

which define the feasible region $\mathbb{F}$ and the search space $\mathbb{D}$ is additionally bounded by the simple bounds $\boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u$.

Most solution methods and algorithms exploit properties such as linearity, differentiability, convexity, separability or non-existence of constraints in order to solve the problem efficiently. However, the general nonlinear problem might not fulfill any of these requirements for the use of efficient solution methods. Hence, stochastic methods such as Evolutionary Algorithms, Simulated Annealing or Particle Swarm Optimization might be applied to the optimization problem.

The Particle Swarm Optimization method was first introduced by Kennedy and Eberhart [10] in 1995. It is based on the observation, interpretation and simulation of both the movement of individuals of bird flocks or fish schools as well as their collective behavior as a swarm. Only a few lines of computer code based on simple mathematical operations are required to implement the basic concept of Particle Swarm Optimization. It utilizes the swarm intelligence by simulating the social interaction of the particles to find the best place in the search space. Particle Swarm Optimization is not a deterministic approach since the social interaction depends on the individual social behavior assigned stochastically to each particle.

The reason for the attraction of PSO is certainly based on its simplicity. The few simple lines required to describe the basic algorithm and the derivative free search for the solution make PSO an easy-to-use algorithm for real life problems. Like well known stochastic optimization methods such as Evolutionary Strategies or Simulated Annealing, PSO is not restricted to a local solution of the optimization problem. Thus, the solution does hardly depend on an initial starting point which can be of great advantage in the optimization process searching for novel designs.

PSO has received a lot of attention in recent years. Many publications presenting PSO variants and applications as well as basic analyses can be found in literature. As mentioned before, the basic algorithm was introduced by Kennedy and Eberhart in [10], followed by a more general work on swarm intelligence [11]. An in-depth study of the basic PSO method and some modified versions and extensions including convergence analysis is presented by Bergh in his dissertation [1]. Many variants were developed tackling different problem formulations and characteristics. Venter and Sobieszczanski-Sobieski [16] as well as Laskari et al. [12] presented some modifications towards integer programming. Clerc [2] modified the PSO algorithm in order to solve combinatorial problems like the Traveling Salesman Problem. He also developed an adaptive, parameter free version of PSO [3]. Coello and Lechuga [4] proposed a PSO algorithm to solve multi-objective optimization problems by computing the Pareto-optimal front. Some examples for engineering applications using Particle Swarm Optimization can be found in Hu et al. [8], in Shutte and Groenwold [14] or in Sedlaczek and Eberhard [15].

Mechanical engineering problems, however, often require optimization methods which are capable of handling the problem immanent equality and inequality constraints. Gradient-based methods usually consider constraints efficiently by the use of Lagrange multipliers, but stochastic methods are often unable to solve constrained problems in a reasonable way. Some methods make use of penalty or barrier functions in order to reduce the constrained problem to an unconstrained problem by penalizing the objective function resulting in an unconstrained pseudo objective function

$$\psi(\boldsymbol{x}) = f(\boldsymbol{x}) + r_p\, \phi(\boldsymbol{x}), \tag{4}$$

where $\phi(\boldsymbol{x})$ is the penalty function that contains the violated constraints and $r_p$ is termed the penalty factor. The most common and simple approach for the penalty function is the use of a quadratic penalization

$$\phi(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) \cdot \boldsymbol{g}(\boldsymbol{x}), \tag{5}$$

where we consider here for simplicity only equality constraints. However, this strategy is often not advisable, since the solution of the original problem is not equal to the solution of the unconstrained pseudo objective function. Consider the following example for demonstration

$$\underset{x \in I\!R^1}{\text{minimize}} \quad f(x) = x^2 \tag{6}$$

$$\text{subject to} \quad g(x) = x - 1 = 0 \tag{7}$$

with the solution of the resulting unconstrained pseudo problem according to Eq.(4) and Eq.(5)

$$x^*(r_p) = \frac{r_p}{r_p + 1}. \tag{8}$$

It is evident that the exact solution can only be obtained with an infinite penalty factor $r_P \to \infty$ yielding an ill-conditioned problem formulation.

This paper presents an advanced PSO technique which combines the basic algorithm with an Augmented Lagrange Multiplier Method [5], which corresponds to an extended non-stationary penalty function approach. There, ill-conditioning is a far less harmful problem and the correct solution can be obtained even for finite penalty factors. The simple structure of Particle Swarm Optimization allows an efficient implementation of the Augmented Lagrange Multiplier Method which results in a robust optimization algorithm. The dynamic updates of penalty factors as well as Lagrange multiplier estimates limit the computational overhead of the resulting nested loop characteristic. Hence, it is possible to benefit from the increasing computing power available nowadays and to apply Particle Swarm Optimization to engineering problems. This is especially true for optimization problems without known initial feasible

design or without available derivatives, which might be due to non-differentiable objective functions or due to expensive derivative calculations. The computational burden due to the large number of required function evaluations is a remaining disadvantage not only of PSO but of all stochastic algorithms. The increasing complexity of the problem formulation often restricts the use of stochastic methods despite increasing computing power. However, the population based structure of Particle Swarm Optimization allows taking full advantage of using parallel computers. The applicability of the presented Augmented Lagrange Particle Swarm Optimization method is not only shown by solving several benchmark problems but also by a demanding mechanical engineering example optimizing the stiffness of an industrial hexapod robot with parallel kinematics.

## 5. The Basic PSO Algorithm

Particle Swarm Optimization is a stochastic optimization method based on the simulation of the social behavior of bird flocks or fish schools. Therefore, it is a population based algorithm similar to Evolutionary Algorithms. The algorithm utilizes swarm intelligence in order to find the best place in the search space. The actual position of a particle is associated with a particular design vector $\boldsymbol{x}$. The trajectory of the $i$-th particle at iteration $k$ can be described with the position update equation

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \Delta\boldsymbol{x}_i^{k+1} \tag{9}$$

and the so called velocity update equation

$$\Delta\boldsymbol{x}_i^{k+1} = w\Delta\boldsymbol{x}_i^k + c_1 r_{1,i}^k(\boldsymbol{x}_i^{best,k} - \boldsymbol{x}_i^k) + c_2 r_{2,i}^k(\boldsymbol{x}_{swarm}^{best,k} - \boldsymbol{x}_i^k), \tag{10}$$

where the position change $\Delta\boldsymbol{x}_i^{k+1}$ is often referred to as "velocity", which is physically not correct, but the typical term in literature. Further, $\boldsymbol{x}_i^{best,k}$ is the best previously obtained position of the particle $i$ and $\boldsymbol{x}_{swarm}^{best,k}$ is the best position in the entire swarm at the current iteration $k$. The random numbers $r_{1,i}^k$ and $r_{2,i}^k$ are uniformly distributed in $[0,1]$ and represent the stochastic behavior of the algorithm. The term $c_1 r_{1,i}^k(\boldsymbol{x}_i^{best,k} - \boldsymbol{x}_i^k)$ is associated with cognition, since it takes into account only the best position of the particle's own experience. The term $c_2 r_{2,i}^k(\boldsymbol{x}_{swarm}^{best,k} - \boldsymbol{x}_i^k)$ represents the social interaction of the particles. Hence, $c_1$ and $c_2$ are referred to as cognitive scaling factor and social scaling factor. Together with the inertia factor $w$ they influence the particle trajectories and thus the behavior of the entire swarm. Applying Eq.(9) and Eq.(10) iteratively to a set of $n_p$ particles (swarm) the basic PSO algorithm consists of the following steps:

- **[PSO1]** Initialize $n_p$ particles with predefined or randomly chosen positions and evaluate corresponding objective function values at each position. Set $k = 0$. Determine $\boldsymbol{x}_i^{best,0}$ and $\boldsymbol{x}_{swarm}^{best,0}$.
- **[PSO2]** Check termination criteria. If satisfied, the algorithm terminates with the solution $\boldsymbol{x}^* = \boldsymbol{x}_{swarm}^{best,k}$. If $k > k_{max}$ the algorithm stops with failure.
- **[PSO3]** Apply update equations Eq.(9) and Eq.(10) to all particles and evaluate corresponding objective function values at each position. Set $k = k + 1$. Determine $\boldsymbol{x}_i^{best,k}$ and $\boldsymbol{x}_{swarm}^{best,k}$.
- **[PSO4]** Proceed with step **[PSO2]**.

The inertia factor as well as the cognitive and social scaling factors can be used to control the behavior of the swarm, especially the convergence and search diversity properties of the algorithm. Bergh [1] derived a simple inequality condition for guaranteed convergence that is

$$w > \frac{1}{2}(c_1 + c_2) - 1. \tag{11}$$

Alternatively, the multiplication of Eq.(10) with a constriction factor, proposed by Clerc [3], can be used to guarantee convergence. However, the optimal values of the control parameters regarding convergence rates and diversity are strongly problem dependent. Several authors suggested therefore a dynamic inertia reduction. A linear decrease in the inertia factor $w$ as the search progresses can improve the

convergence rate of the algorithm. But premature convergence to a potentially inferior local solution might be unwanted in solving multimodal optimization problems. The replacement of the term $\boldsymbol{x}_{swarm}^{best,k}$ in Eq.(10) by $\boldsymbol{x}_{nh,i}^{best,k}$, that is the best position in the neighborhood of particle $i$, maintains multiple attractors and improves the diverse search for a global optimum. Since the social interaction of each particle then takes place only in its neighborhood (subpopulation) any information is passed slower through the entire swarm. Also, additional so called craziness can be assigned to each particle in order to improve diversity. Adding a fourth term to Eq.(10), that assigns a random number to the position change, increases the directional diversity in the swarm. Besides the mentioned modifications many PSO variants and extensions have been developed trying to improve the characteristics of PSO. Further details can be found in [1].

The PSO is obviously related to Evolutionary Computation such as Evolutionary Strategies (ES) or Genetic Algorithms (GA). The PSO maintains a population of individuals representing potential solutions. Neglecting the first term, the PSO velocity update equation Eq.(10) can be interpreted as a crossover operator involving the two solutions $\boldsymbol{x}_i^{best,k}$ and $\boldsymbol{x}_{swarm}^{best,k}$ returning a single offspring. Also, social or peer pressure is existent by selecting the best position of the entire swarm or neighborhood and craziness has some similarities to the mutation operator. However, the first term in the velocity update equation maintains some information of previous iterations which makes each iteration not a process of replacing the previous population with a new one but rather a process of adaption. Furthermore, for small steps and an advanced stage of convergence, this can be seen as a coarse representation of gradient information.

Regarding the similarities with Evolutionary Strategies, it is not surprising that the convergence properties are similar. Figure 1 shows the average convergence rates of 30 independent runs of PSO solving two benchmark functions compared with an ES implementation from [6] and in addition with a Simulated Annealing method as described in [9]. The test functions are described in the Appendix.

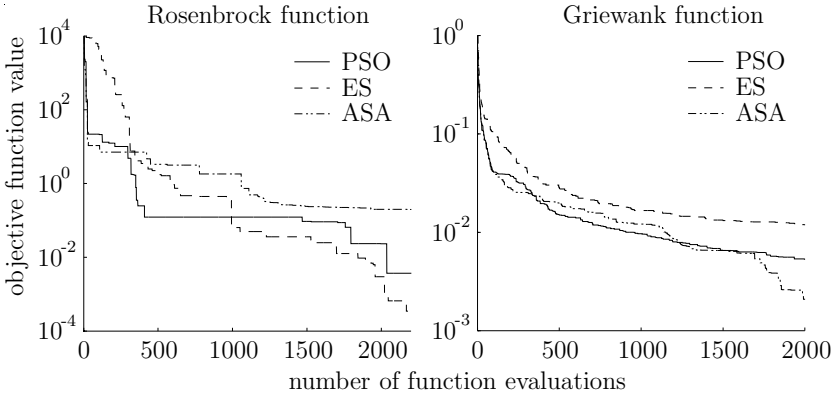


Figure 1: Convergence characteristics of PSO compared to ES and ASA.

**6. Augmented Lagrange Multiplier Method**

Unlike the penalty approach with quadratic penalty functions or barrier functions, the Augmented Lagrange Multiplier Method circumvents the need for infinite penalty factors. The general Lagrange function is given by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}_g) = f(\boldsymbol{x}) + \boldsymbol{\lambda}_g \cdot \boldsymbol{g}(\boldsymbol{x}), \tag{12}$$

where we consider again only equality constraints for simplicity. The solution $\boldsymbol{x}^*$ of the original constrained optimization problem is a stationary point of $L$ for the correct Lagrange multipliers $\boldsymbol{\lambda}_g^*$. This might suggest that Eq.(12) could be used as an unconstrained pseudo function. However, $\boldsymbol{x}^*$ is not necessarily a minimum of $L$. In order to transform $\boldsymbol{x}^*$ from a stationary point into a minimum, the Lagrange function is augmented through the addition of a third term that preserves the stationarity properties at $\boldsymbol{x}^*$, see [5]. We have chosen a quadratic extension which results in the Augmented Lagrange function

$$L_A(\boldsymbol{x}, \boldsymbol{\lambda}_g, \boldsymbol{r}_p) = f(\boldsymbol{x}) + \boldsymbol{\lambda}_g \cdot \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{r}_p \cdot \left[g_1^2(\boldsymbol{x}),\ \ldots\ , g_{m_e}^2(\boldsymbol{x})\right]. \tag{13}$$

In order to be able to penalize each constraint violation separately, we introduced a vector of positive penalty factors $\boldsymbol{r}_p$. Unlike the penalty approach described in Eq.(4) and Eq.(5), it can be shown that there exist finite penalty factors $\boldsymbol{r}_p$ such that $\boldsymbol{x}^*$ is an unconstrained minimum of $L_A(\boldsymbol{x}, \boldsymbol{\lambda}_g^*, \boldsymbol{r}_p)$ and thus of the original constrained problem.

However, the correct Lagrange multipliers $\boldsymbol{\lambda}_g^*$ and the appropriate penalty factors $\boldsymbol{r}_p$ are problem dependent and thus unknown. The solution $\boldsymbol{x}^*$ cannot be directly computed by a single unconstrained minimization of Eq.(13). It rather must be solved a sequence of unconstrained subproblems with subsequent updates of $\boldsymbol{\lambda}_g$ and $\boldsymbol{r}_p$. According to differentiable optimization problems, we apply an update scheme for the Lagrange multipliers based on the solution $\boldsymbol{x}^{*\nu}$ of the stationarity condition of the $\nu$-th subproblem. It holds for $\boldsymbol{x}^\nu \approx \boldsymbol{x}^{*\nu}$ that

$$\left[\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} + \boldsymbol{\lambda}_g^\nu \cdot \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}} + 2\, \boldsymbol{r}_p^\nu \cdot \mathbf{diag}\{g_1(\boldsymbol{x}),\ \ldots\ , g_{m_e}(\boldsymbol{x})\} \cdot \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right]_{\boldsymbol{x}=\boldsymbol{x}^\nu} = \boldsymbol{\epsilon}^\nu \approx \mathbf{0}. \tag{14}$$

Comparing Eq.(14) with the stationarity condition of the Lagrange function Eq.(12), the update scheme for the Lagrange multipliers can be formulated by

$$\boldsymbol{\lambda}_g^{\nu+1} = \boldsymbol{\lambda}_g^\nu + 2\, \boldsymbol{r}_p^\nu \cdot \mathbf{diag}\{g_1(\boldsymbol{x}^\nu),\ \ldots\ , g_{m_e}(\boldsymbol{x}^\nu)\}. \tag{15}$$

If the original optimization problem comprises inequality constraints, we can extend the Augmented Lagrange function to

$$L_A(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{r}_p) = f(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot \boldsymbol{\theta}(\boldsymbol{x}) + \boldsymbol{r}_p \cdot \left[\theta_1^2(\boldsymbol{x}),\ \ldots\ , \theta_{m_e+m_i}^2(\boldsymbol{x})\right] \tag{16}$$

with the penalty function

$$\theta_i = \begin{cases} g_i(\boldsymbol{x}), & i = 1(1)m_e, \\ \max\left[h_{i-m_e}(\boldsymbol{x}), -\dfrac{\lambda_i}{2r_{p,i}}\right], & i = m_e+1(1)m_e{+}m_i. \end{cases} \tag{17}$$

The corresponding update scheme for the Lagrange multipliers may then be defined as

$$\boldsymbol{\lambda}^{\nu+1} = \boldsymbol{\lambda}^\nu + 2\, \boldsymbol{r}_p^\nu \cdot \mathbf{diag}\{\theta_1(\boldsymbol{x}^\nu),\ \ldots\ , \theta_{m_e+m_i}(\boldsymbol{x}^\nu)\}. \tag{18}$$

## 7. Augmented Lagrange Particle Swarm Optimization

The Augmented Lagrange Multiplier Method as described in the previous section can be combined with the Particle Swarm Optimization algorithm. Since the correct Lagrange multipliers $\boldsymbol{\lambda}^*$ and the required magnitude of the penalty factors $\boldsymbol{r}_p$ are unknown, a sequence of unconstrained problems, defined by Eq.(16), must be solved sequentially. This direct approach requires a complete unconstrained minimization with respect to $\boldsymbol{x}$ before performing any update of the Lagrange multipliers and penalty factors. However, if we accept some inaccuracy in the solution of the pseudo objective function Eq.(16), we can obtain improved multiplier estimates and penalty factors more frequently. With the basic PSO algorithm described in Section 5, the Augmented Lagrange Particle Swarm Optimization (ALPSO) comprises the following steps:

**[ALPSO1]** Set $\nu = 0$, $k = 0$, $\boldsymbol{\lambda}^0 = \mathbf{0}$, $\boldsymbol{r}_p^0 = \boldsymbol{r}_{p0}$ and initialize particles with predefined or randomly chosen positions. Evaluate corresponding function values according to Eq.(16) at each position.

**[ALPSO2]** Check termination criteria. If satisfied, the algorithm terminates with the solution $\boldsymbol{x}^* = \boldsymbol{x}^\nu = \boldsymbol{x}_{swarm}^{best,\nu}$, $\boldsymbol{\lambda}^* = \boldsymbol{\lambda}^\nu$. If $\nu > \nu_{max}$ the algorithm stops with failure.

**[ALPSO3]** Solve unconstrained problem Eq.(16) according to steps **[PSO2]** to **[PSO4]** with a limited number of iterations $k_{max}$.

**[ALPSO4]** Update $\boldsymbol{\lambda}$ and $\boldsymbol{r}_p$. Set $\nu = \nu + 1$ and $k = 0$ .

**[ALPSO5]** Proceed with step **[ALPSO2]**.

In order to have not only a stationary point at $\boldsymbol{x}^*$ but a minimum, we apply an heuristic update scheme for the penalty factors $\boldsymbol{r}_p$. If the intermediate solution $\boldsymbol{x}^\nu$ of step [**ALPSO3**] is not closer to the feasible region defined by the $i$-th constraint than the previous solution $\boldsymbol{x}^{\nu-1}$, the penalty factor $r_{p,i}$ is increased. On the other hand, we reduce $r_{p,i}$ if the $i$-th constraint is satisfied with respect to user defined tolerances. This strategy can formulated for equality and inequality constraints by

$$\begin{aligned} r_{p,i}^{\nu+1} &= \begin{cases} 2\, r_{p,i}^{\nu} & \text{if} \quad |g_i(\boldsymbol{x}^\nu)| > |g_i(\boldsymbol{x}^{\nu-1})| \wedge |g_i(\boldsymbol{x}^\nu)| > \epsilon_g, \\ \frac{1}{2}\, r_{p,i}^{\nu} & \text{if} \quad |g_i(\boldsymbol{x}^\nu)| \leq \epsilon_g, \end{cases} \qquad i = 1(1)m_e, \\ r_{p,j}^{\nu+1} &= \begin{cases} 2\, r_{p,j}^{\nu} & \text{if} \quad h_j(\boldsymbol{x}^\nu) > h_j(\boldsymbol{x}^{\nu-1}) \wedge h_j(\boldsymbol{x}^\nu) > \epsilon_h, \\ \frac{1}{2}\, r_{p,j}^{\nu} & \text{if} \quad h_j(\boldsymbol{x}^\nu) \leq \epsilon_h, \end{cases} \qquad j = 1(1)m_i, \end{aligned} \tag{19}$$

where $\epsilon_g$ and $\epsilon_h$ are the user defined tolerances for acceptable constraint violations. Although excessively large penalty factors do not alter the stationarity conditions of Eq.(16), the reduction of corresponding penalty factors is essential for convergent and accurate Lagrange multiplier estimates with a small number of subsequent PSO iterations $k_{max}$. Regarding Eq.(14), it cannot be expected that the result of only $k_{max}$ iterations of step [**ALPSO3**] satisfy the stationarity condition. Hence, it remains a residual $\boldsymbol{\epsilon}^\nu$, whose magnitude is proportional to the penalty factor and which results in a defective Lagrange multiplier estimate. On the other hand, we experienced that a lower bound on the penalty factors yields improved convergence characteristics for the Lagrange multiplier estimates. Regarding the update scheme for the Lagrange multipliers Eq.(18), we maintain the magnitude of the penalty factors such that an effective change in Lagrange multipliers is possible. This lower bound is formulated by

$$r_{p,i} \geq \frac{1}{2}\sqrt{\frac{|\lambda_i|}{\epsilon_{g,h}}}. \tag{20}$$

Table 1 summarizes the experimental results using ALPSO for solving eight constrained benchmark problems. All results show the average values of 30 independent runs on each test function. For comparison, we have chosen problems $P_3$ to $P_8$ according to [13], where an Evolutionary Strategy for solving constrained problems is presented. The dimension of the search space, the number of particles and the maximum number of function evaluations are listed in columns 2 to 4. The known optimal solution $f_{opt}$ is given in column 5. We used cognitive and social scaling factor values of $c_{1,2} = 0.8$ for problem $P_1$ to $P_5$ and $c_{1,2} = 0.4$ for $P_6$ to $P_8$. For all benchmark tests the inertia factor was set to a constant value of $w = 0.9$ while the maximum number of basic PSO iterations was set to $k_{max} = 3$. The constraint tolerances $\epsilon_{g,h} = 10^{-4}$ are used for both equality and inequality constraints in all test runs. All experiments were performed in Matlab.

Table 1: Results of 30 independent runs on 8 benchmark tests using Augmented Lagrange Particle Swarm Optimization. Column 2 shows the number of particles $n_p$ and column 3 the number of function calls $n_f$. Details about the test functions can be found in the Appendix.

| problem | $n$ | $n_p$ | $n_f$ | $f_{opt}$ | $f_{best}$ | $f_{worst}$ | $f_{median}$ | $f_{mean}$ | $f_{std.dev.}$ |
|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 2 | 20 | 30 000 | 13 | 12.9995 | 13.0007 | 13.0000 | 13.0000 | 2.2281E-04 |
| $P_2$ | 2 | 20 | 30 000 | 0.01721 | 0.01719 | 0.01719 | 0.01719 | 0.01719 | 4.2956E-07 |
| $P_3$ | 2 | 20 | 30 000 | -0.09583 | -0.09583 | -0.09583 | -0.09583 | -0.09583 | 1.1132E-11 |
| $P_4$ | 2 | 20 | 30 000 | -6961.81 | -6963.57 | -6958.76 | -6961.84 | -6961.81 | 8.8963E-01 |
| $P_5$ | 2 | 20 | 30 000 | 0.75 | 0.75000 | 0.75000 | 0.75000 | 0.75000 | 6.7309E-07 |
| $P_6$ | 3 | 30 | 45 000 | -1 | -1.00000 | -1.00000 | -1.00000 | -1.00000 | 3.9167E-13 |
| $P_7$ | 5 | 100 | 150 000 | -30665.5 | -30665.5 | -30665.5 | -30665.5 | -30665.5 | 1.1959E-02 |
| $P_8$ | 10 | 100 | 150 000 | -1 | -1.00000 | -0.99996 | -1.00000 | -1.00000 | 1.8789E-05 |

The experimental results show that ALPSO reliably solves all benchmark problems. Due to the small constraint violation tolerances $\epsilon_{g,h}$ the best results of problem $P_1$, $P_2$ and $P_4$ are slightly better than

the expected optimal values. No infeasible solutions were obtained during all test runs. In comparison with [13] the results from ALPSO are comparable or superior with less function evaluations required. The number of function evaluations listed in Table 1 represents an upper limit where we stopped the optimization process. However, the best solution of each run was usually found much earlier. The Augmented Lagrange Particle Swarm Optimization method also reliably detects the active constraints and computes accurate Lagrange multiplier estimates. Table 2 lists the mean values and the corresponding standard deviation of the multipliers obtained during the 30 test runs on the eight benchmark functions.

Table 2: Lagrange multiplier estimates of the 30 runs solving the eight benchmark problems. See the Appendix for their description.

| problem | $\boldsymbol{\lambda}_{opt}$ | $\boldsymbol{\lambda}_{mean}$ | $\boldsymbol{\lambda}_{std.dev.}$ |
|---|---|---|---|
| $P_1$ | [-6 4] | [-6.0000 4.0000] | [6.2682E-05 7.8124E-04] |
| $P_2$ | [0.1396 0] | [0.1396 0] | [1.1589E-04 0] |
| $P_3$ | [0 0] | [0 0] | [0 0] |
| $P_4$ | [1097.1 1229.5] | [1097.1 1230.1] | [3.4552E-00 2.8377E-00] |
| $P_5$ | [1] | [1.0000] | [4.6999e-05] |
| $P_6$ | [0] | [0] | [0] |
| $P_7$ | [403.27 0 0 0 0 809.43] | [403.26 0 0 0 0 809.42] | [2.6272E-02 0 0 0 0 2.9262E-02] |
| $P_8$ | [5] | [4.9998] | [6.6372E-04] |

For the test problems mentioned above as well as for many other test problems conducted during the development of the algorithm, $k_{max} = 3$ subsequent basic PSO iterations without any update of Lagrange multipliers and penalty factors emerged to be sufficient. Many problems could even reliably be solved with $k_{max} = 2$, which almost fully eliminates the computational overhead of the nested loop structure. However, we consider $k_{max} = 3$ as a safeguard that still allows an considerable reduction of the two-loop structure.

As mentioned above, we used a constant inertia factor of $w = 0.9$ and relatively small numbers for the cognitive and social scaling factors of $c_{1,2} = 0.8$ or $c_{1,2} = 0.4$, respectively. Since the Lagrange multipliers as well as the penalty factors are dynamically updated after $k_{max}$ iterations, we apply a non-stationary pseudo objective function as described by Eq.(16) to the PSO algorithm. Premature convergence would make it difficult for the swarm to track the changing extrema. Therefore, we alleviate the influence of the cognitive and social attractors in the velocity update equation Eq.(10) by reducing the corresponding factors. For the same reason, we do not apply a linearly decreasing inertia factor $w$, we rather extend the velocity update equation by a small stochastic craziness term for improved tracking behavior.

## 8. Engineering Example: Hexapod Robot

The applicability of the algorithm for demanding engineering problems was tested during the optimization process of an hexapod robot. Machines with parallel kinematics feature low inertia forces due to low masses of the structure combined with possible high accuracy and stiffness. Such machines are currently under investigation in various fields of engineering like robotics, measurement systems and manufacturing technologies. We investigated and optimized the stiffness behavior of the hexapod robot HEXACT using the Augmented Lagrange Particle Swarm Optimization method. HEXACT is a research machine tool with parallel kinematics, developed at the Institute of Machine Tools at the University of Stuttgart, see Figure 2. A general drawback of parallel kinematic robots is the appearance of kinematically singular configurations that have to be avoided during operation. For the current design of the investigated hexapod machine such singularities are located along the tool axis in the central position of the machine, where the rotational stiffness around the tool axis decreases to zero, see [7]. One way to reduce the flexibility and to eliminate the singular configurations is to alter the angle $\zeta$ between the telescope struts and the end-effector, which are perpendicular in the initial design. Additionally, the radius $r_e$ and the length $l_e$ of the end-effector and thus the mounting points of the struts can be varied to improve the stiffness behavior of the hexapod robot.

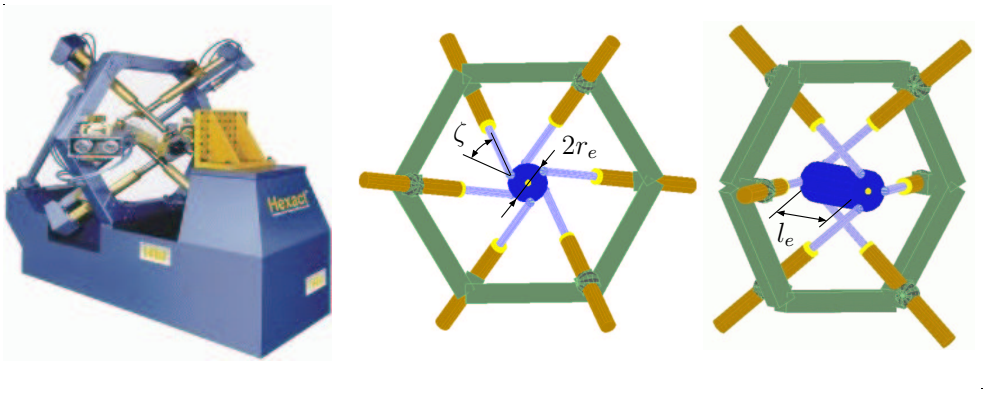


Figure 2: Hexapod robot HEXACT (www.ifw.uni-stuttgart.de) and our model.

The stiffness of the end-effector depends on its translational and rotational position in the workspace, the so-called pose. The relation between the applied forces and torques on the one hand and the evasive displacements of the end-effector on the other hand is highly nonlinear and can be described by the tangential stiffness matrix $\boldsymbol{K}_t$, see [7]. In order to gain more insight into the global flexibility behavior of the machine, it is necessary to evaluate the stiffness matrix at several poses in the workspace. Here, $N = 6^5$ sample poses on a regular grid are regarded. As a global flexibility criterion that is to be minimized, we use the negative average of the minimum principal stiffness of the $N$ sample poses in the entire workspace

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f_k(\boldsymbol{x}) = -\frac{1}{N}\sum_{j=1}^{N} \min(k_i^*)_j, \tag{21}$$

where $\min(k_i^*)$ is the minimum eigenvalue of the tangential stiffness matrix $\boldsymbol{K}_{t,j}$ at pose $j$.

The objective function described by Eq.(21) was solved using Particle Swarm Optimization which enables a gradient free and global search without any difficult or expensive gradient calculations or without any restrictions to a local solution. Two different sets of design variables were considered. The first variant considers only $\zeta$ as a design variable, whereas variant 2 takes into account both $\zeta$ and the dimension of the end-effector described by $r_e$ and $l_e$. Regarding design variant 1, the average of the minimum principal stiffness could be improved by a factor of 35 with respect to the initial design, which is confirmed by the results described in [7]. The optimization of design variant 2 improved the average stiffness by a factor of approximately 200. As expected the geometry of the end-effector has a great influence on the stiffness behavior of the hexapod robot. However, the maximization of the minimum stiffness as formulated by Eq.(21) impairs the stiffness distribution of the hexapod machine in the workspace. The standard deviation of the minimum principal stiffness in the workspace increased by a factor of 11 for design variant 1 and by a factor of 82 for design variant 2. The resulting more non-uniformly distributed stiffness behavior is undesirable for manufacturing processes. Therefore, design variant 3 is defined by the following nonlinearly constrained optimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f_s(\boldsymbol{x}) = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\min(k_i^*)_j - \sum_{q=1}^{N}\frac{\min(k_i^*)_q}{N})^2}, \tag{22}$$

$$\text{subject to} \quad h(\boldsymbol{x}) = f_k(\boldsymbol{x}) - 0.8 f_k^* \leq 0, \tag{23}$$

where $f_s(\boldsymbol{x})$ describes the standard deviation of the minimum principal stiffness of the $N$ poses on a regular grid. The inequality constraint restricts the decrease in the average principal stiffness $f_k^*$ that was optimized in design variant 2. This nonlinear constrained problem was solved using the Augmented Lagrangian Particle Swarm Optimization algorithm with respect to the design variables $\zeta$, $r_e$ and $l_e$. For this optimization process as well as for design variants 1 and 2, we used $n_p = 20$ particles, $\nu_{max} = 30$

and $k_{max} = 3$ iterations. As a result we could reduce the standard deviation of the minimum principal stiffness by only 25% with a reduction in the average stiffness of 20%. Further information about the hexapod robot can be found in [7].

## 9. Conclusion

We have presented an extension of the stochastic Particle Swarm Optimization algorithm by the Augmented Lagrange Multiplier Method in order to solve optimization tasks with problem immanent equality and inequality constraints. The behavior of the basic PSO method is comparable to other stochastic algorithms, especially to Evolutionary Strategies. Our extension resulting in ALPSO shows convincing results regarding convergence properties and robustness in solving constrained problems without the need for infinite penalty factors. The algorithm automatically detects active constraints and provides exact Lagrange multiplier estimates which was shown by solving eight constrained benchmark tests. The applicability to engineering problems was demonstrated by optimizing the stiffness behavior of a hexapod machine tool. In order to increase the stiffness of the hexapod robot, we successfully applied ALPSO to the optimization problem formulated by the use of the tangential stiffness matrix.

We believe that the extended Particle Swarm Optimizer is an alternative technique for solving real-life optimization problems. Its simplicity and its derivative free search for global solutions makes Particle Swarm Optimization attractive for many optimization tasks. Especially ALPSO with its extension to equality and inequality constraints is a highly competitive approach for small and medium-scale problems as they often occur in the field of mechanical engineering. A web-based interface to ALPSO can be found in the World Wide Web at the address `www.mechb.uni-stuttgart.de/research/alpso`. It is limited to two design variables but demonstrates the functionality and the performance of ALPSO.


## 10. Acknowledgements

The authors greatly appreciate the help of C. Henninger from the Institute B of Mechanics, University of Stuttgart. He developed the hexapod model used in Section 8 and provided assistance during the optimization process of the hexapod robot.


## 11. Appendix: Benchmark Problems

The Rosenbrock function ($R$), the Griewank function ($G$) as well as the test problems used in Section 7 are given here for completeness. Further information can be found in [1] and [13].

**R**
$$f(\boldsymbol{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$
$$-10 \le x_i \le 10, \quad i = 1, 2$$

**G**
$$f(\boldsymbol{x}) = \frac{1}{4000}(x_1^2 + x_2^2) - \cos(\frac{x_1}{\sqrt{1}})\cos(\frac{x_2}{\sqrt{2}}) + 1$$
$$-10 \le x_i \le 10, \quad i = 1, 2$$

**$P_1$**
$$f(\boldsymbol{x}) = x_1^2 + x_2^2$$
$$g_1(\boldsymbol{x}) = x_1 - 3 = 0$$
$$h_1(\boldsymbol{x}) = 2 - x_2 \le 0$$
$$-10 \le x_i \le 10, \quad i = 1, 2$$

**$P_2$**
$$f(\boldsymbol{x}) = \frac{1}{4000}(x_1^2 + x_2^2) - \cos(\frac{x_1}{\sqrt{1}})\cos(\frac{x_2}{\sqrt{2}}) + 1$$
$$g_1(\boldsymbol{x}) = x_1 - 3 = 0$$
$$h_1(\boldsymbol{x}) = 2 - x_2 \le 0$$
$$-10 \le x_i \le 10, \quad i = 1, 2$$

**$P_3$**
$$f(\boldsymbol{x}) = \frac{-\sin(2\pi x_1)^3 \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$
$$h_1(\boldsymbol{x}) = x_1^2 - x_2 + 1 \le 0$$
$$h_2(\boldsymbol{x}) = 1 - x_1 + (x_2 - 4)^2 \le 0$$
$$0.1 \le x_1 \le 10, \quad 0 \le x_2 \le 10$$

**$P_4$**
$$f(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$
$$h_1(\boldsymbol{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$
$$h_2(\boldsymbol{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$
$$13 \le x_1 \le 100, \quad 0 \le x_2 \le 100$$

**$P_5$**
$$f(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2$$
$$g_1(\boldsymbol{x}) = x_2 - x_1^2 = 0$$
$$-1 \le x_i \le 1, \quad i = 1, 2$$

**$P_6$**
$$f(\boldsymbol{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$
$$h_j(\boldsymbol{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$
$$0 \le x_i \le 10, \quad i = 1, 2, 3$$
$$p, q, r = 1, 2..., 9, \quad j = 1, 2, ..., 9^3$$

$$\boldsymbol{P_7} \quad f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$$h_1(\boldsymbol{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$
$$h_2(\boldsymbol{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$
$$h_3(\boldsymbol{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$
$$h_4(\boldsymbol{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$
$$h_5(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$
$$h_6(\boldsymbol{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

$$78 \leq x_1 \leq 100, \quad 33 \leq x_2 \leq 45, \quad 27 \leq x_i \leq 45, \quad i = 3, 4, 5$$

$$\boldsymbol{P_8} \quad f(\boldsymbol{x}) = -\sqrt{10}^{10} \prod_{i=1}^{10} x_i$$

$$g_1(\boldsymbol{x}) = \sum_{i=1}^{10} x_i^2 - 1 = 0$$

$$0.1 \leq x_i \leq 1, \quad i = 1(1)10$$

## 12. References


[1] Bergh van den, F.: An Analysis of Particle Swarm Optimizers. *Dissertation*, University of Pretoria, 2001.

[2] Clerc, M.: Discrete Particle Swarm Optimization. *New Optimization Techniques in Engineering*, Springer, 2004.

[3] Clerc, M.: The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, 1999, pp. 1951-1957.

[4] Coello, C.A.; Lechuga, M.S.: A Proposal for Multiple Objective Particle Swarm Optimization. *Technical Report EVOCINV-01-2001*, Evolutionary Computation Group at CINVESTAV-IPN, Mexico, 2001.

[5] Gill, P.E.; Murray, W.; Wright, M.H.: Practical Optimization. *Academic Press Limited*, London, 1981.

[6] Hansen, N.; Kern, S.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions. *Proceedings of 8th International Conference on Parallel Problem Solving from Nature*, Berlin, Springer 2004, pp. 282-291.

[7] Henninger, C.; Kübler, L.; Eberhard, P.: Flexibility Optimization of a Hexapod Machine Tool. *GAMM Mitteilungen*, 2004, 27(1), pp. 46-65.

[8] Hu, X.; Eberhart, R.; Shi, Y.: Engineering Optimization with Particle Swarm. *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, 2003, pp. 53-57.

[9] Ingber, L.: Very Fast Simulated Reannealing. *Mathematical Computational Modeling*, 1989, 12, pp. 967-973.

[10] Kennedy, J.; Eberhart, R.: Particle Swarm Optimization. *Proceedings of the International Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942-1948.

[11] Kennedy, J.; Eberhart, R.: Swarm Intelligence. *Academic Press*, London, 2001.

[12] Laskari, E.C.; Parsopoulos, K.E.; Vrahatis, M.N.: Particle Swarm Optimization for Integer Programming. *Proceedings of the IEEE Congress on Evolutionary Computation*, Honolulu, 2002.

[13] Runarsson, T.P.; Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 2000, 4, pp. 284-294.

[14] Schutte, J.F.; Groenwold, A.A.: The Optimal Sizing Design of Truss Structures Using the Particle Swarm Optimization Algorithm. *In Proc. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, 2002.

[15] Sedlaczek, K.; Eberhard, P.: Optimization of Nonlinear Mechanical Systems under Constraints with the Particle Swarm Method. *Proceedings of Applied Mathematics and Mechanics*, 2004, 4(1), pp. 169-170.

[16] Venter, G.; Sobieszczanski-Sobieski, J.: Particle Swarm Optimization. *Proceedings of the 43rd AIAA/ ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Denver, 2002.