Optimisation of parallel mechanisms with joint limits and collision constraints

Durgesh H Salunkhe[a,*], Shivesh Kumar[b], Damien Chablat[a]

[a]*Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, 44000 Nantes, France*
[b]*Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI GmbH), 28359 Bremen, Germany*

Abstract

In this chapter, a new optimization methodology for parallel kinematic manipulator (PKM) is proposed, addressing constraints related to singularities, passive joint limits, and self-collisions. The proposed optimization approach combines a local search method (Nelder-Mead algorithm) with global search methodologies such as low discrepancy distribution. This combination allows for faster and more efficient exploration of the optimization space. The algorithm optimizes a global kinematic quality along with the length of the prismatic actuators. The design constraints can be introduced modularly. This allows for a better understanding of the impact of specific criteria on the final result. The chapter also discusses the effect of the dimension of the search space. It is shown that initial knowledge on PKM can help reduce the dimension of the search space and result in more intuitive results. The presented approach is applied to optimize a PKM with a motion constraint generator of 2 degrees of freedom. This case study demonstrates the effectiveness of the proposed methodology in addressing the challenges associated with the optimization of PKMs.

Keywords: parallel mechanisms, optimization, Nelder-Mead

1. Introduction

Owing to their advantages, Parallel Kinematic Manipulators (PKM) are employed as sub-mechanism modules in various fields such as humanoid robots (THOR [1], LOLA [2], Charlie [3]), exoskeletons [4, 5], haptic interfaces [6], surgeries [7], and industrial applications [8, 9]. An extensive survey on PKM with classification based on degrees of freedom and their applications is presented in [10]. PKMs are also widely used in high-speed industrial assembly lines, like the DELTA + 1 DOF wrist robot [11]. Another significant application of PKMs is

*Corresponding author

in machining tasks such as milling operations and high-speed machining tasks [12, 13, 14].

Given the broad range of applications, PKM designs must cater to user needs and adhere to constraints associated with different processes. These needs may involve robot mobility, workspace size, movement precision, dynamic performance, and stiffness. Numerous performance indices have been established to address these requirements, which can be applied to optimization problems. Some workspace and kinematic performance indices include the Jacobian matrix conditioning, velocity amplification factors [15, 16, 17], regular workspace shapes [18], and safe working zones [19].

Various optimization methods have been proposed for mechanism synthesis in the past. Some employ the mathematical formulation of the objective function to implement gradient descent methods [20], while others use numerical approaches and evolutionary algorithms when the objective function is not available in closed form or gradient-based algorithms cannot be used. Some of these algorithms include Differential Evolution (DE) [21], Genetic Algorithms (GA) [22], Branch and Prune [23], Interval-based analysis [18], and Non-dominated Sorting Genetic Algorithm II (NSGA-II) [24, 25, 26] for multi-objective optimization. These methods are typically computationally expensive, with efficiency heavily reliant on population size.

A recent development in mechanism design optimization involves co-optimization with motion trajectories [27]. This approach employs efficient algorithms to explore implicitly defined manifolds, leveraging the advantages of representing the problem as an implicit function for faster and more efficient convergence.

Local search methods can decrease the computational cost of mechanism optimization. The Nelder-Mead algorithm, a geometric-based search for the next best solution, is well-suited for mechanism optimization, as it allows easy optimization of link lengths. To prevent convergence to a local optimum, different methodologies combine local optimization methods with global searches [28, 29, 30, 31].

This chapter introduces an accelerated, general algorithm for PKM design optimization that is flexible concerning objective function definition and adaptable to constraints. The method optimizes the design for the maximum safe working zone while considering the physical stroke of the prismatic actuator. A fast local search algorithm, the Nelder-Mead algorithm, combined with a global search procedure, enables quicker progress toward a global optimum, even for mechanisms with computationally expensive objective functions.

2. Design considerations in PKM optimization

In the parallel kinematic mechanism design, the following choices have to be made:

1. Architecture of the manipulator (e.g: 3RRR(Revolute-Revolute(actuated)-Revolute), 3RPR(Revolute-Prismatic(actuated)-Revolute) etc.)

2. Type of joints: different combinations of joints to achieve the same degrees of freedom (dof) (e.g.: U<u>P</u>S(Universal-Prismatic(actuated)-Spherical), <u>R</u>US, <u>R</u>RPS)

3. Pose of the joints: where and how to place a particular joint's frame?

Making a particular choice is non-trivial, especially because of its effect on the workspace, the direct and inverse kinematic model, and the size of the mechanism. Another interesting challenge is that the same architecture can perform different tasks with either kinematic or dynamic constraints and thus have to be optimized accordingly. The following subsections elaborate on the common objective functions and constraints involved in mechanism optimization to motivate the choice of the algorithm.

2.1. Objective function

It is important to evaluate the quality of the motion performed while designing a manipulator with kinematic characteristics. The quality indices widely used in the past are the conditioning number [15] and the manipulability ellipsoid [16]. The feasible workspace and the global quality of the manipulator are directly related in the presented case and thus can be implemented together with appropriate weights.

2.1.1. Manipulator workspace

If the workspace involves only orientation or translation, Regular Dextrous Workspace (RDW) is an objective function representing an n-dimensional sphere within the n-dimensional output space. The necessary workspace is not considered a constraint, but rather, the algorithm aims to achieve the largest feasible workspace within the desired RDW (RDW_d) [18]. Concurrently, the notion of *safe working zone* for parallel manipulators has been presented in [19], defining a feasible workspace as one devoid of singularities, internal link collisions, and adhering to passive joint limits. The feasible set ($\mathcal{F}$) concept in this text pertains to the collection of all points in the discretized output space ($\mathcal{K}$), such that:

1. They correspond to non-singular configurations

2. Adhere to passive joint limitations

3. Ensure no internal collisions between actuators and the moving platform for all postures

2.1.2. Quality of the manipulator

To measure the motion quality, the conditioning number (κ) was introduced in [15]. It signifies the asymptotic worst-case relative change in the output for a relative change in the input, evaluating the output sensitivity to input changes.

The geometrical interpretation of κ relates to the ellipsoid's eccentricity proportionality, providing information about the ease of movement in a specific direction from the current end effector pose. When the κ equals 1, it corresponds to a sphere and the *isotropic configuration*. The κ value ranges from 1 to ∞, and its inverse, κ^{-1}, is used for bounded values and is given by (1), where σ represents the Jacobian matrix, $\mathbf{J}$, singular values.

$$\kappa^{-1} = \frac{\sigma_{min}}{\sigma_{max}}, \quad \kappa^{-1} \in [0, 1] \tag{1}$$

The Jacobian matrix's dimensional non-homogeneity affects the conditioning number and is unsuitable for manipulators whose workspace is not a subset of either $\mathbb{R}^3$ or $SO(3)$ [32]. This issue is vital to consider when implementing the proposed optimization methodology for a general manipulator. The manipulators shown in Section 4 have only rotational dof, so the inverse conditioning number is chosen as the quality index. A *global conditioning index (κ_g^{-1})* (GCI), the mean quality index (κ^{-1}) over the RDW, is defined as follows,

$$\kappa_g^{-1} := \frac{\sum_{1}^{RDW_d} \kappa^{-1}}{RDW_d} \tag{2}$$

2.2. Constraints

PKM's most common constraints to implement feasible workspace include:

- non-singular constraint
- passive and active joint limits
- internal collision constraints
- feasible actuator range

Among these constraints, the first three are self-explanatory and will not be elaborated further. The constraint regarding feasible actuator range is pertinent in optimising PKM with prismatic actuators and is discussed in detail to emphasize its importance.

2.2.1. Feasible actuator range

The active joint ranges are an essential constraint during PKM design. This constraint is particularly relevant to mechanisms with prismatic joints as actuators. The goal is to constrain the actuator selection to maximize the points in $\mathcal{F} \bigcap RDW_d$. Typically, a prismatic joint is represented as a constraint with a specific minimum and maximum range and with a constraint on the ratio between the length in the fully actuated state and its default length:

$$\rho_{min} \leq \rho \leq \rho_{max} \tag{3}$$

$$\rho_{max} \leq \text{stroke} \cdot \rho_{min}, \quad stroke \in [1, 2] \tag{4}$$

Equation 4 originates from the physical structure of general prismatic joints. If the actuator's unextended length is ρ_{min}, then it is impractical for typical prismatic joints to extend beyond their original length ($\rho_{max} < 2 \cdot \rho_{min}$). The novelty in expressing the actuator range in the current work is that we do not have a static value as a limit as mentioned in Equation 3, i.e., we express the constraint solely in terms of the stroke ratio defined in Equation 4. This allows us to select the optimal actuator ranges to maximize the feasible workspace without imposing constraints on the prismatic joint's minimum or maximum size. This is demonstrated in figures 1 and 2, which introduce an example for a 2 dof 2UPS-1U orientation mechanism from [7]. The points in the dotted space in figure 1 correspond to actuator lengths in a feasible configuration. The objective is to search for an optimized bracket, $[\rho_{min}, \rho_{max}]$, i.e., a bracket encompassing as many blue points as possible, with the constraint that the square's side does not exceed a given proportion relative to its minimum value.

An algorithm presented in [33] (Algorithm 1) details the method used to obtain the optimized bracket for the actuators. After discretizing the RDW_d, we acquire the set of all valid points belonging to $\mathcal{F}$. Upon calculating the actuator length values at each point, the minimum ρ_{min} and maximum ρ_{max} value for the actuator is determined. The algorithm input is an n x 3 matrix for the n valid points, with columns corresponding to the actuator lengths and the evaluation at that point. If the ratio of the maximum value to the minimum value of the actuator length respects the stroke ratio, the algorithm returns the actuator range without modification. Otherwise, a bracket of $[\rho_{min}, \text{stroke}.\rho_{min}]$ is generated, and the actuator length values for each point in the set of valid points are checked against the bracket, and the number of points satisfying the bracket is stored. This process repeats by incrementing the ρ from ρ_{min} to $\text{stroke}.\rho_{min}$.

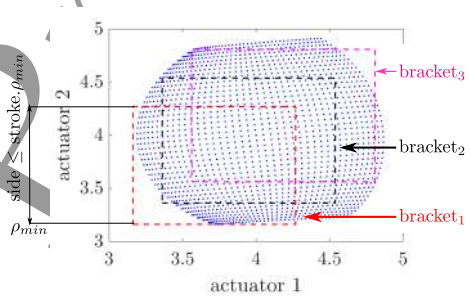
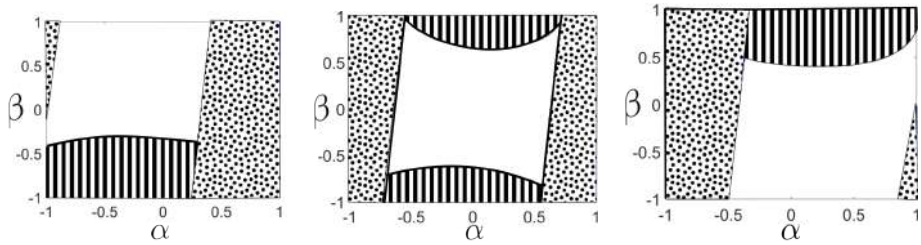


Figure 1: Different search brackets within the actuator space (input space). The dots correspond to the pair of lengths of actuators for a configuration in RDW.

2.2.2. Implementation of constraints and evaluation function

The process of implementing constraints and evaluating the performance of a set of parameters is explained in algorithm 1. The optimization space is first



(a) Feasible workspace (white) when bracket 1 in figure 1 is implemented

(b) Feasible workspace (white) when bracket 2 in figure 1 is implemented

(c) Feasible workspace (white), bracket 3 in figure1 is implemented

Figure 2: Comparison of feasible workspace (white space) within the RDW_d for different search brackets and a specific mechanism (2UPS-1U). The striped and dotted part represent the violation due to actuator lengths of first and second leg, respectively.

discretized to evaluate the parameters, and each point is evaluated for compliance with the constraints. Some constraints are strictly enforced, meaning that if any point in the RDW_d violates them, the set of parameters is considered invalid. In the current algorithm, the singularity constraint is strict. If the singularity curve intersects with even one point of RDW_d, the evaluation of the given parameters is negative. If the RDW_d is singularity-free, each point is evaluated for compliance with other constraints, such as passive joint limits and collision constraints. If a point satisfies all the constraints, it is rewarded with the corresponding κ^{-1} value. If any constraint (except singularity) is violated, the point in RDW_d is given a 0 value.

As each point in the discretized workspace is evaluated, the final evaluation is the cumulative value of κ^{-1} over the workspace where all constraints are satisfied. The rewarding strategy can be customized per the designer's requirements, and appropriate weightage can be assigned to the constraints to achieve an optimized design for a specific need. The algorithm demonstrates modularity with the constraints, where each constraint is independent. The flexibility to activate, deactivate, or add constraints without changing the algorithm is particularly useful for mechanism design. The designer can experiment with various constraints to understand their effects on the final feasible workspace. Each constraint can be designed to reward or penalize a specific set of parameters, allowing for a mix of strict and non-strict constraints in the optimization. The designer can also identify which constraint hinders the optimization and requires modification.

In summary, evaluating a given set of parameters involves discretizing the optimization space, assessing each point for compliance with the constraints, and rewarding the points that satisfy all constraints. The algorithm is modular and flexible, allowing the designer to experiment with various constraints to achieve an optimized design for specific needs.

Algorithm 1: Method to calculate the evaluation and ρ_{range} for a set of parameters

Result: evaluation at a given point in optimization space and the corresponding actuator lengths

input $\rightarrow \mathbf{v}$ ▷ It is a n-dimension point in given n-dimension optimization space;
$x_i, i \in 1, .., n,$ ▷ i^{th} variable of the n-dimension optimization space;
ρ_1 and ρ_2 ▷ actuator lengths at a given configuration;
e = 0 ▷ Initialising the evaluation;
for x_1 ***from*** x_{1min} ***to*** x_{1max} ***by*** $interval_i$ **do**
    ... ▷ Add loops as a function of the dimension of the space
    for x_n ***from*** x_{nmin} ***to*** x_{nmax} ***by*** $interval_n$ **do**
        f($\mathbf{v}$) ▷ function that solves IGS, collision distance and κ^{-1};
        $[\det(\mathbf{J}), \mathbf{q}_p,\ \rho_1,\ \rho_2,\ \kappa^{-1},\ d_c] = \text{f}(\mathbf{v})$;
        f($\mathbf{v}$) returns the value of the determinant of Jacobian, the passive joint angle vector, $\mathbf{q}_p$, actuator lengths, $[\rho_1, \rho_2]$, the inverse of the conditioning number, κ^{-1} and the collision distance, d_c, between the actuators;
        ▷ 1. Checking for singularity constraints;
        if *det*($\mathbf{J}$) ***is*** *0* **then**
            e = -∞;
            break;
        else
            $reward = \kappa^{-1}$
        end
        ▷ 2. Checking the passive joint limits;
        for *i* ***from*** *1* ***to*** *length of* $\mathbf{q}_p$ **do**
            if $q_{pi} \geq q_{pmax}$ ***or*** $q_{pi} \leq q_{pmin}$ **then**
                $reward = 0$
            else
                $reward = \kappa^{-1}$
            end
        end
        ▷ 3. Checking for collision constraints;
        if $d_c \geq threshold$ **then**
            $reward = 0$
        end
        e = e + reward;
        valid_points[i] = [ρ_1, ρ_2, reward];
    end
    ...
end
Implement the algorithm presented in [33] (Algorithm 1);
return valid_points, e, ρ_1, ρ_2

3. Proposed Algorithm for Mechanism Optimization

In this section, we present the complete optimization method. As discussed in previous sections, the goal is to develop an algorithm capable of managing non-smooth objective functions and PKM design constraints. This section is organized into three subsections, explaining the local search, global search, and the approach used to combine them for faster and more efficient solutions, respectively.

3.1. Local search algorithm: The NM (NM) algorithm

The NM-algorithm, a derivative-free optimization algorithm, was proposed by John Nelder and Roger Mead [34]. It is also called the *downhill-simplex algorithm* since it employs *simplexes* to conduct a local space search. In this section, we introduce the algorithm for a *single start*, which looks for the optimal solution in the local vicinity of the initial simplex. We then discuss the algorithm's application in mechanism optimization and describe the method for extracting the best actuator ranges from the solution. The section concludes with an overview of the algorithm and its implementation, highlighting its strengths and weaknesses.

To avoid *premature convergence* in an n-dimensional optimisation space ($\mathcal{O}$), a simplex with at least n+1 points in $\mathcal{O}$ is needed. As shown in the figure, this can be visualized with a simple graphic for a 2-dimensional, $\mathcal{O}$. The algorithm starts with a sorted simplex of n+1 points $(\mathbf{v_0}, \mathbf{v_1}, ...\mathbf{v_n})$ such that the objective function evaluated at the i^{th} vertex has a value better than or equal to that of the $(i+1)^{th}$ vertex. A mean point $(\mathbf{v_m})$ is calculated by excluding the worst point $(\mathbf{v_n})$:

$$\mathbf{v_m} := \frac{\sum_{i=0}^{n-1} \mathbf{v_i}}{n} \tag{5}$$

The optimization algorithm then compares the mean point and searches for better points by geometrical operations termed as (i) reflection, (ii) expansion, (iii) inside contraction, (iv) outside contraction and (v) shrinkage. These operations are defined as follows:

1. Reflection $(\mathbf{v_r})$:

$$\mathbf{v_r} = \mathbf{v_m} + r\,(\mathbf{v_m} - \mathbf{v_n}), \quad r = \text{reflection coefficient}\,(r > 0) \tag{6}$$

2. Expansion $(\mathbf{v_e})$:

$$\mathbf{v_e} = \mathbf{v_m} + e\,(\mathbf{v_r} - \mathbf{v_m}), \quad e = \text{expansion coefficient}\,(e > 1) \tag{7}$$

3. Outside contraction $(\mathbf{v_{oc}})$:

$$\mathbf{v_{oc}} = \mathbf{v_m} + k\,(\mathbf{v_m} - \mathbf{v_n}), \quad k = \text{contraction coefficient}\,(0 < k < r) \tag{8}$$

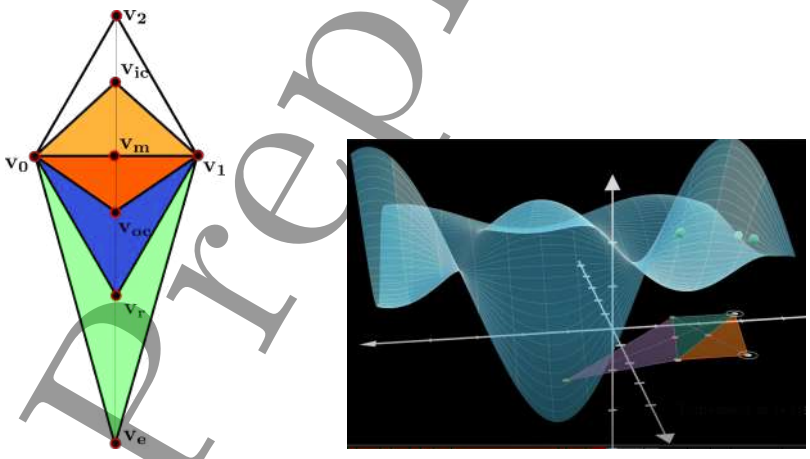
4. Inside contraction ($\mathbf{v_{ic}}$):

$$\mathbf{v_{ic}} = \mathbf{v_m} - k\,(\mathbf{v_m} - \mathbf{v_n}), \quad k = \text{contraction coefficient} \tag{9}$$

5. Shrinkage:

$$\forall\, i \in [1, n] \quad \mathbf{v_i} = s\,.\mathbf{v_i}, \quad s := \text{shrinkage factor}\,(0 < s < 1) \tag{10}$$

The introduction of a new point ($\mathbf{v_n}$) into the simplex relies on the evaluation of $\mathbf{v_r}$, $\mathbf{v_e}$, $\mathbf{v_{oc}}$, and $\mathbf{v_{ic}}$ (refer to Algorithm 2). The process continues until the stopping criteria are met. The simplex halts if it shrinks below a specific value, ϵ_1, and the evaluations of every vertex of the reduced simplex deviate by a maximum threshold, ϵ_2. The algorithm can also be stopped by limiting the number of iterations. Algorithm 2 provides the full procedure for a single start of the Nelder-Mead (NM)-algorithm, and the stopping criteria algorithm can be found in [33] (Algorithm 3). Figure 3a illustrates an example of the operations in a 2-dimensional optimization space, $\mathcal{O}$, demonstrating the geometric search nature of the $\mathcal{O}$ in the NM-algorithm. Figure 3b graphically depicts an example of the points explored during an optimization process. The optimization space of 2 dimensions of the evaluation is a function of these parameters.



(a) An example of an operation on a simplex (defined by v_0, v_1, v_2) in 2-dimensional $\mathcal{O}$

(b) An example of the travel path of optimization in NM-algorithm

Figure 3: The single start of the Nelder-Mead local search

Algorithm 2: Single start of the NM-algorithm

Result: Local minimum evaluation and the optimized parameters

initial sorted simplex $\{\mathbf{v_0}, \mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_{n-1}}, \mathbf{v_n}\}$;
evaluations $\{e_0, e_1, e_2, ..., e_{n-1}, e_n\}$;
while *stop = 0* **do**
    calculate $\mathbf{v_m}, \mathbf{v_r}$ and e_r;
    if $(e_n < e_r < e_0)$ **then**
        $\mathbf{v_n} = \mathbf{v_r}$;
    else if $(e_0 < e_r)$ **then**
        if $(e_r < e_e)$ **then**
            $\mathbf{v_n} = \mathbf{v_e}$;
        else
            $\mathbf{v_n} = \mathbf{v_r}$;
        end
    else if $(e_n < e_r < e_{n-1})$ **then**
        if $(e_{oc} > e_r)$ **then**
            $\mathbf{v_n} = \mathbf{v_{oc}}$;
        else
            $\forall\, i \in [1, n] \quad \mathbf{v_i} = s.\mathbf{v_i}$;
        end
    else if $(e_r > e_n)$ **then**
        if $(e_{ic} > e_r)$ **then**
            $\mathbf{v_n} = \mathbf{v_{ic}}$;
        else
            $\forall\, i \in [1, n] \quad \mathbf{v_i} = s.\mathbf{v_i}$;
        end
    sort the simplex;
    if $\mathbf{v_{0new}} > \mathbf{v_0}$ **then**
        iter = 0
    else
        iter = iter + 1
    end
    Update 'stop' from algorithm in [33] (Algorithm 3)
end
return $\mathbf{v_0}, e_0$

3.1.1. Advantages and drawbacks of the NM-algorithm

The NM-algorithm offers a simple approach for modelling optimization problems in mechanism design, enabling the development of a general methodology applicable to any parallel mechanism. As a derivative-free algorithm, it introduces complex objective functions that may be difficult to formalize, such as the quality index, κ_g^{-1}, described in Section 2.2. Additionally, the NM-algorithm is a local search algorithm that returns a stationary point in a relatively short time compared to currently employed global optimization methods. This en-

ables the designer to develop computationally expensive objective functions and construct constraints modularly, facilitating experimentation with different constraints throughout development. The geometric search method inherent to the NM-algorithm is another significant advantage relevant to mechanism design. The optimization space's foundation in the NM-algorithm is the optimization variables themselves, making it logical to use this method as the following best design parameters are chosen based on the combination of previous simplex parameters rather than using complex methods to represent a mechanism in the optimization space that may not have a geometrical explanation for selecting the next best proposal (e.g., chromosomes in Genetic Algorithm). It is also possible to tune exploring parameters, such as reflection, expansion, contraction, and shrinkage coefficients, using human intuition and prior knowledge regarding the importance of different parameters.

Although well-suited for our application, the NM-algorithm has certain disadvantages. It has been proven to converge to dimension two under specific hypotheses [35] but lacks proof of convergence for optimization beyond two dimensions. It can collapse simplex patterns if implemented incorrectly, leading to convergence to a non-stationary solution [36]. Convergence highly depends on the initial simplex size and the coefficient choices, as discussed in [37]. Despite these limitations, the NM-algorithm is valuable for our purposes, as the goal is not to find the absolute optimized design parameter but to satisfy all constraints and achieve acceptable performance quality. Indeed, it has been successfully implemented in various applications [30, 31]. Convergent variants have been proposed to circumvent premature convergence [38], allowing the algorithm to explore additional points in case of near collapse.

The NM-algorithm's local search is combined with a multi-start technique for global search in the optimization space, as discussed in the following section.

3.2. Global search algorithm

The NM algorithm has been combined with other global search methods, such as low-discrepancy points [39], genetic algorithm [28], and Powell optimization [29], in previous work. To explore global optimisation space, we implement a multi-start NM-algorithm with low discrepancy points [40, 41, 42]. In this approach, the NM-algorithm is executed with different initial simplexes. It is crucial to have uniformly distributed initial simplexes to explore the maximum optimisation space area.

3.2.1. Starting simplexes for multi-start

A practical approach to generate a sampling set $\mathcal{O}_M \subset \mathcal{O}$ is *Monte Carlo sampling* using a uniform distribution (refer to, for instance, [43]), or in other words, *random sampling*. Unfortunately, it is recognized [41] that these points tend to create clusters, particularly in high-dimensional scenarios, compromising the uniformity of the discretization. A superior alternative involves distributing the , M points of the discretization , $\mathcal{O}_M$ of , $\mathcal{O}$ more evenly. Specifically, the points should be sufficiently close together without leaving any under-sampled

regions. Certain deterministic sampling methods can be employed for this purpose, as demonstrated in [42, 44]. The characteristics of such techniques are explained in [42]. The study in [42] proposes that an effective strategy to generate uniformly dispersed deterministic point sets involves using finite segments of so-called *low-discrepancy sequences* like the *Halton sequence*, the *Hammersley sequence*, and the *Sobol sequence*. The presented work applies initial simplexes selected from the Sobol sequences, as they demonstrate a more uniform distribution.

[44] compared a sampling of a 2-dimensional unit cube using a sequence of 500 points based on the uniform distribution and a sampling of the same cube obtained through a low-discrepancy sequence (in this instance, the *Sobol sequence* [45]). It has been sufficiently demonstrated that the second sequence provides better space coverage, with the largest voids among the points occurring in the case of uniform distribution.

3.3. Cascade optimization

In a standard execution of the NM-algorithm, the iteration ceases either when the simplex has contracted to a desirable size with approximately equal evaluations or if the same best point is encountered for a predetermined maximum number of iterations, as referred to the stopping algorithm [33] (Algorithm 3). Aiming to reduce the time for local convergence and enable the exploration of more initial simplexes, we adopt a method inspired by rough and fine-turning practices in lathe machines. Generally, when removing excess material from a workpiece as quickly as possible, the feed rate is increased, and the focus is not on the work's finish. Later, the feed rate is reduced when approaching the desired dimensions, and the emphasis shifts to the work's finishing. Figure 4 depicts the algorithm's entire flow. Initially, the simplexes derived from the Sobol sequence are integrated into the multi-start NM-algorithm, and a coarse search is conducted for local optima. Subsequently, local optima from some selected initialized simplexes are utilized to enforce stricter stopping criteria, enabling convergence to a stationary point with higher precision. We discard local optima that do not promise satisfactory evaluations even after an extended search, significantly reducing computation time. Moreover, with an optimized vertex as an initial simplex, we can construct the remaining vertices according to our preferences, thereby regulating the initial simplex's size.

3.3.1. Coarse search

In the coarse search, our goal is to hasten local convergence, enabling us to maximize the number of starts in our optimization approach. This is achieved by employing a coarser search space and easing the stopping criteria. During the coarse search, the output space is discretized using an interval ten times larger than in the finer search, significantly reducing computational time. The objective of the coarse search is to identify simplexes situated on comparatively steeper slopes in the optimization space. By relaxing the stopping criteria, the maximum number of iterations permitted to repeat with the same evaluation

is limited to 10, which aids in terminating the local search more quickly. One example of such a coarse implementation is detailed in Algorithm 3, where the condition for incrementing the iteration is altered. We apply a condition stating that the new evaluation found is considered better than the previous one only if it surpasses the last evaluation by 5%. The algorithm ceases as soon as we reach 90% of the maximum expected value.

3.3.2. Fine search

In the fine search, we sift through various local optima obtained from the coarse search. The evaluations of the local optima are sorted in ascending order, and the top 10% of the gathered optima are selected for further evaluation. In the fine search, we enforce stricter stopping criteria, modify the constraint of maximum expected evaluation to 100%, and discretize the output space with a ten times finer interval. The threshold considered as an improvement is reduced to 1%. These changes directly impact the computational time and result in a significantly longer duration with an increasing dimension of the output space. All the optimized parameter sets from the NM-algorithm with stricter constraints are compared, and the best point is suggested as an optimized parameter of the PKM.

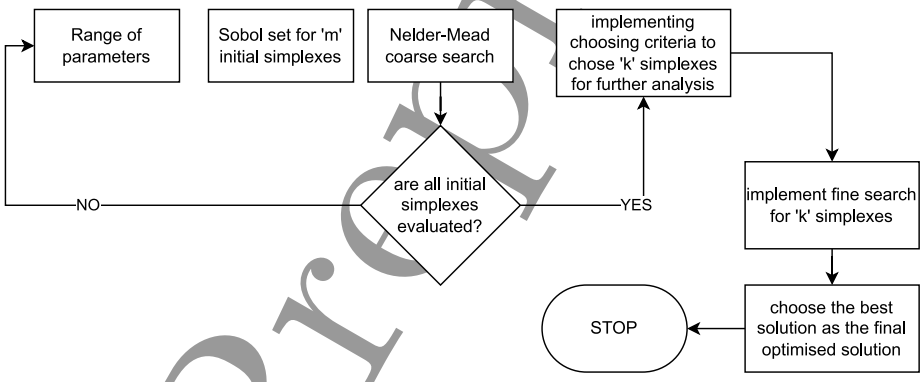


Figure 4: The flowchart for the complete implemented optimization methodology

4. Results and discussion

The optimization algorithm described in this chapter was employed to optimize a 2UPS-1U parallel mechanism to verify the general implementation. The chosen mechanism is widely utilized in the industry, and the significance of the selected objective function is also discussed in this section. An open-source implementation of the proposed algorithm and examples, including the lambda mechanism (1 dof) and 3RRR mechanism (3 dof), can be found at: `https://github.com/salunkhedurgesh/ParaOpt`.

Algorithm 3: implementation of coarse and fine local search criteria

Result: Optimised parameter set $\mathbf{v_0}$
input: Initial set of simplexes;
e_0: the best evaluation from the previous iteration;
e_{max} : Maximum expected evaluation;
limit: The percentage of maximum evaluation that is considered best;
For coarse search;
max_iter = 3 n;
margin = 1.05 ... (suggesting ≥5% increment is considered improvement);
limit = 0.8 ... (suggesting that 80% of maximum evaluation is a criterion to stop);
For fine search;
max_iter = 10 n;
margin = 1.01 ... (suggesting ≥1% increment is considered improvement);
limit = 1;
stop = 0;
while *stop = 0* **do**
    Perform algorithm 2 except for last step of checking stop from algorithm in [33] (Algorithm 3);
    Perform algorithm 1 with finer intervals;
    if $e_{new} \geq margin \times e_0$ **then**
        iter = 0
    else
        iter = iter + 1
    end
    if $iter \geq max\_iter$ **then**
        return stop = 1;
    end
    if $e_{new} \geq limit \times e_{max}$ **then**
        return stop = 1;
    end
end
return $\mathbf{v_0}$ from the algorithm 2

Algorithm 4: An example of implemented multi-start optimization

Result: Optimized parameter set of the mechanism and its evaluation

Assuming we have 'm' starts for an 'n' dimensional optimization problem;

Choose m.(n+1) valid n-dimensional points from the Sobol set generated;

Choose 'k' local optima for further fine search; generally, k ≤ 0.1 m;

for *start = 1:m* **do**

  Initial simplex = $\{\mathbf{v_{(m-1).(n+1)}} \cdots \mathbf{v_{mn+m-1}}\}$;

  Implement Single start from Algorithm 2 with coarse search from Algorithm 3;

  $\mathbf{v}_{chosen}(start, 1:n+1) = [\mathbf{v}_0, e_0]$;

end

sort $\mathbf{v}_{chosen}$ by evaluation of the corresponding parameter set;

for *fine_start = 1:k* **do**

  Generate n more parameter sets around $\mathbf{v}_{chosen}$(fine_start);

  Implement Single start from Algorithm 2 with fine search from Algorithm 3;

  $\mathbf{v}_{fine}$(fine_start, 1:n+1) = $[\mathbf{v}_0, e_0]$;

end

sort $\mathbf{v}_{fine}$ by evaluation of the corresponding parameter set;

return $\mathbf{v}_{fine}[1, 1:n]$, $\mathbf{v}_{fine}[n+1]$

4.1. 1-dof lambda mechanism

The λ-mechanism is a singular closed loop (1-RR$_{\mathrm{P}}$R) mechanism utilized in legged robots as a simplification of the revolute joint [2, 46, 47], as depicted in Figure 5. This mechanism is employed for stiffer actuation when a compact yet strong force is necessary and non-linear transmission characteristics are preferred. The constraint equations in this situation are simple and have been thoroughly examined in [48]. The mechanism was optimized using the determinant of the Jacobian, j, as the GCI and a modified VAF. The determinant is a scalar for the given case. For the lengths and variables illustrated in Figure 5, the computations for these measures are:

$$\begin{gathered}\rho^2 = l_1^2 + l_2^2 - 2l_1 l_2\cos(\theta), j = l_1 l_2 \frac{\sin(\theta)}{\rho} \\ \mathrm{GCI}_i = j, \mathrm{VAF}_i \begin{matrix} = \dfrac{1}{1+\sqrt{2}(j-1)^2} \\ = 0 \end{matrix} \Bigg\} \begin{matrix} \mathrm{VAF}_{min} < j < \mathrm{VAF}_{max} \\ \text{otherwise} \end{matrix} \\ \mathrm{GCI} = \frac{\sum_{i=1}^{n} \mathrm{GCI}_i}{n}, \mathrm{VAF} = \frac{\sum_{i=1}^{n} \mathrm{VAF}_i}{n}\end{gathered} \tag{11}$$

Parameters	Value	Parameters	Value
optimization dimension	1	Range of parameter	[1, 4]
Number of starts	100	Number of iterations	10
Objective choice	Workspace, GCI, VAF	Velocity amplification range	[0.3, 3]
Workspace (θ_1 range)	45^0 to 135^0	stroke ratio	1.5

Table 1: The parameters set for the optimization of 1-dof lambda mechanism

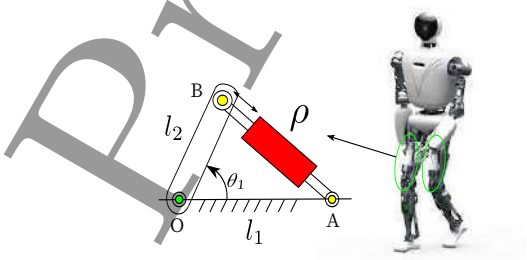


Figure 5: 1-dof lambda mechanism with real-life implementation [33]

In this mechanism, the l_1(OA) length is optimized with respect to l_2(OB), utilizing three distinct objective functions with parameters provided in Table 1. Initially, the workspace was maximized to identify an optimal length that covers the revolute joint's range from 45° to 135°. Subsequently, the GCI and VAF were employed as objective functions. The mechanism's acceptable velocity amplification span ranged from 0.3 to 3. The stroke ratio, the prismatic actuator's

fully extended length divided by its unextended length, was $\frac{3}{2}$. To approach a superior global optimum, 100 individual local Nelder Mead optimization starts were employed, and the number of iterations for the same evaluation within a single start was limited to 10. For all objective functions, multiple solutions with equal evaluation exist. It was observed that $l_1 = 4$ was proposed as the global optimum while optimizing for all different objective functions. Since the optimization dimension was only 1, this process was swift, completing 100 coarse single starts and 10 refined starts in 21 seconds. The evaluation increases to a specific value (3.39) and remains constant. This value is also the maximum possible evaluation in an ideal scenario. The results for the 1-dof lambda mechanism optimization are summarized in Table 2.

Parameters	GCI	VAF
Time for 1 coarse evaluation	1 second	1 second
Time for single coarse start	0.01 seconds	0.01 seconds
Time for one fine evaluation	5 seconds	3.1 seconds
Time for single fine start	0.04 seconds	0.02 seconds
Best point(l_2)	4	3.4
Best actuator range	[3.37 4.76]	[2.78, 4.17]

Table 2: The results for the optimization of 1-dof lambda mechanism

4.2. 2 dof RCM mechanism

In this section, a popular 2-dof parallel mechanism, 2UPS-1U, is optimized. This mechanism features a motion constraint generator and can be considered relatively complex for design optimization. This class of mechanisms has been employed in medical applications [7] as well as in implementing joint modules in humanoids (see [49, 50] for application as an ankle joint and [47, 51] for application as a torso joint). The first joints in leg 1 and leg 2 with respect to

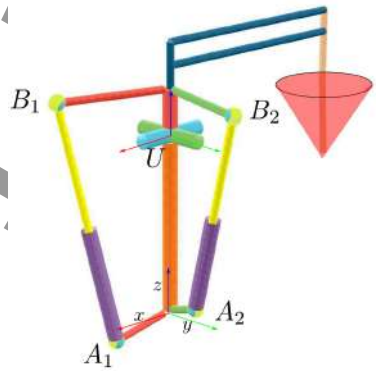


Figure 6: The parameters to be optimized in 2UPS-1U

the base can be given as:

$$A_1 = \begin{bmatrix} a_1 \cos\phi_1 \\ a_1 \sin\phi_1 \\ h_1 \end{bmatrix}, A_2 = \begin{bmatrix} a_2 \cos\phi_2 \\ a_2 \sin\phi_2 \\ h_2 \end{bmatrix}$$

where, a_i is the distance of the first joint of i^{th} leg from the origin of the base frame and ϕ_1 is the angle between the xy-projection of vector from the origin of the base frame to the joint and the x-axis. Similarly, ϕ_2 is the angle between the xy-projection of vector from the origin of the base frame to the joint and the y-axis. The joints of each leg are at height h_1 and h_2 respectively. The universal joint (U) in the motion constraint generator leg is given as $[0, 0, t]^T$ with respect to the base frame. The spherical joints in each leg are represented with respect to a frame with U as its origin and are given as:

$$B_1 = \begin{bmatrix} b_1 \cos\psi_1 \\ b_1 \sin\psi_1 \\ h_3 + t \end{bmatrix}, B_2 = \begin{bmatrix} b_2 \cos\psi_2 \\ b_2 \sin\psi_2 \\ h_4 + t \end{bmatrix}$$

where, b_i and ψ_i are used to express the spherical joints in the legs and have similar interpretation as that of a_i and ϕ_i. The joints of each leg are at height $h_3 + t$ and $h_4 + t$ respectively.
Thus, the mechanism can be parameterized by 13 parameters after assuming that the motion constraint generator lies on the z-axis of the base. The 13 mechanism parameters to be optimized, as shown in figure 6 and detailed above are: $[a_1, \phi_1, h_1, b_1, \psi_1, h_2, a_2, \phi_2, h_3, b_2, \psi_2, h_4, t]$. The optimization parameters and the constraints along with their range are shown in Table 3.

Parameters	Value	Parameters	Value
optimization dimension	13	Range of a_i	[0.25, 1.5]
Range of b_i	[0.25, 2]	Range of ϕ_i and ψ_i	[-1.745, 1.745]
Range of h_i	[-0.1, 0.1]	Range of t	[1, 4]
Number of starts	200	Number of iterations	10 and 20
Objective choice	Workspace, GCI, VAF	Velocity amplification range	[0.3, 3]
Range of b_i	[0.25, 2]	Range of ϕ_i and ψ_i	[-1.745, 1.745]
Workspace (in roll and pitch)	circle of radius 1	stroke ratio	1.5
limits on spherical joints	$\pm\pi/6 radians$	Collision constraint	considered

Table 3: The parameters set for the optimization of 2-dof RCM mechanism

The computational expense of optimizing this mechanism stems from the increase in optimization space, the number of degrees of freedom, and the considered workspace. A thorough examination of the regular dextrous workspace for the specified mechanism can be found in [52]. Results are contingent upon both the chosen objective and the reward strategy. Table 3 presents the outcomes obtained by optimizing the GCI and awarding valid points in the workspace with a value of 1 and invalid points with 0. The time necessary for evaluating one instance (a particular set of parameters) and the mean time for a single start (the full operation until the algorithm stops and returns locally optimized

parameters) are documented in Table 4. Further analysis was conducted to determine the effects of different objectives on the overall optimization time. The fine search process was found to be significantly more time-consuming compared to coarse searches, highlighting the algorithm's efficiency. Table 4 contains the results, and the computational time was measured using the same system, intended for comparison purposes only. The schematic plot of the mechanism optimized for maximum GCI, along with the heatmap for GCI evaluation using the optimized parameters, is shown in Figure 7. Likewise, Figure 8 displays the schematic and heatmap of the quality linked to the VAF for the associated optimized parameters. The schematics shown in both figures indicate that the optimized parameters gravitate towards an architecture with actuated legs separated by $\frac{\pi}{2}$ radians and aligned with the universal joint axes found in the motion constraint generator. This observation implies that human intuition and experience can be employed to decrease the optimization space's dimension, leading to accelerated optimization and more easily manufacturable designs.

Parameters	GCI	VAF
Time for 1 coarse evaluation	14 seconds	18.3 seconds
Time for single coarse start	291 seconds	347.5 seconds
Time for one fine evaluation	50.5 seconds	51 seconds
Time for single fine start	1072 seconds	1077 seconds
Best point		
$[a_1, \phi_1, h_1, b_1, \psi_1, h_2, a_2, \phi_2, h_3, b_2, \psi_2, h_4, t]$ (refer figure 6)	[1.13, -1.02, -0.06, 1.47, -1.01, -0.05, 0.72, 0.44, -0.02, 1.52, 0.54, 0.02, 3.04]	[0.68, -0.25, 0.08, 1.03, 0.1, 0.04, 0.25, -1, 0.01, 1.1, -1.45, 0.17, 2.4]
Best actuator range	[2.54, 3.8]	[2, 3]
evaluation	GCI	VAF
mean	0.79	0.48
standard deviation	0.18	0.29
maximum evaluation	1	0.99
configuration ($[\alpha, \beta]$)	[0.39, 0.13]	[0, 0.43]
minimum evaluation	0.318	-1.2
configuration ($[\alpha, \beta]$)	[0.86, 0.51]	[-0.99, 0.14]

Table 4: The results for the optimization of 2-dof RCM mechanism

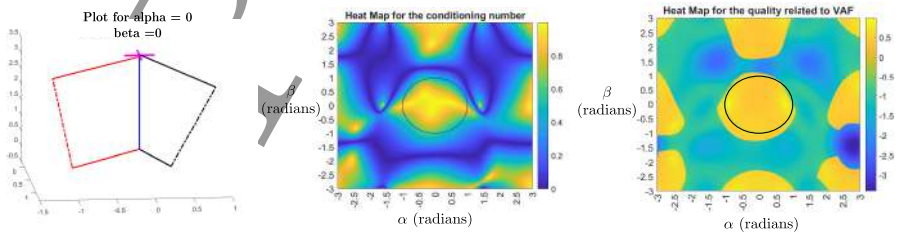


Figure 7: The schematic plot for the mechanism optimized for GCI and the heatmap for the evaluation. Calculation of GCI for this mechanism is discussed in [52]. The rightmost subfigure is the heatmap for the VAF quality corresponding to the same parameters.

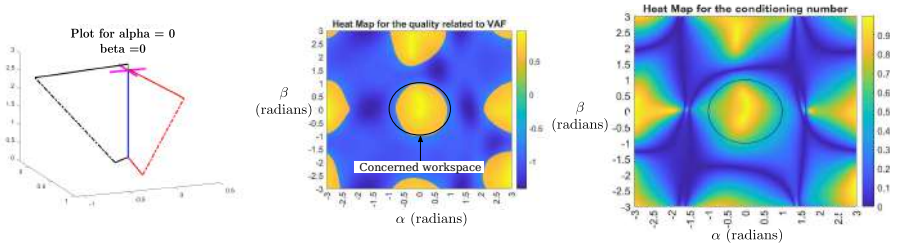


Figure 8: The schematic plot for the mechanism optimized for VAF and the heatmap for the evaluation. The rightmost subfigure is the heatmap for the GCI corresponding to the same parameters.

4.3. Dimension reduction

Human intuition can be implemented to further reduce the optimization space such that the hybrid series-parallel system can be optimized faster and in an efficient manner. The observations presented in the previous section confirms that basic analysis of the mechanism can greatly help in reducing the optimization space. In the case of 2UPS-1U, fixing the z-coordinate of the first universal joint in each leg as zero reduces 2 parameters(h_1, h_2). As we observed that the two legs are optimized when 90° apart, if we fix A_1 along x-axis and A_2 along y-axis, we further reduce two parameters (ϕ_1, ϕ_2). Similar process is used for B_1 and B_2 such that $h_3 = h_4 = h$, $\psi_1 = 0$, and $\psi_2 = 0$. In order to make the mechanism modular, the legs can be made symmetrical such that $a_1 = a_2 = a$, and $b_1 = b_2 = b$. This process reduces the 13 parameter space to only 4 dimensional space with a, b, h and t as the optimization parameters. Such reduction can help optimize mechanisms in cascade and also provide designs that are easier to manufacture and assemble thus adding practical advantages to the optimization.

5. Conclusions

In this chapter, we presented a novel optimization algorithm for parallel manipulators that is able to implement the joint limits and the collision of prismatic joints as constraints. The optimization methodology is also able to optimize the length of the actuator stroke, which enables the designer greater flexibility and clarity in the choice of the actuators. The Nelder-Mead algorithm uses geometric methods to search for a local optimum, which is relevant for mechanism optimization. The algorithm implements a two-step search by combining a faster local search Nelder-Mead algorithm with initial simplexes spread over all the parameter space and then uses a finer search by using the locally optimized points in the step 1. The algorithm is general and can adapt to any non-redundant parallel mechanisms with prismatic as well as revolute joint. The paper presents two different mechanism optimization as an example to present the flexibility of the algorithm. It is observed in the design of 2UPS-1U that the optimal solutions correspond to an orthogonal arrangements

of the legs. This confirms that human feedback and mechanism knowledge can be used to reduce the dimension of the search space.

When it comes to design optimization of series-parallel hybrid mechanisms, we have only scratched the surface of the problem. A holistic treatment of the design optimization problem for series-parallel hybrid robots would require dealing with a very large dimensional space of design variables for which one would require more computationally efficient optimization schemes. Additionally, the study presented in this chapter took into account only the kinematic properties of the mechanism. Including the dynamics into account during the co-design process is also a very important avenue for future work. Millions of years of biological evolution has led to the interesting muscle combinations that we witness in animals. It remains an open problem in the robotics community to develop co-design frameworks which are capable to produce robot designs which have similar athletic performance as their natural counterparts.

References

[1] B. Lee, C. Knabe, V. Orekhov, D. Hong, Design of a human-like range of motion hip joint for humanoid robots, in: Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Buffalo, New York, USA, 2014, pp. 8–18.

[2] S. Lohmeier, T. Buschmann, H. Ulbrich, F. Pfeiffer, Modular joint design for performance enhanced humanoid robot LOLA, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, IEEE, Orlando, FL, USA, 2006, pp. 88–93.

[3] D. Kuehn, M. Schilling, T. Stark, M. Zenzes, F. Kirchner, System Design and Testing of the Hominid Robot Charlie: System Design and Testing of the Hominid Robot Charlie, Journal of Field Robotics 34 (4) (2017) 666–703.

[4] S. Kumar, B. Bongardt, M. Simnofske, F. Kirchner, Design and kinematic analysis of the novel almost spherical parallel mechanism active ankle, Journal of Intelligent & Robotic Systems 94 (2) (2018) 303–325.

[5] S. Kumar, H. Wöhrle, M. Trampler, M. Simnofske, H. Peters, M. Mallwitz, E. A. Kirchner, F. Kirchner, Modular design and decentralized control of the recupera exoskeleton for stroke rehabilitation, Applied Sciences 9 (4) (2019).

[6] J. Arata, H. Kondo, M. Sakaguchi, H. Fujimoto, A haptic device delta-4: Kinematics and its analysis, in: Proceedings of World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Salt Lake City, UT, USA, 2009, pp. 452–457.

[7] G. Michel, D. Salunkhe, D. Chablat, P. Bordure, A new RCM mechanism for an ear and facial surgical application, in: Proceedings of Advances in Service and Industrial Robotics, Springer International Publishing, Poitiers, France, 2020, pp. 408–418.

[8] A. Dutta, D. H. Salunkhe, S. Kumar, A. D. Udai, S. V. Shah, Sensorless full body active compliance in a 6 DOF parallel manipulator, Robotics and Computer-Integrated Manufacturing 59 (2019) 278–290.

[9] A. D. Udai, S. K. Saha, A. Dayal, Overlaid Orthogonal Force Oscillations for Robot Assisted Localization and Assembly, ISME Journal of Mechanics and Design 2 (1) (2018) 9–25.

[10] S. Kumar, H. Wöhrle, J. de Gea Fernández, A. Müller, F. Kirchner, A survey on modularity and distributivity in series-parallel hybrid robots, Journal of Mechatronics 68 (2020).

[11] J. Brinker, N. Funk, P. Ingenlath, Y. Takeda, B. Corves, Comparative Study of Serial-Parallel Delta Robots With Full Orientation Capabilities, IEEE Robotics and Automation Letters 2 (2) (2017) 920–926.

[12] S. Caro, D. Chablat, R. Ur-Rehman, P. Wenger, Multiobjective design optimization of 3–PRR planar parallel manipulators, in: Proceedings of 20^{th} CIRP Design Conference, Nantes, France, 2010, pp. 373–383.

[13] Z. Ma, A.-N. Poo, M. H. Ang, G.-S. Hong, H.-H. See, Design and control of an end-effector for industrial finishing applications, Robotics and Computer-Integrated Manufacturing 53 (2018) 240–253.

[14] P. Wenger, D. Chablat, Kinematic analysis of a new parallel machine tool: the Orthoglide, in: Proceedings of Advances in Robot Kinematics, Slovenia, 2000, pp. 1–11.

[15] C. Gosselin, J. Angeles, A global performance index for the kinematic optimization of robotic manipulators, Journal of Mechanical Design 113 (3) (1991) 220–226.

[16] S. Chiu, Kinematic characterization of manipulators: an approach to defining optimality, in: Proceedings. 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 1988, pp. 828–833.

[17] D. Chablat, P. Wenger, F. Majou, The Optimal Design of Three Degree-of-Freedom Parallel Mechanisms for Machining Applications, in: In the Proceedings of 11th International Conference on Advanced Robotics, 2003, Coimbra, Portugal, 2003, pp. 1–6.

[18] D. Chablat, P. Wenger, F. Majou, J.-P. Merlet, A novel method for the design of 2-dof parallel mechanisms for machining applications, International Journal of Robotics Research 23 (6) (2007) 615–624.

[19] R. A. Srivatsan, S. Bandyopadhyay, Determination of the safe working zone of a parallel manipulator, in: Proceedings of Computational Kinematics, Dordrecht, Netherlands, 2014, pp. 201–208.

[20] C. Germain, S. Caro, S. Briot, P. Wenger, Optimal design of the IRSBot-2 based on an optimized test trajectory, in: Proceedings of 37th Mechanisms and Robotics Conference, American Society of Mechanical Engineers, Portland, Oregon, USA, 2013, pp. 1–11.

[21] M. H. Saadatzi, M. T. Masouleh, H. D. Taghirad, C. Gosselin, M. Teshnehlab, Multi-objective scale independent optimization of 3-RPR parallel mechanisms, in: Proceedings of 13th World Congress in Mechanism and Machine Science, Guanajuato, Mexico, 2011, pp. 1–11.

[22] M. Gallant, R. Boudreau, The synthesis of planar parallel manipulators with prismatic joints for an optimal, singularity-free workspace, Journal of Robotic Systems 19 (1) (Jan. 2002).

[23] S. Caro, D. Chablat, A. Goldsztejn, D. Ishii, C. Jermann, A branch and prune algorithm for the computation of generalized aspects of parallel robots, in: Principles and Practice of Constraint Programming, Berlin, Heidelberg, 2012, pp. 867–882.

[24] S. Kucuk, A dexterity comparison for 3-DOF planar parallel manipulators with two kinematic chains using genetic algorithms, Mechatronics 19 (6) (2009) 868–877.

[25] S. Ganesh, A. Koteswara Rao, B. Sarath kumar, Design optimization of a 3-DOF star triangle manipulator for machining applications, Materials Today: Proceedings 22 (12) (2020) 1845–1852.

[26] V. Muralidharan, A. Bose, K. Chatra, S. Bandyopadhyay, Methods for dimensional design of parallel manipulators for optimal dynamic performance over a given safe working zone, Mechanism and Machine Theory 147 (2020) 103721.

[27] S. Ha, S. Coros, A. Alspach, J. Kim, K. Yamane, Computational co-optimization of design parameters and motion trajectories for robotic systems, The International Journal of Robotics Research 37 (13-14) (2018) 1521–1536.

[28] N. Durand, J.-M. Alliot, A combined nelder-mead simplex and genetic algorithm, in: Proceedings of GECCO 1999, Genetic and Evolutionary Computation Conference, Orlando, FL, USA, 1999, pp. 1–7.

[29] A. Koscianski, M. Luersen, Globalization and parallelization of Nelder-Mead and Powell optimization methods, in: Proceedings of Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering, Dordrecht, Netherlands, 2008, pp. 93–98.

[30] M. A. Luersen, R. Le Riche, Globalized Nelder–Mead method for engineering optimization, Computers & Structures 82 (23-26) (2004) 2251–2260.

[31] P. Niegodajew, M. Marek, W. Elsner, Ł. Kowalczyk, Power plant optimisation—effective use of the Nelder-Mead approach, Processes 8 (3) (2020) 357.

[32] K. H. Hunt, Review: don't cross-thread the screw!, Journal of Robotic Systems 20 (7) (2003) 317–339.

[33] D. H. Salunkhe, G. Michel, S. Kumar, M. Sanguineti, D. Chablat, An efficient combined local and global search strategy for optimization of parallel kinematic mechanisms with joint limits and collision constraints, Mechanism and Machine Theory 173 (2022) 104796.

[34] J. A. Nelder, R. Mead, A simplex method for function minimization, The Computer Journal 7 (4) (1965) 308–313.

[35] J. C. Lagarias, J. A. Reeds, M. H. Wright, P. E. Wright, Convergence properties of the Nelder–Mead simplex method in low dimensions, SIAM Journal on Optimization 9 (1998) 7.

[36] K. I. M. McKinnon, Convergence of the Nelder–Mead simplex method to a nonstationary point, SIAM Journal on Optimization 9 (1) (1998) 148–158.

[37] P. C. Wang, T. E. Shoup, Parameter sensitivity study of the Nelder–Mead Simplex Method, Advances in Engineering Software 42 (7) (2011) 529–533.

[38] D. Byatt, Convergent variants of the Nelder-Mead algorithm.pdf, Ph.D. thesis, University of Canterbury, England (2000).

[39] S. Zapotecas Martínez, C. A. Coello Coello, A proposal to hybridize multiobjective evolutionary algorithms with non-gradient mathematical programming techniques, in: Proceedings of Parallel Problem Solving from Nature – PPSN X, Berlin, Heidelberg, 2008, pp. 837–846.

[40] H. Niederreiter, Random number generation and quasi-monte carlo methods, in: Proceedings of CBMS-NSF regional conference series in applied mathematics, Vol. 63, Philadelphia, Pennsylvania, 1992, pp. 1 – 243.

[41] K.-T. Fang, Y. Wang, Number-Theoretic Methods in Statistics, Chapman & Hall, London, 1994.

[42] A. Alessandri, C. Cervellera, D. Macciò, M. Sanguineti, Optimization based on quasi-Monte Carlo sampling to design state estimators for nonlinear systems, Journal of Optimization 59 (2010) 963–984.

[43] J. M. Hammersley, D. C. Handscomb, Monte Carlo Methods, Methuen, London, 1964.

[44] A. Alessandri, C. Cervellera, M. Sanguineti, Design of asymptotic estimators: an approach based on neural networks and nonlinear programming, IEEE Trans. on Neural Networks 18 (1) (2007) 96–96.

[45] I. M. Sobol', The distribution of points in a cube and the approximate evaluation of integrals, Zh. Vychisl. Mat. i Mat. Fiz. 7 (1967) 784–802.

[46] S. Bartsch, M. Manz, P. Kampmann, A. Dettmann, H. Hanff, M. Langosz, K. v. Szadkowski, J. Hilljegerdes, M. Simnofske, P. Kloss, M. Meder, F. Kirchner, Development and control of the multi-legged robot mantis, in: Proceedings of ISR 2016: 47^{th} International Symposium on Robotics, Munich, Germany, 2016, pp. 1–8.

[47] J. Esser, S. Kumar, H. Peters, V. Bargsten, J. d. G. Fernandez, C. Mastalli, O. Stasse, F. Kirchner, Design, analysis and control of the series-parallel hybrid rh5 humanoid robot, in: 2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), 2021, Munich, Germany, 2021, pp. 400–407.

[48] S. Kumar, Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots, Ph.D. thesis, University of Bremen, DFKI-RIC, Bremen, Germany (Sep. 2019).

[49] S. Kumar, A. Nayak, H. Peters, C. Schulz, A. Müller, F. Kirchner, Kinematic analysis of a novel parallel 2sprr+1u ankle mechanism in humanoid robot, in: Proceedings of Advances in Robot Kinematics, Bologna, Italy, 2018, pp. 431–439.

[50] C. Stoeffler, S. Kumar, H. Peters, O. Brüls, A. Müller, F. Kirchner, Conceptual design of a variable stiffness mechanism in a humanoid ankle using parallel redundant actuation, in: Proceedings of 18th International Conference on Humanoid Robots (Humanoids), Beijing, China, 2018, pp. 462–468.

[51] M. Boukheddimi, S. Kumar, H. Peters, D. Mronga, R. Budhiraja, F. Kirchner, Introducing rh5 manus: A powerful humanoid upper body design for dynamic movements, in: 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 01–07.

[52] D. Chablat, G. Michel, P. Bordure, S. Venkateswaran, R. Jha, Workspace analysis in the design parameter space of a 2-DOF spherical parallel mechanism for a prescribed workspace: Application to the otologic surgery, Mechanism and Machine Theory 157 (Mar. 2021).