Strategies for mechanical metamaterial design

Singh, N.

Chapter 3

Rational Design of Flexible Yet Generic 2D Mechanical Metamaterials

Abstract – In this chapter, we describe a nature-inspired algorithm based optimization methodology to design flexible yet generic 2D mechanical metamaterials. The procedure is based on optimizing the design of the underlying idealized hinging mechanism such that a target curve that characterizes the internal motion is achieved. To perform this, we utilize *Particle Swarm Optimization (PSO)* [60]. Through hyperparameter optimization for the governing PSO parameters, i.e. ω - *inertial weight*, c_1 - *cognition parameter* and c_2 - *social parameter*, we will demonstrate that the search behavior of PSO gets strongly influenced by these parametric setting. We will locate the 'sweet-spot' in the parameter space and establish that low values of c_1 and high values of c_2 are conducive for our particular problem, while ω plays a minor role. To verify the role of parametric setting, we adopt *Sammon's Mapping* as a technique to reduce dimensionality and visualize the search process. We further show that these parametric settings affect the statistical distribution type of the final solution. We show PSO generated a vast number of unique structures whose hinging motion is very close to that of a single degree-of-freedom mechanism, while their spatial structure shows that they are far from true single degree-of-freedom mechanisms. We characterize PSO by probing the search from a statistical point of view and distinguish solutions corresponding to true local

minima from those that are not. Finally, we introduce a method to obtain regular 2D tesselations that shadows the deformation pattern of its constituting unit cell. We bring these computer-designed structures to real life through 3D printing and retrieve ideal deformation patterns experimentally.

3.1 Introduction

In several cases, the deformation behavior of 2D mechanical metamaterials can be mapped onto hinging structures consisting of rotating rigid units connected together through *pin-joints* (hinge-joints). Such structures can consist of polygons or rods or a combination of both [18,38–41], . Well known examples include a rich variety of systems describing the observed *auxetic* (*Poisson's ratio*, $\nu < 0$) behavior. For example, the *NPR polyurethane foam* with re-entrant structure reported by Lakes et al. that had a negative value of ν = -0.70, can be idealized to a hinging/flexing re-entrant honeycomb structure [Fig. 3.1] [79]. J.N. Grima et al. and others identified numerous single degree-of-freedom mechanisms that show auxetic behavior emerging from rotating rigid units, eg. triangles, squares, different-sized squares, or rectangles [Fig. 3.2 (a-b)] [40, 80–82]. Al Grant examined which 'irregular' polygons can render fully hinging tessellation and showed that similar mechanisms can be constructed through trapezoids and cyclic quadrilaterals by connecting them together in a special manner, [Fig. 3.2 (c-d)] [83]. In a more recent work, A. Rafsanjani reported a class of reconfigurable 2D materials exhibiting simultaneous auxeticity and structural bistability, the deformation pattern of which can essentially be mapped onto hinging mechanisms [36].

In fact, the main scheme behind the design of these *mechanism-based metamaterials* is to base the desired deformation mode of the unit cell upon the zero-energy, free motion mode of an underlying idealized mechanism [38]. Suitable arrangements of such unit cells enable to achieve highly nonlinear bulk behavior [37]. For the *low-energy deformations* of these materials to closely follow the zero-energy motion of the underlying mechanism, hinge geometry plays a critical role, the elastic-distortions introduced by which are ideally minimized by connecting stiffer elements with slender, flexible connectors [38, 39].

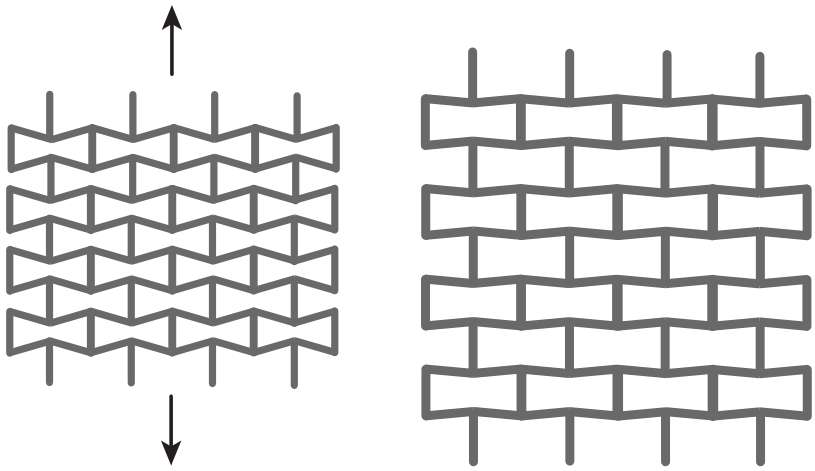


Figure 3.1: (a) Contracted, and (b) expanded states of a re-entrant honeycomb structure which deforms predominantly via the hinging mechanism of the diagonal ribs. Re-entrant honeycomb is a typical example among structures exhibiting negative Poisson's ratio, ν , also known as auxetics [79].

Rational design – Mechanism-based metamaterials are prevalent - careful geometrical design of the constituting rigid elements of the mechanism allows to achieve arbitrary complex internal deformations [41]. Concurrently, this also opens up the opportunities to come up with rational strategies to reversely design for a desired target behavior. Surprisingly, little attention has been paid towards this. Important contributions include [84–86]. Still, majority of the earlier reported examples have been designed through intuition or trial based strategies; thus the mechanical properties emerge as a result of the design. This explains why many of the examples have a periodically repeating subunit element. However, aperiodic structures lying hidden in the design space can be discovered by an optimization-based strategy to design a structure such that it meets a certain target criteria. Following this approach, the focus of this chapter is to design generic flexible 2D mechanical metamaterials consisting of hinging blocks, when a target functionality characterizing the internal deformation of the underlying mechanism consisting of pin-jointed quadrilaterals is desired.

It is important to note that generic pin-jointed quadrilaterals can never

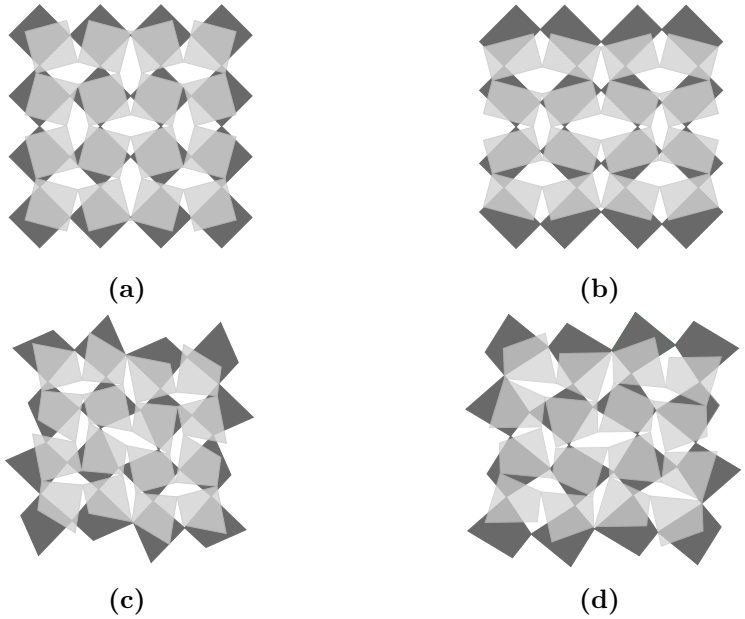


Figure 3.2: Single degree-of-freedom (DOF) mechanisms consisting of pin-jointed (a) squares, (b) rectangles, (c) trapezoids and (d) cyclic quadrilaterals. These mechanisms exhibit scale-independent auxetic behavior. The neutral states are shown in the color dark gray and the deformed states are shown in the color light gray [40, 80, 83].

form *exact mechanisms*. We show this through a counting argument on a limiting case in the following discussion and provide a formal proof in §3.2. There we also demonstrate the possibility that (fortunately) such systems can at best form *approximate mechanisms* i.e. near-perfect mechanisms [35]. From an optimization point of view, it means that these systems correspond to approximate solutions situated at deep local minima in the *objective function landscape* (assuming a minimization problem at hand), and the global minima are occupied by the exact mechanisms. This is not entirely bad considering: *(i)* the tendency of metaheuristics to get trapped inside the deep local minima, and *(ii)* that the 'good-quality' approximate solutions serve well for practical purposes.

Maxwell's counting rule – Before sketching the design process, we will first present the possibility for the existence of generic approximate mechanisms consisting of pin-jointed quadrilateral polygons. Maxwell introduced a simple rule for calculating the *degree(s)-of-freedom (DOF)* of networks consisting of bars and pin-joints according to which a system having n rigid bodies connected through j joints will have $3n - 2j - 3$ *non-trivial DOF* i.e. excluding the global translational and rotational DOF [87]. The explanation is as follows: each rigid body independently can have 3 DOF i.e. translation in x and y directions and a global rotation. Thus, n rigid bodies should have $3n$ DOF. Coupling these bodies through pin-joints brings constraints into the system with each joint removing 2 DOF [1], leading to a total count of $3n - 2j - 3$ non-trivial DOF.

We utilize this counting argument now. We begin with a closed network formed by four arbitrarily shaped pin-jointed polygons [Fig. 3.3(a), color dark gray]. Maxwell's formula predicts that the network has 1 non-trivial DOF, which corresponds to the internal deformation mode. Essentially the polygons form the boundary of a four-bar linkage, which is the simplest movable closed-chain linkage. The system in its deformed state is shown in light gray. Through the same counting argument, one can confirm that the networks shown in Fig. 3.3(b,c) also have 1 non-trivial DOF, thereby forming a mechanism [2]. However, applying Maxwell's formula to a 3x3 network [Fig. 3.3 (d)] leads to a 0 non-trivial DOF. This implies that generally 3x3 networks do not contain any internal mechanism and are instead rigid, monostable systems.

States of self-stress and isostaticity – However, Maxwell's formula does not takes into account the geometry of the system - non-generic examples, such as the 3x3 network sections of Fig. 3.2 contain an internal mechanism! The counting of redundant constraints in Maxwell's formula causes this discrepancy. In order to correctly calculate the non-trivial DOF for systems consisting of possible redundant constraints, Calladine extended Maxwell's rule as: number of non-trivial DOF $= 3n - 2j - 3 + n_{ss}$, where n_{ss}

[1] Imagine a system where two rigid polygons are connected through a pin-joint. If for the first polygon, its position in terms of x and y coordinates and rotational information is provided, then we only require the rotational information of the second polygon to specify the full configuration of the system.

[2] Although to what degree does the mechanism deforms internally cannot be answered immediately.

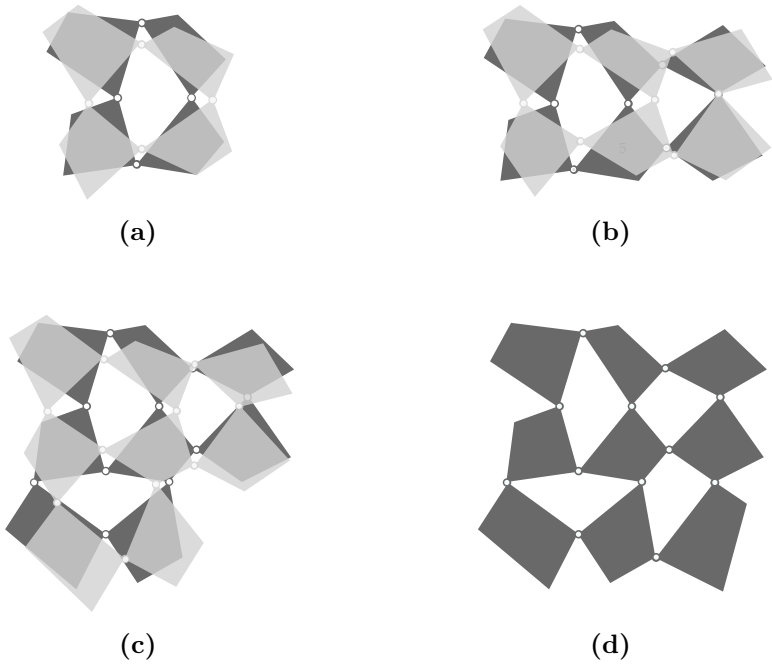


Figure 3.3: Networks of pin-jointed generic polygons. Counting of the non-trivial degree(s) of freedom (DOF) by the Maxwell's rule [87] reveals that (a)-(c) has 1 DOF and (d) has 0 DOF. Such generic 3x3 networks cannot form perfect mechanisms, however special geometric configurations may allow to form *approximate mechanisms*.

is the total number of *states of self-stress* [88]. Non-generic 3x3 networks contain one internal non-trivial DOF. Hence, for these systems $n_{ss} = 1$. On the other hand, generic 3x3 networks generally have zero non-trivial DOF and have therefore $n_{ss} = 0$. Such marginal systems that do not contain any internal zero mode nor have any states of self-stress are called *isostatic systems* [11, 89].

> Still, if the polygonal shape is such that one out of the total 24 constraints is 'almost' redundant, the possibility of generic 3x3 networks forming approximate mechanisms exists.

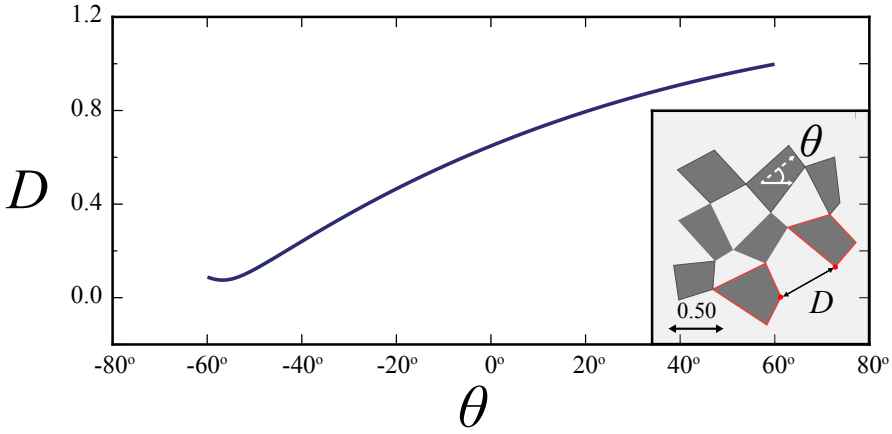


Figure 3.4: The $D(\theta)$ curve. A characteristic curve that captures the internal deformation of the eight-polygon mechanism shown in the inset figure. D (labeled) records the distance between the 'open vertices' of the specific red-bordered polygons and θ records the amount of internal deformation of the mechanism. The method of measuring θ is explained in §3.4.1 and for now it suffices to know θ as a parameter that measures the amount of internal deformation.

***D*(*θ*) curve** – We focus on the network shown in Fig. 3.3(c) that has an internal mechanism. Once the geometric information of each individual polygon is specified, only one independent variable is required, in terms of which the complete configuration of the system can be described. We use the parameter θ shown in Fig. 3.4(inset) as a measure of how much the *eight-polygon mechanism* has deformed internally[3] and plot the distance D between the 'open' unconnected corners as a function of θ. We will refer to this characteristic plot as $D(\theta)$ curve or $D(\theta)$ function. The $D(\theta)$ curve of one such example is shown in Fig. 3.4. The shape of individual polygons determines the trajectory of $D(\theta)$ curve, which can result in different possibilities if the 3x3 network is completed:

(i) Monotonic $D(\theta)$ curves — A rigid system: a monotonic $D(\theta)$ curve

[3] θ is more clearly defined in §3.4.1 but for now it suffices to know θ as a parameter that measures the amount of internal deformation.

implies that for every value of θ, there is a unique value of D. Therefore, with the connecting edge length of the ninth polygon within the given range of D, there is only one configuration of the 3x3 network that can be realized.

(ii) Quasi-constant and constant $D(\theta)$ curves — An approximate mechanism : a $D(\theta)$ curve with a *quasi-constant* value of D corresponds to an approximate 3x3 mechanism when the ninth polygon has edge length equal to the mean value of D. With suitable material type and hinge geometry, mechanical metamaterials based on these approximate mechanisms can feature a single low-energy deformation mode. In the current chapter, we focus on this case. Strictly constant $D(\theta)$ curve would lead to exact 3x3 mechanisms.

(iii) Parabolic (or more complex) $D(\theta)$ curves — A bi(multi)-stable 3x3 system consisting of stiffer elements connected with flexible hinges is possible if the $D(\theta)$ curve has same values of D for two (multiple) values of θ. We explore this case in the next chapter.

The $D(\theta)$ curve of the eight-polygon mechanism serves as a functionality to measure the flexibility of the 3x3 networks. The question of designing flexible 3x3 networks can thus be framed as a typical design optimization problem: optimize the design parameters which specify the polygonal shapes such that a specified constant target curve $D_t(\theta)$ is achieved to an arbitrarily high degree of approximation. Classical techniques, mainly based on gradient-finding methods, are likely to fail in such optimization problems concerning high dimensionality [56]. A much more advantageous choice in such cases are population based evolutionary search heuristics such as genetic algorithms, genetic programming, evolutionary strategies, swarm algorithms and other nature-inspired algorithms which rely on the principles of selection (competition) and reproduction (cooperation) among the population members [57–62]. In the current work, we implement one such algorithm inspired by the food searching capability of a bird flock: *Particle Swarm Optimization (PSO)* [60, 90, 91].

The outline of this chapter is as follows. In the next section, §3.2, we describe the ideal *loop condition* which allows a 3x3 network of polygons to form a perfect mechanism; forming the mathematical basis for our search. We describe the numerical tool to obtain discretized $D(\theta)$ curve of an arbitrary eight-polygon mechanism in §3.3. We lay down the entire

machinery for optimization-based design methodology in §3.4-§3.6, which includes the description of PSO, its optimal parameter settings and their effect on the distribution type of final solutions. We utilize Sammon's mapping to visualize the search process in §3.7 and gain understandings about the effect of PSO parameters on the search behavior. We show the actual optimized structures and characterize the search exploration of PSO in §3.8. Finally, in §3.9, we show the 3D printed samples and suggest a method to tile the flexible unit cells into periodic tessellations - the *metatilings*.

3.2 Mathematical Loop Condition

In the previous section, we learned that generically the 3x3 networks are rigid or monostable, although special polygonal shapes can possibly exist for which the network approaches to a mechanism. In this section, we will establish an ideal *loop condition*, the proximity of whose fulfillment determines how close a 3x3 network can get to form a mechanism.

Transmission function — Consider a (hypothetical) 3x3 mechanism shown in Fig. 3.5. The network consists of 'internally connected' four four-bar linkages, $\Lambda_1, \Lambda_2, \Lambda_3$ and Λ_4. In order to analyze this system, we will begin with a single constituent four-bar linkage and relate its input and output angles in terms of its bar lengths.

Consider a planar four-bar linkage O_AABO_B show in Fig. 3.6. The linkage consists of four pin-connected rigid bars with bar lengths a_1, a_2, a_3 and a_4 (grounded). The input bar a_1 makes an angle α, while the output bar makes an angle β with the grounded bar a_4. In terms of the geometrical parameters a_1, a_2, a_3 and a_4, our aim would be to relate the input angle α with the output angle β of the linkage. Consider the rectangular coordinate system O_Axy with respect to which the x and y coordinates of A - (x_2, y_2) and B - (x_3, y_3) may be written as follows:

For A: $\quad x_2 = a_1 \cos\alpha; \quad y_2 = a_1 \sin\alpha \ ,$

For B: $\quad x_3 = -a_4 + a_3 \cos\beta; \quad y_3 = a_3 \sin\beta.$

Since the distance AB is fixed and is equal to a_2, application of Pythagoras'

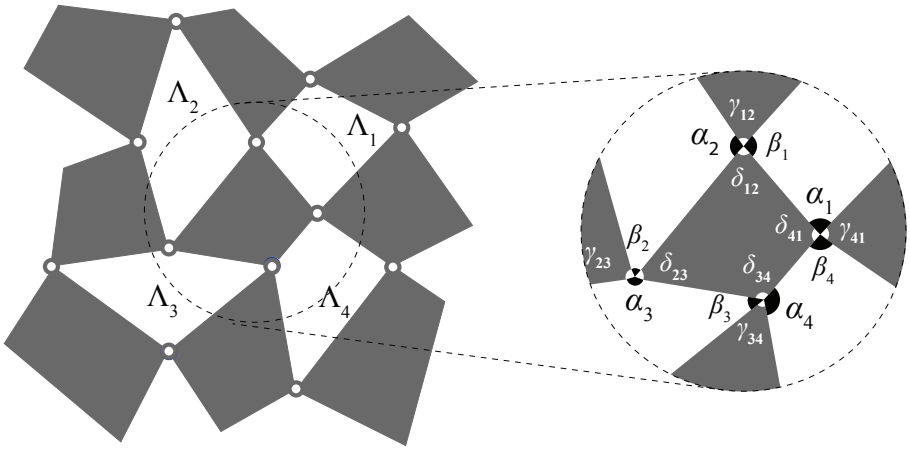


Figure 3.5: A (hypothetical) 3x3 mechanism consisting of four internal four-bar linkages: $\Lambda_1, \Lambda_2, \Lambda_3$ and Λ_4. For a linkage Λ_i, the input and the output angles are labeled as α_i and β_i respectively. $\gamma_{i,i+1}$ and $\delta_{i,i+1}$ are the fixed vertex angles that conjugate the linkages Λ_i and Λ_{i+1}. Subscripts follow a cyclic order.

theorem yields:

$$(x_2 - x_3)^2 + (y_2 - y_3)^2 = a_2^2 \,. \tag{3.1}$$

Substituting the values of x_2, y_2, x_3 and y_3 in Eq. (3.1) gives:

$$(a_1 \cos\alpha + a_4 - a_3 \cos\beta)^2 + (a_1 \sin\alpha - a_3 \sin\beta)^2 = a_2^2 \,. \tag{3.2}$$

After trigonometric simplifications, the above equation may be written as:

$$A \sin\beta + B \cos\beta = C, \tag{3.3}$$

where:

$$A = \sin\alpha, \qquad B = \frac{a_4}{a_1} - \cos\alpha, \qquad C = -\frac{a_4}{a_3}\cos\alpha + \frac{a_1^2 - a_2^2 + a_3^2 + a_4^2}{2a_1a_2} \,.$$

Expressing $\sin\beta$ and $\cos\beta$ in terms of $\tan(\beta/2)$:

$$\sin\beta = \frac{2\tan(\beta/2)}{1+\tan^2(\beta/2)}, \qquad \cos\beta = \frac{1-\tan^2(\beta/2)}{1+\tan^2(\beta/2)} \,.$$

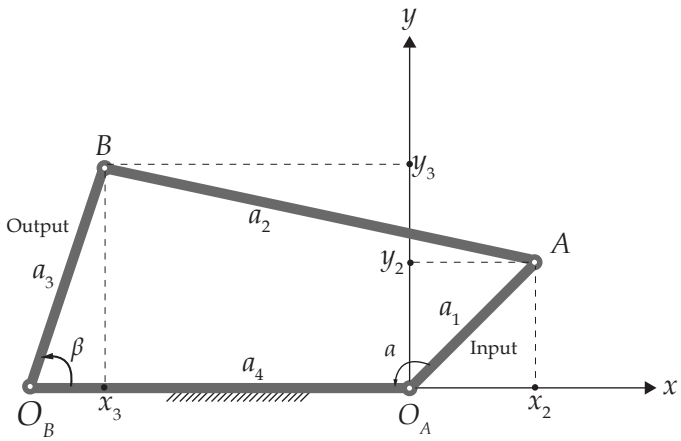


Figure 3.6: A planar four-bar linkage. Bar lengths are labeled as a_1, a_2, a_3 and a_4 (grounded). Input arm a_1 makes an angle α, and the output arm makes an angle β with the grounded bar a_4.

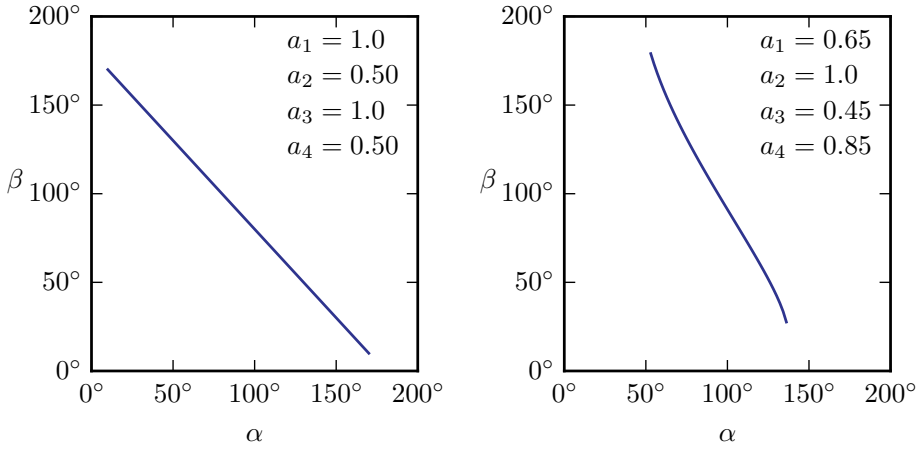
Substituting these values in Eq. 3.3 and solving for the solutions of the resulting quadratic equation, β can be found explicitly as a function of α and the parameters $a_1, a_2, a_3,\ a_4$.

$$\tan(\beta/2) \;=\; \frac{A \;\pm\; \sqrt{A^2 \;+\; B^2 \;+\; C^2}}{B \;+\; C}\,. \tag{3.4}$$

Thus, for one single value of α, two distinct solutions can be obtained which correspond to the two configurations in which a four bar linkage can exist. These two configurations are: *(i)* non self-intersecting, and *(ii)* self-intersecting. Ignoring the latter case, the relevant solution is:

$$\beta = 2\tan^{-1}\left(\frac{A \;+\; \sqrt{A^2 \;+\; B^2 \;+\; C^2}}{B \;+\; C}\right). \tag{3.5}$$

Relating the input angle α with the output angle β, Eq. 3.5 represents the *transmission function* of a four-bar linkage. The transmission functions of a parallelogram and a generic four-bar linkages are shown in Fig. 3.7.



(a) **(b)**

Figure 3.7: The transmission function for (a) a parallelogram four-bar linkage is linear with slope -1, i.e. $\beta = \pi - \alpha$, and (b) a generic four-bar linkage is nonlinear. Bar lengths a_1, a_2, a_3 and a_4 are labeled inside the figures.

Loop condition – We will now utilize the transmission function to derive the loop condition to form a 3x3 mechanism. We refer back to the network shown in Fig. 3.5. For the linkage Λ_1, α_1 can be mapped to β_1 using the transmission function given by Eq. 3.5. Since,

$$\alpha_2 = 2\pi - \gamma_{12} - \delta_{12} - \beta_1 \,, \tag{3.6}$$

we also have a mapping between α_1 to α_2. Let's denote this mapping function that relates the input angle α_i of linkage Λ_i with the input angle α_{i+1} of linkage Λ_{i+1} (cyclic permutations understood) by $\boldsymbol{M}_{i,i+1}$, $i \in \{1,2,3,4\}$,

$$\alpha_2 = \boldsymbol{M}_{1,2}(\alpha_1) \,. \tag{3.7}$$

Extending the same argument to the remaining adjacent pair of linkages:

$$\alpha_3 = \boldsymbol{M}_{2,3}(\alpha_2) \quad \text{or} \quad \alpha_3 = \boldsymbol{M}_{2,3}(\boldsymbol{M}_{1,2}(\alpha_1)) \,, \tag{3.8}$$

$$\alpha_4 = \boldsymbol{M}_{3,4}(\alpha_3) \quad \text{or} \quad \alpha_4 = \boldsymbol{M}_{3,4}(\boldsymbol{M}_{2,3}(\boldsymbol{M}_{1,2}(\alpha_1))) \,. \tag{3.9}$$

The final circularly transported angle $\bar{\alpha_1}$ is given by:

$$\bar{\alpha_1} = \boldsymbol{M}_{4,1}(\alpha_4) \quad \text{or} \quad \bar{\alpha_1} = \boldsymbol{M}(\alpha_1), \tag{3.10}$$

where $\boldsymbol{M} = \boldsymbol{M}_{4,1}(\boldsymbol{M}_{3,4}(\boldsymbol{M}_{2,3}(\boldsymbol{M}_{1,2})))$

For an exact 3x3 mechanism, the mapping function $\boldsymbol{M}$ over α_1 should be an identity function. This precisely happens when every four-bar linkage Λ_i in Fig. 3.5 is a parallelogram linkage and the graph between β_i and α_i is linear with slope -1 $i \in \{1,2,3,4\}$ [Fig. 3.7(a)].

The non-linear input-output response of every linkage Λ_i in some generic 3x3 network may so 'adjust' themselves such that the final mapping function $\boldsymbol{M}$ is extremely close to an identity function, leading to an approximate mechanism and indeed this is the mathematical motivation behind the search for optimal design of perfect or near-perfect mechanisms. Thus, by generic flexible systems, we refer to those systems where all the internal four-bar linkages are not simultaneously parallelograms.

> Throughout the rest of the text, we mainly concern ourselves with systems of eight pin-jointed polygons forming a mechanism and therefore simply use the word *mechanism* while referring to these systems.

3.3 Numerical Model

In this section, we present the precise implementation of a numerical model based on the principle of energy minimization that is suitable to obtain the $D(\theta)$ curve of an arbitrary mechanism. Mathematically, the problem is to connect all eight polygons $(P_1, P_2.....P_8)$ [Fig. 3.8] precisely at their corners for every θ configuration. To turn this into a numerically tractable optimization problem, we connect the polygons that share a connection with zero rest length linear springs. This ensures that zero energy states correspond to the configurations where the polygons are precisely connected at the corner tips. With this understanding, the problem is now reduced to finding the zero energy states of the spring-connected polygon system for different θ configurations.

We start with initial position of each polygon in the 2D plane. From this high energy state, the springs are allowed to relax which can result in either of the two scenarios: *(i)* the system relaxes to a numerically zero energy value implying that such a system forms a mechanism, *(ii)* or else, the system

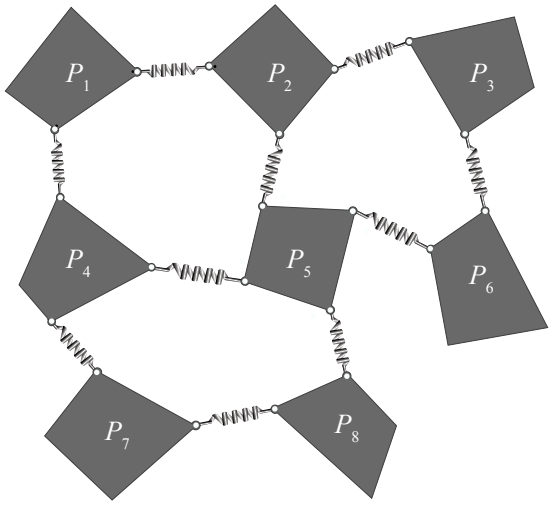


Figure 3.8: A system of eight polygons (labeled $P_1, P_2 ... P_8$) connected with linear springs of zero rest length in such a manner that if able to relax to a zero energy state, we end up with a system of eight polygons (similar to Fig. 3.3(c)) that are precisely connected at their corners.

relaxes to a non-zero finite energy value implying that such a system cannot form a mechanism. Given the total elastic energy stored in the springs as a function of position and orientation of each polygon, gradient based methods like *Steepest Descent*, *Conjugate Gradient (CG)*, *LBFGS* etc. can be employed to solve for the stable configuration of the system [92, 93]. We use a nonlinear CG method, specifically the version suggested by Flethcher-Reeves with Newton Raphson line search modification [94]. The details of the numerical model are presented in the following subsections.

3.3.1 Energy Functional

The position and orientation of a polygon are specified through three independent parameters:
- x: x-coordinate of the centroid of the polygon.
- y: y-coordinate of the centroid of the polygon.
- ϕ: orientation of the polygon measured *w.r.t.* the positive x-axis.

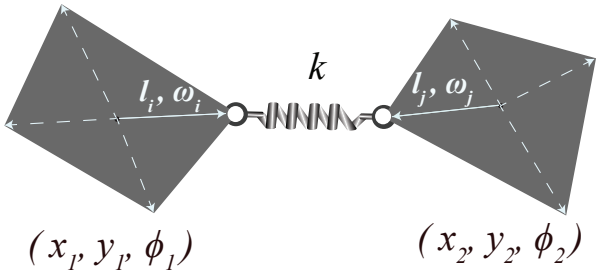


Figure 3.9: Two polygons sharing a spring connection k of zero rest length. (x, y, ϕ) denotes the centroid position and orientation of a polygon *w.r.t.* the positive x-axis. i^{th} corner of a polygon is parameterized through a vector of length l_i emanating from the centroid and making an angle ω_i *w.r.t.* the positive x-axis. The i^{th} corner of the first polygon is connected to the j^{th} corner of the second polygon.

Shape parameterization – The shape of a polygon is completely parameterized by four vectors emerging from the centroid. The polygon is constructed by joining the outer tip of these vectors. Each of the four vectors have a length l_i and oriented at ω_i w.r.t. the positive x-axis, $i \in \{1, 2, 3, 4\}$. In Fig. 3.9, we show two polygons sharing a connection k. The position and orientation of the first polygon are defined by the tuple (x_1, y_1, ϕ_1) while that of the second polygon are defined by (x_2, y_2, ϕ_2). The i^{th} corners of the first polygon is connected to the j^{th} corner of the second polygon. The corners involved in the connection belong to the tip of the vectors (l_i, ω_i) and (l_j, ω_j) of the respective polygons. Elastic energy stored in the linear spring k can be expressed as:

$$E_k = ((x_1 + l_i \cos(\phi_1 + \omega_i)) - (x_2 + l_j \cos(\phi_2 + \omega_j)))^2 +$$

$$((y_1 + l_i \sin(\phi_1 + \omega_i)) - (y_2 + l_j \sin(\phi_2 + \omega_j)))^2 \tag{3.11}$$

Given the position, orientation and shape information of each polygon, the total *energy functional*, $\boldsymbol{E}$ for the spring-connected system shown in Fig. 3.8 is given by:

$$\boldsymbol{E} = \sum_{k=1}^{10} E_k \tag{3.12}$$

3.3.2 Energy Minimization

To contain the complete information about the configuration of the system at any instant, we store the position and orientation of all the eight polygons in a single vector called *position vector*, $\boldsymbol{p}$. A more natural choice should have been $\boldsymbol{x}$, which is saved for some later use. $\boldsymbol{p}$ is given by:

$$\boldsymbol{p} = [x_1, y_1, \phi_1, x_2, y_2, \phi_2 .....x_8, y_8, \phi_8]^{\mathbf{T}} \tag{3.13}$$

where subscripts denote the polygon number [Fig. 3.8]. If a small change from $\boldsymbol{p}$ to $\boldsymbol{p}+\Delta\boldsymbol{p}$ changes the energy functional value from $\boldsymbol{E(p)}$ to $\boldsymbol{E}(\boldsymbol{p}+\Delta\boldsymbol{p})$, then through the second order Taylor expansion of $\boldsymbol{E}$ around $\boldsymbol{p}$, $\boldsymbol{E}(\boldsymbol{p}+\Delta\boldsymbol{p})$ can be expressed as:

$$\boldsymbol{E(p}+\Delta\boldsymbol{p}) = \boldsymbol{E}(\boldsymbol{p}) + \boldsymbol{J}|\Delta\boldsymbol{p} + \Delta\boldsymbol{p}^{\boldsymbol{T}}|\boldsymbol{H}|\Delta\boldsymbol{p} \tag{3.14}$$

where *w.r.t.* all 24 elements in $\boldsymbol{p}$, $\boldsymbol{J}$, the *Jacobian* is the first derivative vector and $\boldsymbol{H}$, the *Hessian* is the second order partial derivative matrix of the total energy functional $\boldsymbol{E}$. We want to find $\Delta\boldsymbol{p}$ such that $\boldsymbol{E}(\boldsymbol{p}+\Delta\boldsymbol{p})$ corresponds to the minima of $\boldsymbol{E}$. We seek to solve the equation that sets the derivative of $\boldsymbol{E}(\boldsymbol{p}+\Delta\boldsymbol{p})$ *w.r.t.* $\Delta\boldsymbol{p}$ to zero:

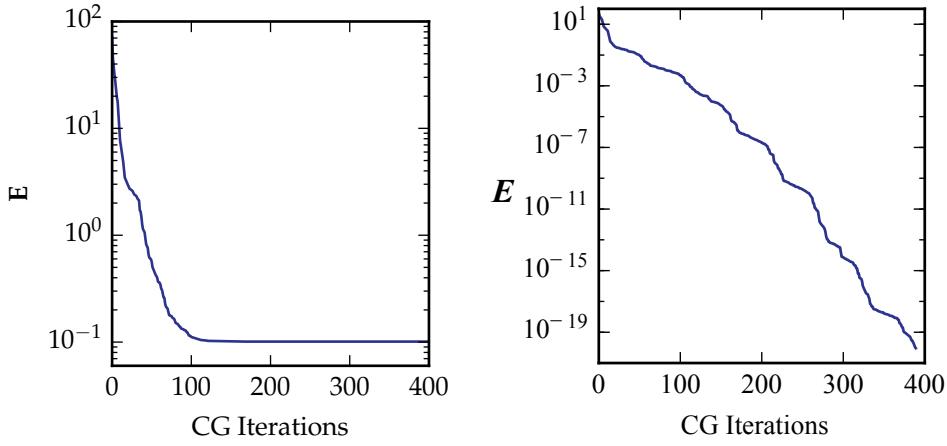
$$\frac{d}{d\Delta p}\Big(\boldsymbol{E}(\boldsymbol{p}) + \boldsymbol{J}|\Delta\boldsymbol{p} + \Delta\boldsymbol{p}^{\boldsymbol{T}}|\boldsymbol{H}|\Delta\boldsymbol{p}\Big) = 0 \tag{3.15}$$

We get:

$$\boldsymbol{H}|\Delta\boldsymbol{p} = -\boldsymbol{J}(\boldsymbol{p}) \tag{3.16}$$

Eq. 3.16 represents a set of 24 nonlinear equations, which we solve using the nonlinear Conjugate Gradient (CG) method, the main reason being that CG is well suited for problems where $\boldsymbol{H}$ is a sparse matrix and requires fairly small storage requirements [94]. Although, the performance may vary, other gradient based methods are also expected to work. The detailed working and pseudocode of the algorithm that we implemented can be found elsewhere [94].

CG convergence and the cutoff value of $\boldsymbol{E}$ – As pointed out earlier, depending upon the polygonal shapes, the system shown in Fig. 3.8 can or cannot relax to a mechanism. This can be understood more intuitively



(a) (b)

Figure 3.10: Conjugate Gradient (CG) convergence curve of a eight-polygon spring connected system [Fig. 3.8] that: (a) relaxes to a finite energy state and (b) relaxes to a zero energy state and forms a mechanism, at least in the vicinity of the relaxed state.

through an extreme case where the sizes of the polygons are so disproportionate that they do not 'fit in' together. In Fig. 3.10, we plot the convergence curve of CG algorithm i.e. the value of $\boldsymbol{E}$ versus each iteration step of CG for two different systems. The final value of $\boldsymbol{E}$ indicates that the system for which the convergence curve is shown in Fig. 3.10(a) equilibrates to a finite energy whereas the one corresponding to Fig. 3.10(b) equilibrates to numerically zero energy value, thus forming a mechanism at least in the vicinity of the relaxed state.

In Fig. 3.11, through a histogram plot, we summarize the final value of $\boldsymbol{E}$ for 10^4 randomly generated/relaxed systems. There are two distinct clusters of $\boldsymbol{E}$ with the left one corresponding to the equilibrated systems while the right one to the systems that could not. The correspondence between the final value of $\boldsymbol{E}$ and whether or not the system relaxes to a mechanism is utilized to save considerable computational power. We set the cutoff value of $\boldsymbol{E}$ to be 10^{-10}, below which CG is terminated as the system is assumed to be practically at zero energy state.

Generating a $D(\theta)$ curve appears straightforward now: relax a system

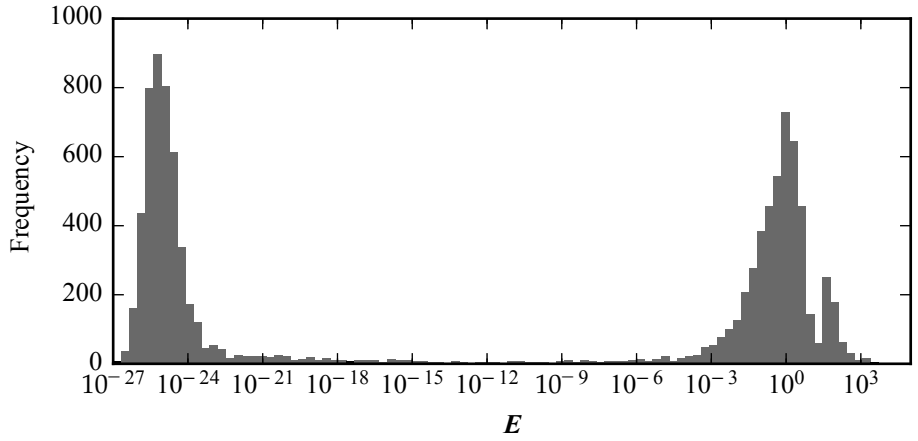


Figure 3.11: Distribution of the final value of $\boldsymbol{E}$ for 10^4 randomly generated/relaxed spring-connected eight-polygon systems [Fig. 3.8]. The left and the right clusters represent systems that relax to a zero energy state and a finite energy state respectively. The two well separated clusters allow to set a cutoff value of $\boldsymbol{E}$, below which the system can be assumed to be at practically the zero-energy state and in which case CG can be terminated. This saves a considerable amount of computational power. We set the cutoff value of $\boldsymbol{E}$ to be at 10^{-10}.

of spring-connected polygons over fixed predetermined θ configurations and measure D. This is stated more clearly in the §3.4.1, where we present a precise definition of θ as well.

3.4 Design Problem Formulation

Complex design problems where the relationship between the *design variables* and the *target functionality* is non-trivial and difficult to construct or approximate can be solved by following a numerical optimization methodology. First one constructs an *objective function* in terms of the design variables, along with imposing the *constraints* to avoid *unfeasible designs*. The objective function quantifies the proximity of a system to the desired functionality. Then, depending on the type of the problem, the objective function is minimized or maximized to find the optimal values of the design

variables. Here, we focus on a mechanism formed by the eight pin-jointed polygons and optimize for a constant target curve $D_t(\theta)$.

In this section, we formulate the design optimization problem, the final purpose of which is to arrive upon a clear definition of the design variables and the objective function that includes *constraint handling* as well. We start with identifying the relevant design variables and present the objective function for the unconstrained optimization. To handle unfeasible designs (such as a system of self-intersecting polygons), we then establish the necessary *constraints* and describe the method to handle them. Finally, we present the *augmented objective function* for constrained optimization.

3.4.1 Design Variables

Design variables are numerical parameters in terms of which the design is specified. During the optimization, design variables change and influence the objective function value. Determining the optimal values of the design variables is the principle aim in design optimization as their optimal values correspond to the optimal design.

Removing the redundant design variables – According to the notation introduced in §3.3.1, while writing the energy functional of the spring-coupled polygon system, we required the values of 11 variables i.e. x, y, ϕ, l_1, ω_1, l_2, ω_2, l_3, ω_3, l_4, ω_4 to completely specify the position and orientation of one single polygon. Thus, it may appear that the total number of design variables for the current problem is 11x8=88. However, a simplified representation of the mechanism, shown in Fig. 3.12(b) illustrates that only the positioning of twelve points in the 2D plane are sufficient to construct any arbitrary system of eight pin-jointed polygons. These twelve points are essentially the connecting pin-joints. By removing the redundant vertices, the alternate representation simplifies the task to only find out the optimal positioning of twelve points in the 2D plane in order to match the target curve $D_t(\theta)$. These points are labeled as $A, B...L$. Note that for the calculation of the Jacobian $\boldsymbol{J}$ and Hessian $\boldsymbol{H}$, we map the coordinates of the simplified notation back to the shape parameterization given by the 88 variables. Given the positioning of the 12 points, this is not so difficult, for example the rod labeled P_1 in Fig. 3.12(b) can be seen as a four-sided polygon with 'overlapping' two pairs of vertices, and so on.

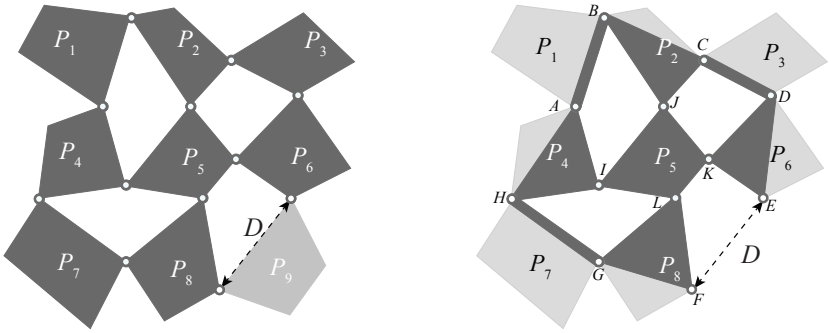


Figure 3.12: Redundant polygonal vertices in (a), which have no effect on the resulting $D(\theta)$ curve removed in (b). Such a formulation helps to reduce the design problem to essentially optimizing for the position of the twelve connecting pin-joints labeled $A, B.....K$ in order to match the target curve $D_t(\theta)$.

While formulating an optimization problem, from the computational points of view, it is immensely important to remove the redundant design variables and find the most precise parameterization of the design. A presence of large number of redundant variables can lead to a difficulty in convergence towards an optimal solution due to the search algorithm's tendency to get stuck in a region where the change in the value of the variables has no effect on the value of the objective function. The new representation produces a compact, lower-dimensional and numerically effective encoding of the problem which now only requires to optimize the value of 12x2=24 design variables.

Measurement of the angle $\boldsymbol{\theta}$ – At this stage, we are in a suitable position to describe the method chosen to measure θ. We remind that θ measures the amount of internal deformation of a mechanism (§3.1). Imagine a mechanism constructed from randomly positioned 12 points in the 2D plane. For simplicity, assume the mechanism to be without any self-intersections. From this *undeformed configuration* of the mechanism, we can back-construct the shape information of every polygon. For convenience, one can presume the polygons in their current orientation to be unrotated i.e. set the parameters $\phi_1, \phi_2 ... \phi_8$ to zero. θ is set to 0^o at this *neutral*

configuration. The hinging motion is then mimicked by relaxing the system over desired θ configurations, throughout which the central polygon P_5 is always kept from rotation and translation, as if 'nailed' into the 2D plane. Numerically, this is achieved by always explicitly setting (x_5, y_5, ϕ_5) in the position vector $\boldsymbol{p}$ to their initial values and fixing all their related derivatives in the $\boldsymbol{H}$ and $\boldsymbol{J}$ to zero, thereby preventing them from adjusting during the CG convergence procedure.

Now we are able to vary θ and actuate the mechanism. The most natural way to do so is to rotate polygon P_2 around the pin-joint connecting it with the polygon P_5, numerically which is achieved by explicitly specifying ϕ_2 and once again fixing all the derivatives related to it in $\boldsymbol{H}$ and $\boldsymbol{J}$ to zero. Forced to relax in this configuration, the remaining six polygons adjust their positions and orientations accordingly. Following this procedure, we determine $D(\theta)$ for θ ranging from -60° to 60°, with the total motion discretized into $N = 20$ steps; θ takes on values -60°, -54°. . . .0°, 0°. . . .54°, 60°. At each of these steps, D is recorded once the system relaxes, yielding a discretized version of the $D(\theta)$ curve[4]. Fig. 3.13 demonstrates the procedure in a flowchart form.

3.4.2 Objective Function and Constraint Handling

The Objective function is a measure to decide how good or how bad a candidate solution is depending on the distance of its $D(\theta)$ curve from the target curve $D_t(\theta)$. The objective function used here is a combination of two parts. The first part computes the position error as the mean of the square of the Euclidean distances between each D_t^i and the corresponding D_c^i where:
- $\{D_t^i\}$ is a set of points on $D_t(\theta)$ which should be met by the optimal mechanism. $i \in 1, 2...20$. D_t^i can be written in a Cartesian orthogonal coordinate system Oxy as:

$$D_t^i = \left[D_{xt}^i, D_{yt}^i\right]^T, \tag{3.17}$$

[4]We follow a slightly indirect protocol to deform a mechanism! We begin from neutral state, $\theta = 0°$ and on either sides, use the final relaxed state of the previous θ step as the initial condition for the next θ step. This is done to remain computationally economic. CG converges within lesser iterations if the desired final configuration and initial configuration are 'adjacent'.

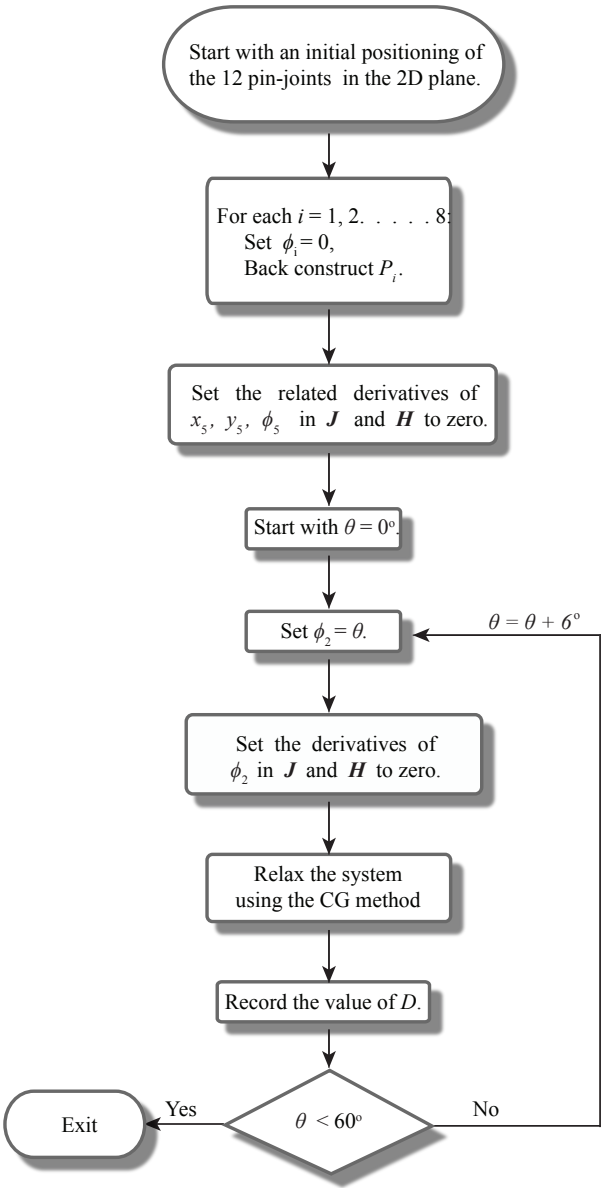


Figure 3.13: A flowchart depicting the numerical procedure to obtain a discretized version of the $D(\theta)$ curve of a mechanism.

where D^i_{xt} and D^i_{yt} are respectively the x and y components of D^i_t.
- $\{D^i_c\}$ is the set of points on the $D(\theta)$ curve of a candidate mechanism for a set of values of θ. Expressing D^i_c in the Cartesian coordinate system:

$$D^i_c = \left[D^i_{xc}, D^i_{yc}\right]^T. \tag{3.18}$$

Accordingly, the first part of the objective function, $\boldsymbol{g}$ can be expressed as:

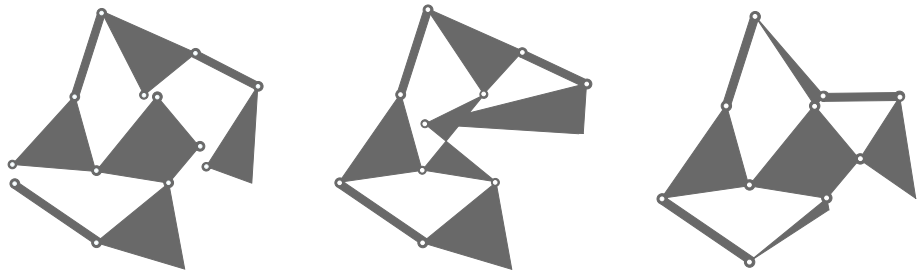
$$\boldsymbol{g} = \frac{1}{N}\sum_{i=1}^{N}\left[(D^i_{xt} - D^i_{xc})^2 + (D^i_{yt} - D^i_{yc})^2\right], \tag{3.19}$$

where N is the number of points on the $D(\theta)$ curve to be synthesized, and is set to 20 in the present study. Clearly, lower the value of $\boldsymbol{g}$, the better is the solution quality.

Constraints to avoid unfeasible designs – One must expect optimal solutions to deform as a mechanism for the entire range of θ, however this can not be ensured, as we may and in fact will encounter systems that do not equilibrate to zero energies for all the θ configurations. Therefore, to avoid any confusion, we will not use the term mechanism while discussing the constraints below, and instead use a much more general term *system* by which we mean an equilibrated spring-coupled polygonal system, although for simplicity in the terminology, in the later sections, we will be a little less strict in this regard.

For our purposes, we require to impose constraints to avoid encountering those systems during the search that can not be realized in the real-world (we in fact have already hinted at one). Such systems lead to an unfeasible design. We identify the following three types of undesirable systems: *(i)* systems that do not equilibrate to a mechanism, *(ii)* systems that contain self-intersection within or among polygons, and *(iii)* systems with disproportionate polygonal size, i.e. unworkable edge length of polygons. An example for each of these systems is shown in Fig. 3.14 (a-c). In an unfeasible design, these cases can appear both separately or together. Below, we establish the constraints to avoid each of these cases and present the method adopted to quantify their violation:

Constraint $\boldsymbol{\Gamma_1}$ – The first constraint is imposed to avoid systems that do not equilibrate to a mechanism [Fig. 3.14(a)], the violation of which



(a) (b) (c)

Figure 3.14: Examples of unfeasible designs. (a) non-equilibrating system, (b) self-intersecting system, and (c) system with disproportionate polygon size.

requires to be checked at every θ step. The measure of its violation on the i^{th} step is given by p_i. We constructed the following *ramp function* to quantify p_i:

$p_i =$	$\boldsymbol{E}_i$
$(\log_{10} \boldsymbol{E}_i)/10 + 1$	$\boldsymbol{E}_i \geq 10^{-10}$
0	$\boldsymbol{E}_i < 10^{-10}$

where for the i^{th} θ step, $\boldsymbol{E}_i$ is the final energy functional value the system relaxes to [Eq. (3.12)]. This relationship is consistent with the previous assumption that all the values of $\boldsymbol{E} < 10^{-10}$ are assumed to be numerically zero, thereby inherently implying that the system equilibrates to zero energy and abides by the current constraint i.e. $p_i = 0$. Systems that equilibrate to finite energy have $p_i > 0$. The total violation is denoted by $\boldsymbol{p}$ and is calculated by summing over all the θ steps:

$$\boldsymbol{p} = \sum_{i=1}^{20} p_i. \tag{3.20}$$

Constraint $\boldsymbol{\Gamma_2}$ – During optimization, the design variables are free to change values which may result in *self-intersecting* systems [Fig. 3.14(b)]. To detect and avoid these systems, we impose another constraint. Identifying

self-intersecting systems is not straightforward and we now show how we implemented it. Through an example of a perfect system, shown in Fig. 3.12(b), three distinct closed polygons shall first be identified: *(i)* outermost polygon *ABCDEFGH*, denoted by P_o, *(ii)* intermediate 'windmill-shaped' polygon *ABICDJEFKGHL*, P_w and *(iii)* innermost polygon *IJKL*, P_i . The necessary and sufficient conditions to guarantee a non self-intersecting system are: *(i)* P_i is 'contained within' P_o, and *(ii)* all three P_o, P_w and P_i are simple and do not self-intersect individually. A system not satisfying any of these two conditions will inevitably display self-intersection.

Similar to the previous constraint, we need to check for the violation of the present constraint for every θ step. Its value at the i^{th} θ step is denoted by q_i. A simple binary quantification for q_i is implemented, where it is assigned a value 1 if self-intersection occurs and 0 otherwise.

$q_i =$	Self Intersection
1	Yes
0	No

The total violation for the complete range of θ is given by $\boldsymbol{q}$:

$$\boldsymbol{q} = \sum_{i=1}^{20} q_i. \tag{3.21}$$

It is important to note that q_i can only be calculated if $p_i = 0$. For $p_i \neq 0$, q_i is simply assumed to be zero.

Constraint $\boldsymbol{\Gamma_3}$ – We have observed that binary penalization of self-intersection increases the chances of systems that have disproportionate polygonal size [Fig. 3.14(c)]. This happens due to the sharp constraint boundary. In order to avoid such systems, we impose the final constraint whose aim is to keep every edge length of every polygon within a desired range. It is quantified using a dynamic method, where for every edge we specify a minimum edge length: l_{min} and a maximum edge length: l_{max}. We set l_{min} at 0.5 and l_{max} at 2.5. These choices are based to keep the polygonal edge lengths within a reasonable limit. Note that we do not need to check for the violation of this constraint and at every θ step and thus we employ the subscript i for another purpose. The violation for the i^{th}

edge length is given by s_i. Because there are a total of 20 edges, i $\in$ {1, 2 . . . 20}. For an edge with edge length l, the following relationship is constructed to quantify r_i:

$r_i =$	Edge Length, l
$0.05(1 - 0.5l)$	$0 \leq l < 0.5$
0	$0.5 \leq l < 2.5$
$0.025(1 - 0.5l)$	$2.5 \leq l < 3.0$
0.05	$l \geq 3.0$

The total violation, $\boldsymbol{r}$ is summed for all the θ steps, and is given by:

$$\boldsymbol{r} = 20 \sum_{i=1}^{20} r_i. \tag{3.22}$$

The functions constructed to measure the violation of the above mentioned constraints are not unique, and in fact other suitable choices can also lead to satisfactory results. We justify our choices solely on the basis of making the scales of $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{r}$ similar. The minimum value that $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{r}$ can attain is zero. The maximum value of $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{r}$ is 20. Hence, we ensure that the values are on a similar scale. This is important because a substantial difference in the magnitudes can lead to an undue imbalance on the search process where satisfying one constraint can be favored over others. A *feasible solution* should satisfy all the three constraints $\Gamma_1, \Gamma_2, \Gamma_3$ and thus these constraints together form the boundary of the *feasible region*. Solutions lying inside the feasible region, will have $\boldsymbol{p} = \boldsymbol{q} = \boldsymbol{r} = 0$.

Penalized objective function – Constraints Γ_1, Γ_2 and Γ_3 form the second part of the objective function. Since, we are solving a minimization problem, violation of these constraints i.e. $\boldsymbol{p}$, $\boldsymbol{q}$ and $\boldsymbol{r}$ are added with the original objective function, $\boldsymbol{g}$ as extra *penalty terms*. This ensures that a solution violating any of the assigned constraints will have higher objective function value than the one that does not. The augmented objective function, $\boldsymbol{f}$ is given by:

$$\boldsymbol{f} = \boldsymbol{g} + \boldsymbol{p} + \boldsymbol{q} + \boldsymbol{r}. \tag{3.23}$$

We now have a complete understanding of the objective function that takes care of the constraints as well. Next, we optimize for the values of design variables such that $\boldsymbol{f}$ is minimized to match the given target $D_t(\theta)$ curve.

3.5 Particle Swarm Optimization and Implementation Details

For complex design optimization problems, the global landscape of the objective function is rugged, multi-modal and patchy, where feasible regions are separated by unfeasible regions in the design variable space. The landscape becomes more complex as the dimensionality of the problem increases. Gradient-based methods terminate at local optimum and unless the objective function consists of only one optimum, will therefore converge to a local optimum. In such problems, we therefore need gradient-free search algorithms that possess a global search capability.

Optimization problems where one can numerically obtains the influence of the decision variables on the value of the objective function, even in the absence of some analytic algebraic model of the system are well suited to be solved through gradient-free population based optimization metaheuristic techniques, popularly known as evolutionary algorithms [55]. The search procedure of these algorithms uses mechanisms inspired by biological evolution, such as reproduction (cooperation), mutation, recombination and selection (competition). One begins with an initial pool of guess solutions. This pool of solutions is referred as *population* and the *population size* indicates the number of candidate solutions in the population. Based on the algorithmic rules, the population members explore the search space and evolve through *generations* by exchanging information between fellow members. With an appropriate implementation, one can expect the population members to converge towards the region where the mean value of the objective function is low (for a minimization problem). The standard terms for the objective function value are: *cost* for a minimization problem and *fitness* for a maximization problem. However, we would simply mention it by the *objective function value*. Convergence of the population towards an extremely low mean objective function value suggests that the algorithm approaches or approximates the global optimum. Specifically, from the perspective of a design optimization problem, this would mean that the algorithm has managed to discover an optimal design.

3.5.1 Particle Swarm Optimization (PSO)

PSO – For our purpose, we utilize an algorithm that simulates swarm behavior: Particle Swarm Optimization (PSO). PSO is a nature-inspired population based search algorithm that is originally accredited to Eberhart, Kennedy and Shi [60, 90, 91]. The algorithm mimics the social behavior of birds flocking and fishes schooling. It starts with an initial population of randomly distributed potential solutions, which are given the name *particles*. Each individual particle occupies a particular *position* and is also initialized with some random *velocity* in the hyper-dimensional search space. Based on the quality measure (objective function value), the algorithm aims to improve the solutions. The particles 'fly' through the search space by means of a set of mathematical expressions. In the standard form of the PSO, these mathematical expressions indicate the balanced movement of each particle towards the best position discovered by that particle individually and by the best position discovered by the swarm so far. Stochasticity is induced into these mathematical expressions to ensure a wide exploration of the search space. Below we explain the governing equations of PSO.

Position and velocity update equations – Let us assume a high dimensional search space. The position and velocity of a particle i is initialized by the vector x_i and v_i respectively [5]. $x_{i,j}$ and $v_{i,j}$ are the position and velocity of the particle in the j^{th} dimension. In our case x_i and v_i are 24 dimensional vectors i.e. j=24. During the search, each particle is attracted towards its best location achieved so far, $xbest_i$ and by the best location discovered by the swarm so far across the whole population, $xgbest$. The simplest update rule for velocity that one can construct is:

$$v_{i,j}^{k+1} = v_{i,j}^{k} + (xbest_{i,j}^{k} - x_{i,j}^{k}) + (xgbest_{j}^{k} - x_{i,j}^{k}), \tag{3.24}$$

where k denotes the generation number. The update rule for position is:

$$x_{i,j}^{k+1} = x_{i,j}^{k} + v_{i,j}^{k+1}. \tag{3.25}$$

The schematic movement of a particle based on Eq. 3.25 is shown in Fig. 3.15. The search process is however, entirely deterministic. To include stochasticity into the search process, the second and the third terms in Eq.

[5] We are using the usual notation of PSO, which should not be confused with the notation x_i used in the §3.3.2 to denote the x-coordinate of the centroid of a polygon P_i.

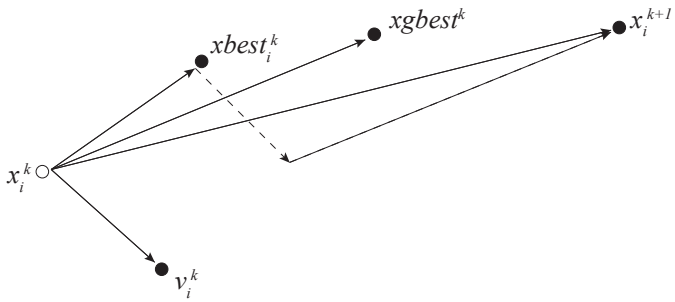


Figure 3.15: Schematic movement of a particle in PSO based on the Eq. 3.24 and 3.25. x_i^k (v_i^k) denotes the position (velocity) of particle i at the generation k. $xbest_i^k$ ($xgbest^k$) denotes the best position found by the i^{th} particle (entire swarm) till the generation k.

3.24 are multiplied with r_1 and r_2 - j dimensional vectors whose elements are random numbers distributed uniformly between 0 and 1. The equation becomes:

$$v_{i,j}^{k+1} = v_{i,j}^k + r_1(xbest_{i,j}^k - x_{i,j}^k) + r_2(xgbest_j^k - x_{i,j}^k). \tag{3.26}$$

Cognition and social parameters, c_1 and c_2 – Along with this, the second and the third terms are also multiplied with the parameters c_1 and c_2 respectively. c_1 and c_2 are two parameters representing the particles confidence in itself (*cognition*) and in the swarm (*social*) respectively. These two parameters are among the most important parameters in the algorithm in that they control the balance between *exploration* and *exploitation*. A relatively high value of c_1 will attract the particles towards their local best experiences while higher values of c_2 will attract them towards the swarm leader. The equation thus becomes:

$$v_{i,j}^{k+1} = v_{i,j}^k + c_1 r_1(xbest_{i,j}^k - x_{i,j}^k) + c_2 r_2(xgbest_j^k - x_{i,j}^k). \tag{3.27}$$

Inertial weight, w – Let us further dissect Eq. 3.27 to gain a better intuition of the algorithm. Without the second and the third parts, the particles will independently keep on flying at the current velocity in the

same initially assigned direction. PSO will not find an acceptable solution unless there is one on their flying trajectories, which is rare. Without the first part, movement of particles becomes memoryless. A particle which has the global best position will be flying at a zero velocity. All other particles will be attracted towards this particle until another particle takes over the global best position. Therefore, it can be imagined that without the first part, the search space shrinks through the generations and the method then resembles a local search algorithm. On the other hand, by adding the first part, the particles have a tendency to expand their search space, that is, they have the ability to explore the new areas. So, more global search ability is expected by including the first part. Shi added another parameter to the equation, an inertia weight w [91].

$$v_{i,j}^{k+1} = wv_{i,j}^{k} + c_1 r_1 (xbest_{i,j}^{k} - x_{i,j}^{k}) + c_2 r_2 (xgbest_{j}^{k} - x_{i,j}^{k}). \tag{3.28}$$

w can be a positive constant or even a positive linear or nonlinear function of generations [95, 96]. Eq. 3.28 represents mathematically the velocity update equation of the PSO in its standard form. In the literature, an abundance of different variants of PSO can be found, but for our purposes, we will adhere to the standard PSO described here, where the velocity update rule is given by Eq. 3.28 and the position update rule is given by Eq. 3.25.

PSO Notations – We will now define some notations. The objective function value of the i^{th} particle at the k^{th} generation is given by $\boldsymbol{f}(x_i^k)$. We introduce a simple rule to write down useful notations compactly: a combination of bracket(s) and the immediate subscript(s) is used to denote an operation eg. $< a >_b$ denotes obtaining the mean of a quantity a over the objects b and similarly through the use of square brackets, $[a]_b$ denotes obtaining the best value of a quantity a over the objects b. Using this notation: *(i)* individually the best position found by the i^{th} particle till the k^{th} generation is given by $[x_i^k]_k$, *(ii)* the objective function value corresponding to $[x_i^k]_k$ is given by $[\boldsymbol{f}(x_i^k)]_k$, *(iii)* globally the best position found by the entire swarm till the k^{th} generation is given by $[x_i^k]_{i,k}$, and *(iv)* the objective function value corresponding to $[x_i^k]_{i,k}$ is given by $[\boldsymbol{f}(x_i^k)]_{i,k}$.

Through a flowchart, we present the complete functioning of PSO in Fig. 3.16. PSO begins with initializing x_i and v_i for each particle i. After initialization, we set $[x_i^k]_k$ to x_i^k and $[\boldsymbol{f}(x_i^k)]_k$ to $\boldsymbol{f}(x_i^k)$ for each particle i. Thereafter, we find the global best position i.e., the position of the swarm

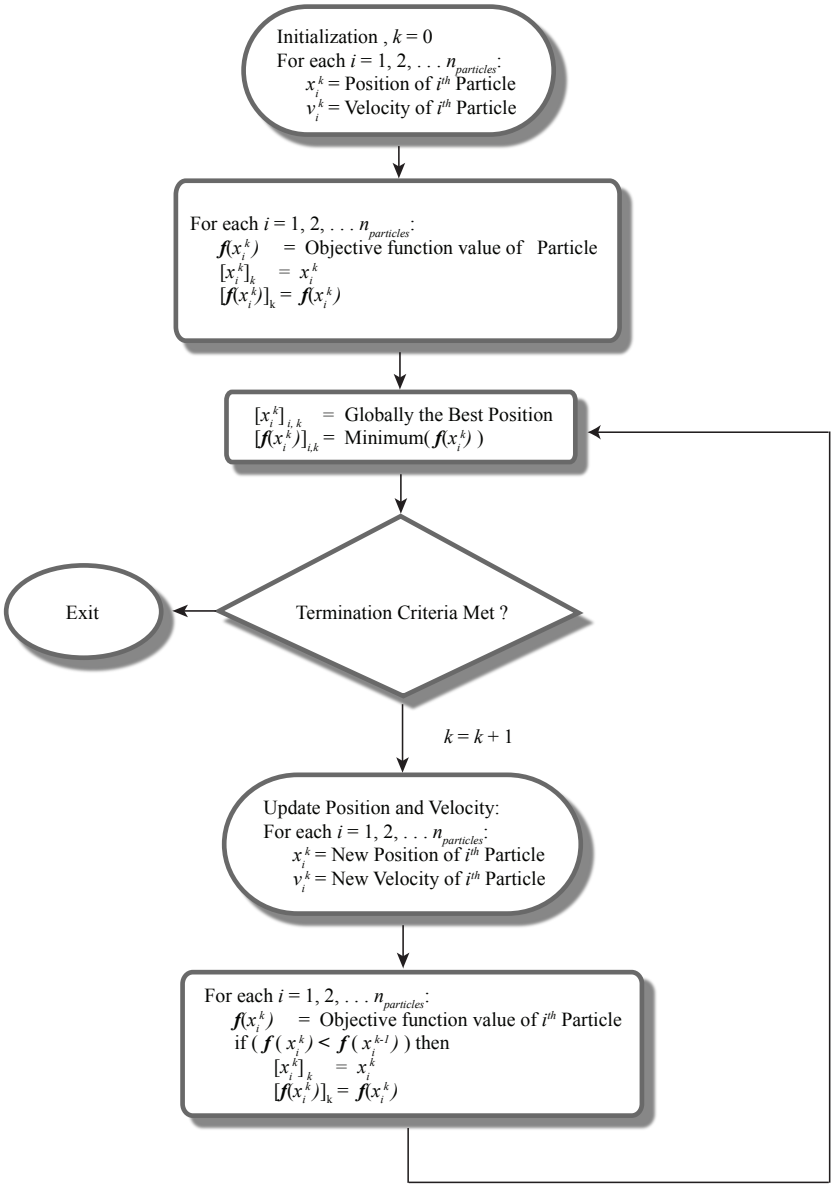


Figure 3.16: Particle swarm optimization depicted in a flowchart form.

leader $[x_i^k]_{i,k}$ and its objective function value $[\boldsymbol{f}(x_i^k)]_{i,k}$. After finding these values, x_i and v_i are updated according to the 3.25 and Eq. 3.28 respectively. With the new position and velocity, $[x_i^k]_k$, $[\boldsymbol{f}(x_i^k)]_k$, $[x_i^k]_{i,k}$ and $[\boldsymbol{f}(x_i^k)]_{i,k}$ are updated if required and the next round of position and velocity update is performed. This is repeated till some termination criteria is satisfied. We discuss our method for terminating PSO in §3.5.2.

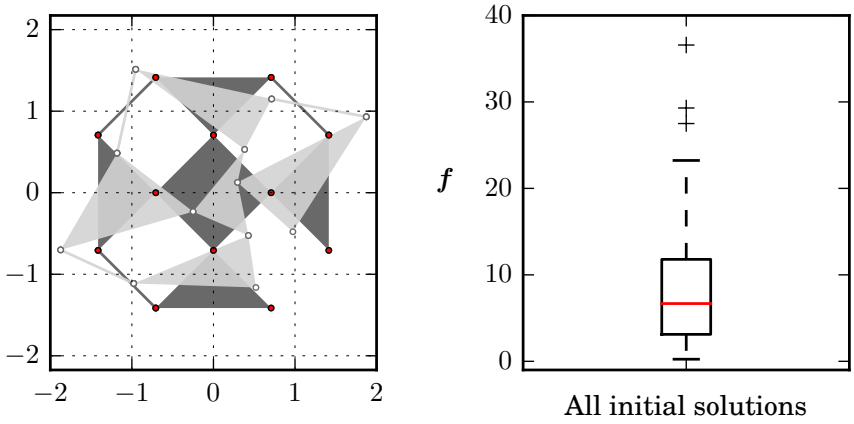
3.5.2 Implementation of PSO

In this section, we cover the swarm initialization and other PSO implementation details that include: *(i)* position initialization of the particles, *(ii)* velocity initialization of the particles, *(iii)* the swarm size, *(iv)* the termination criteria, and *(v)* the target curve.

Position initialization – The search procedure begins by initializing the position of the particles in the search space. The particles should be spread as uniformly as possible. A poor initialization of the swarm, covering the search area inefficiently, may lead to an inability to explore the potential regions thereby missing many of the potentially good solutions. A number of methods have been suggested in the literature to initialize the position of the particles. The usual method is by drawing a uniformly distributed random number along each dimension in the problem space. In many other simple variations, the initial distribution of random numbers is changed from the uniform distribution to other probability distribution like exponential, lognormal and Gaussian distributions [97, 98].

To generate a particle, we first begin with a placement of 12 vertices in the 2D plane (marked by filled red markers in Fig. 3.17(a). As it can be seen that this setting leads to the most simplest mechanism that consists of rotating squares. The mechanism is shown in dark gray color and it is similar to the one previously shown in Fig. 3.2(a). The diagonal length of the square is 1.5. Later, we will provide the constant target function $D_t(\theta)$: $D = 1.0$. Thus, these 12 points are chosen to be placed at a distance such that the initial population is practical for the given target function.

A generic mechanism is now constructed by perturbing this *precursor mechanism* in each of the 24 dimensions by random magnitudes. This is achieved by adding a randomly generated number that's distributed uniformly between (-0.50, 0.50) in each of the 24 dimensions. Fig. 3.17(a) shows this schematically. The perturbed mechanism is shown in the color light



(a) **(b)**

Figure 3.17: (a) Schematic diagram demonstrating the generation of the initial population. We perturb the mechanism shown in the dark gray with random numbers to obtain the generic mechanism shown in light gray. (b) A box-plot summarizing the objective function value $\boldsymbol{f}$ of the initial population with the population size $n_{particles} = 50$ for target curve $D_t(\theta) = 1.0$.

gray. Random numbers are generated within this range so as to maintain as much diversity among the different mechanisms as possible. However, this may also lead to self-intersection and disproportionate polygonal size (violating constraints Γ_2 and Γ_3, see §3.4.2), to check which, all the particles are repeatedly initialized until they satisfy these two constraints. This ensures that the swarm does not dissipates its diversity in the unfeasible search space.

Velocity initialization – The second part of the initialization involves imparting random velocities to particles. We initialize the velocity in each dimension by a random number uniformly distributed between 0 and 1. Initializing with low velocity values works well as it ensures that the particles do not leave the search space.

Swarm size – Swarm size refers to the total number of particles in the swarm. Selecting a correct swarm size is important as it controls the

exhaustive exploration of search space, and is highly problem dependent. A smaller swarm is incapable to explore the search space effectively and a larger swarm leads to a problem in convergence along with being computationally expensive. To find an optimal swarm size, one can start from a relatively few particles and increase the number until one finds the swarm size that works best. We select a swarm of fixed size 50 i.e $n_{particles} = 50$. Through a box-plot shown in Fig. 3.17(b), we summarize the initial objective function values, $\boldsymbol{f}$ of the 50 particles for target curve $D_t(\theta) = 1.0$.

Termination criteria – A termination criteria marks the end of a PSO run. The commonly used criteria are to terminate when the algorithm has exceeded a maximum number of generations or when an acceptable solution has been found. For our purpose, we use the former one and set a maximum of generations, k_{max} to 100 and use it as the termination criteria. We found out that this works well by ensuring that the swarm converges to a final solution, thereby avoiding the chances of any significant premature termination.

Target curve – We optimize for a constant $D(\theta)$ response of the mechanism. Specifically, we provide the target function $D_t(\theta) = 1.0$. The range of θ is [-60°, 60°]. Thus, we desire a total rotational motion of 120° at the hinge connecting polygon P_2 and P_5 [Fig 1.12]. A total rotational motion of 120° serves as a reasonable choice allowing for sufficient internal motion in the mechanism. We believe other choices of D within a reasonable range should work although much higher or lower values of D would render the initial population ineffective.

Code development and deployment – We first developed a quick and dirty Python program to test our model on a general-purpose computer and used guess values of the hyperparameters $\{\omega, c_1, c_2\}$. We noticed a general decay in the mean objective function values of the swarm with generation. As we rationalize in the next sections, our purposes required to run the code for considerable number of times and hence demanded high-performance computing facility. For this reason, we developed our main version in the more efficient programming environment of C++, and deployed it over the Lisa computer cluster of SURFsara [99].

3.6 Parameter Selection for Optimum Search

As a common feature of nearly all metaheuristics, the parameter setting influences the search behavior and their right values are mainly problem dependent [100]. In this section, we first understand the influence of the parameters $\{\omega, c_1, c_2\}$ on the search quality of PSO with the main aim of finding the optimal values of these parameters that impart the maximum (global) search capability to the algorithm. We remind that ω is the inertial weight, c_1 is the cognition parameter and c_2 is the social parameter. We then take a look at the distribution type of the final solution and reveal that it also depends on the values of c_1 and c_2.

3.6.1 Hyperparameter Optimization

Eq. 3.28 gives the velocity update equation for the PSO. The parameter tuple $\{\omega, c_1, c_2\}$ are the central parameters. A balance between exploration and exploitation is crucial in all gradient-free algorithms, achieving which in the case of PSO relies on a correct parameter setting of $\{\omega, c_1, c_2\}$. These parameters characterize the movement of particles, and the success and failure of the search is heavily dependent on the values of these parameters. In our case, 'poor' search can be identified as the one where the swarm is unable to locate optimal or close to optimal solutions with sufficiently low values of the objective function $\boldsymbol{f}$. Either of these scenarios can result in a poor search: *(i)* particles lose their velocities rapidly and become stationary, *(ii)* particles converge to a locally optimal but poor solution and are unable to 'leap-out' of it, *(iii)* particles fly around in random fashion, and *(iv)* particles gain excessive velocities and 'fly-out' of the search space

Effect of c_1 and c_2 parameters – In order to locate the *sweet-spot* in the parameter space, we first understand the effect of the cognition parameter c_1 and the social parameter c_2 on the quality of the final solution while keeping the inertial weight ω fixed. We begin with a low value of ω and keep it fixed at 0.25. We choose the traditional method for performing hyperparameter optimization: *grid search*, which is simply an exhaustive searching through a specified subset of the parameters [101]. We cover a total range of [0.0, 3.50] in both c_1 and c_2 dimensions and discretize the (c_1, c_2) phase space linearly into a 15×15 grid so that c_1 and c_2 take on values 0.0, 0.25, 0.50 . . . . 3.50. To average out the effect of random

numbers, PSO is run 100 times (denoted by n_{PSO}) for every parameter setting.

Following the notational rule introduced in the §3.5, we define two more notations: *(i)* $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$ denotes the mean objective function value of the best solution discovered at the termination of a PSO search for a certain parameter setting. The mean is calculated over n_{PSO} runs. We explain the notation. By following the notational rule introduced earlier, $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ denotes the objective function value of the best solution discovered across the entire swarm at the generation k_{max} i.e. at the termination of a search and $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$ denotes the mean value of it calculated across n_{PSO} runs. *(ii)* Similarly, $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ denotes the best objective function value that could be discovered while carrying out n_{PSO} runs with a certain parameter setting, thereby representing the extreme statistics.

The most intuitive method to visualize and examine the influence of the cognition parameter c_1 and social parameter c_2 on the search capability of PSO is through a heatmap in the (c_1, c_2) phase space. Through a heatmap in Fig. 3.18(a), we show the variation of $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$, and in Fig. 3.18(b), we show the variation of $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ as we parse through the c_1-c_2 grid. The values corresponding to the color in the grids can be estimated from the colorbar on the right.

Based on the figure, we draw several observations. *(i)* As expected, the parameter setting with zero social parameter value i.e. $c_1 \neq 0$ and $c_2 = 0$ leads to an immensely poor performance. Particles have no communication with each other and only individually explore the search space. *(ii)* Owing to significantly high values of $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$, c_1, c_2 parameter values that belong to the top right corner of the grid also lead to a poor performance. We will show in the next section that within this region, the particles either display non-convergent search behavior i.e. they fly randomly or they display divergent search behavior i.e. they fly out of the search space. *(iii)* For the rest of the (c_1, c_2) values, although we do not notice a huge difference in the general scales of $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$, the values of $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ help to separate out the parameter settings that outperforms the rest i.e. high c_2 values and low c_1 values. Hence, the large blue region in Fig. 3.18(b) corresponds to a much improved PSO performance. Most of these (c_1, c_2) values satisfy the following two conditions:

$$c_1 + c_2 \leq 3.50; \quad c_2 \geq 1.50. \tag{3.29}$$

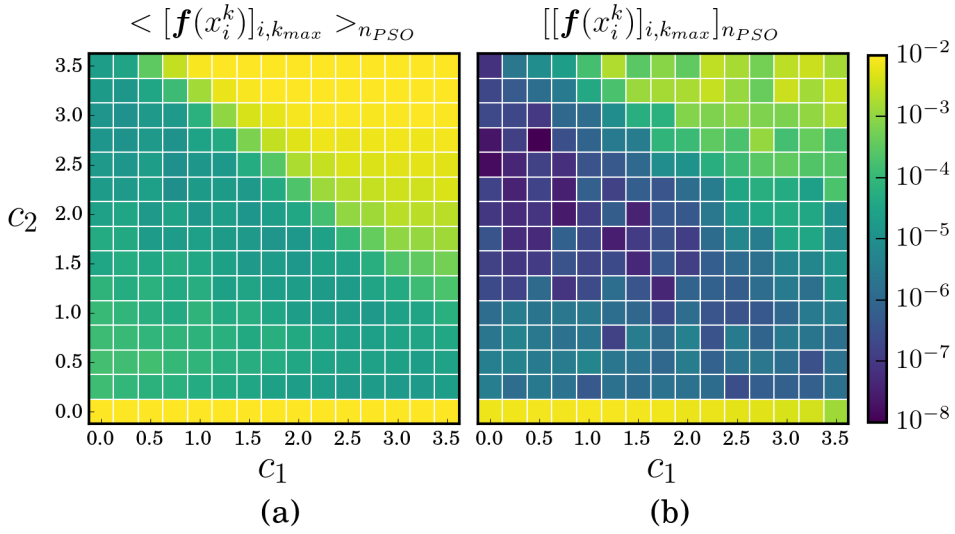


Figure 3.18: For $\omega = 0.25$, the values of $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$ in (a) and $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ in (b) shown for different (c_1, c_2) parameter settings. Over separate n_{PSO} runs, $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$ denotes the mean value of the best objective function values and $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ denotes the best value itself. The colorbar on the right specifies the numerical values corresponding to the colors in the matrix. Lower values of $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ in (b) for low values of c_1 and high values of c_2 confirm their overall better performance.

Effect of the inertial weight ω – We now vary ω and examine its influence on $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$ and $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$. The variation of these two quantities with ω is shown in Fig. 3.19 for its three different values: (a,b) - $\omega = 0.40$, (c,d) - $\omega = 0.60$ and (e,f) - $\omega = 0.80$. Increasing ω results in: *(i)* Expansion of the poor-performing region, and *(ii)* vertical downward shift of the sweet-spot. This seems justified - an increase in ω results in higher velocity magnitudes of the particles, thereby increasing chances for non-convergent search behavior. On the other hand, an increase in ω also results in increased global search capability even for low values of c_2. The influence of ω seems systematic, however, we do not observe any significant improvement in the quality of solutions. For further analysis, we keep the value of ω fixed at 0.25.

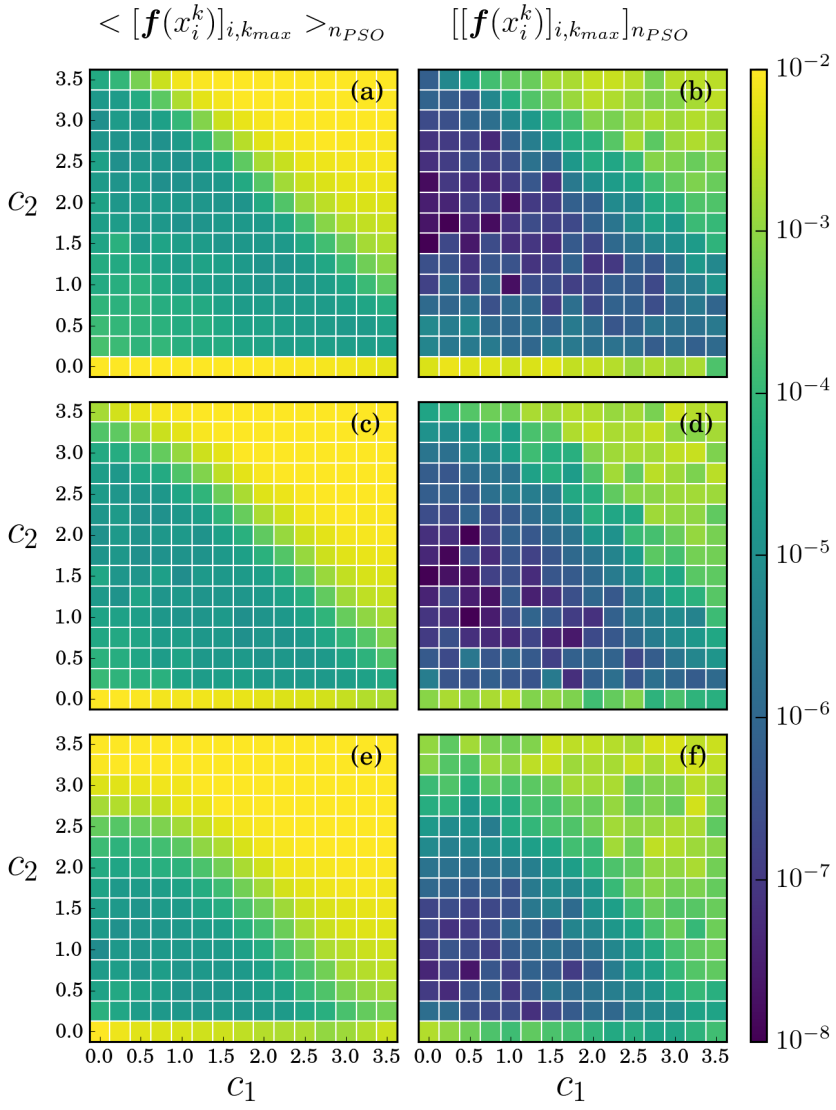


Figure 3.19: Variation of $< [\boldsymbol{f}(x_i^k)]_{i,k_{max}} >_{n_{PSO}}$ (first column) and $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$ (second column) in the (c_1, c_2) space for different values of ω. (a, b) - $\omega = 0.40$. (c, d) - $\omega = 0.60$. (e, f) - $\omega = 0.80$. Effect of ω is systematic but there is no notable effect on the performance of PSO.

3.6.2 Distribution of Final Solutions

We will now take a detailed look at the distributions of $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ through which we will demonstrate that: *(i)* within the non-convergent regime, i.e. for higher values of both c_1 and c_2, the final solution $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ can essentially be described by a gaussian distribution, and *(ii)* we will also show that a lognormal distribution fits the data well for the best performing parameter setting in the (c_1, c_2) space, given by Eq. 3.29.

In Fig. 3.20(a), cumulative distribution function (CDF) of $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ is shown for several parameter settings with higher values of both c_1 and c_2 (labeled on a grid). A sharp jump in the CDF's is noticed for many of the settings due to no improvement from the initial population itself. Overall, these parameters result in a poor PSO performance. In Fig. 3.20(b), it is shown that rescaling each CDF by subtracting its mean and dividing by its standard deviation causes all the data to collapse reasonably well onto the CDF of standard gaussian distribution, $D(\chi)$ for a random variable χ:

$$D(\chi) = \frac{1}{2}\left[1 + erf\left(\frac{\chi}{\sqrt{2}}\right)\right], \tag{3.30}$$

where erf is the error function. $\boldsymbol{f}$ is rescaled as:

$$\boldsymbol{f}_c = \frac{[\boldsymbol{f}(x_i^k)]_{i,k_{max}} - \left\langle [\boldsymbol{f}(x_i^k)]_{i,k_{max}} \right\rangle}{SD\left([\boldsymbol{f}(x_i^k)]_{i,k_{max}}\right)}. \tag{3.31}$$

The CDF for a standard Gaussian distribution is shown in red in Fig. 3.20(b). The collapse suggests that for each of these parameter settings, the variable $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ is normally distributed:

$$[\boldsymbol{f}(x_i^k)]_{i,k_{max}} \sim N(\mu, \sigma^2), \tag{3.32}$$

with general values of μ and σ. This is not surprising - at higher values of c_1 and c_2, PSO behaves similar to a random search. We show evidence of this in the next section.

The quality of fit can be measured through residual plots. The residual measures the deviation between the data points and the fit by calculating their difference. In Fig. 3.20(c), we plot the residual values for the Gaussian distribution fit. We remind that for one parameter setting, there are a total of 100 marker points each corresponding to one PSO simulation. We observe the lack of any general trend.

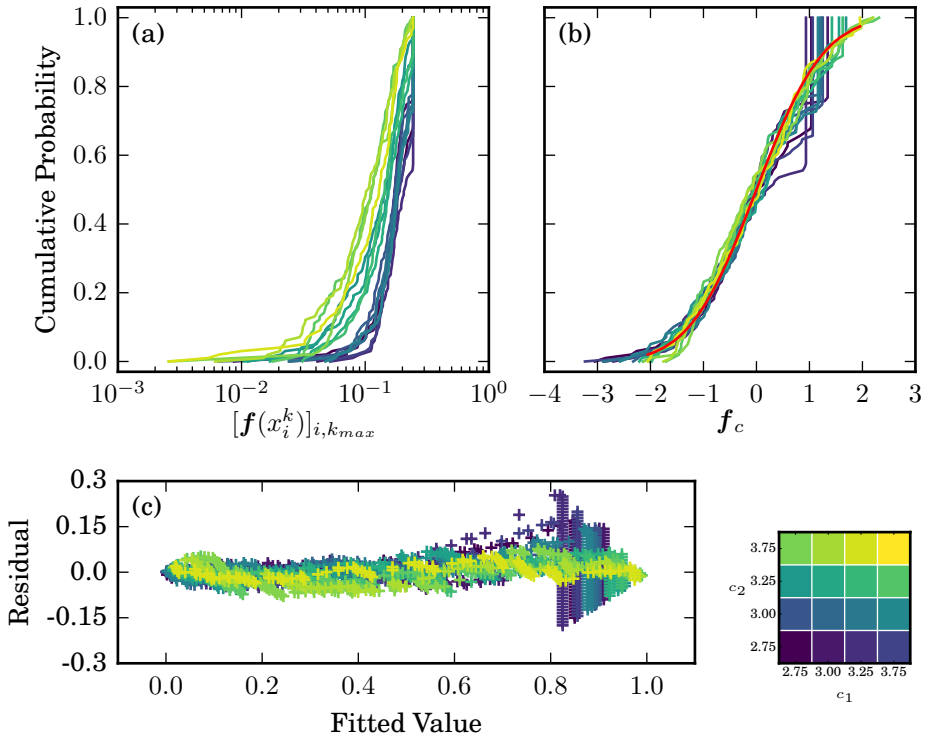


Figure 3.20: For high values of both c_1 and c_2 (see color-guide on the bottom-right): (a) cumulative distribution function (CDF) of $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$, (b) rescaled cumulative distribution function (CDF) of $[f(x_i^k)]_{i,k_{max}}$ [Eq. (3.31)] collapses the data reasonably well to a standard Gaussian distribution (shown in red). (c) Residual plot for the rescaled CDF's lacks any particular trend as such and thus shows that the CDF of the fitted standard Gaussian distribution matches the behavior very well.

We now probe the region in the (c_1, c_2) space that performed better than the rest on the metrics of $[[\boldsymbol{f}(x_i^k)]_{i,k_{max}}]_{n_{PSO}}$. Refer to Fig. 3.18(b) and Eq. 3.29. For a subset of these parameter values, we show the CDF's of $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ in Fig. 3.21(a). Color-codes are provided on the bottom-right of the figure for c_1, c_2 values. In order to collapse all the data on to a single curve, a rescaling of the data is performed. We first log-transform $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ by taking its natural log. The log-transformed data is then

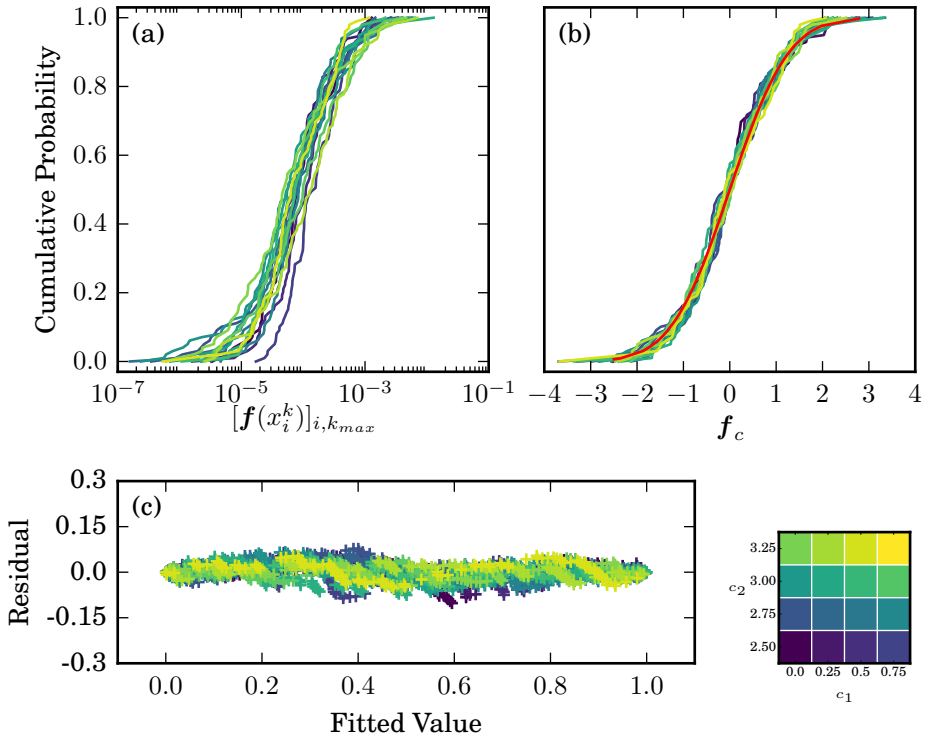


Figure 3.21: For low values of c_1 and high values of c_2 (see color-guide on the bottom-right): (a) cumulative distribution function (CDF) of $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$, (b) rescaled cumulative distribution function (CDF) of $[f(x_i^k)]_{i,k_{max}}$ [Eq. (1.34)] collapses the data reasonably well to a log-normal distribution. The fitted log-normal CDF is shown in the red color. (c) Residual plot for the rescaled CDF's lacks any systematic trend and shows that the fitted log-normal distribution matches the empirical distribution well.

divided by its standard deviation. This makes the standard deviation of distribution equal to 1. Let us denote this rescaled value of $\boldsymbol{f}$ by $\boldsymbol{f}_s$. We then subtract the mean of $\boldsymbol{f}_s$ from $\boldsymbol{f}_s$ itself and divide the result by the standard deviation of $\boldsymbol{f}_s$. Finally the rescaled values are denoted by $\boldsymbol{f}_c$. The

following equations summarizes up the rescaling of $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$:

$$\boldsymbol{f}_s = \frac{\ln([\boldsymbol{f}(x_i^k)]_{i,k_{max}})}{SD\left(\ln([\boldsymbol{f}(x_i^k)]_{i,k_{max}})\right)} \tag{3.33}$$

$$\boldsymbol{f}_c = \frac{\boldsymbol{f}_s - \langle \boldsymbol{f}_s \rangle}{SD\left(\boldsymbol{f}_s\right)} \tag{3.34}$$

For the same parameter subset, the rescaled CDF's are plotted in Fig. 3.21(b). Notice that the CDF's collapse reasonably well to the CDF of a log-normal distribution which is shown in the red color. The formula for the CDF of a log-normally distributed random variable χ is given by:

$$D(\chi) = \Phi\left(\frac{\ln(\chi)}{\sigma}\right), \quad \chi \geq 0; \sigma \geq 0 \tag{3.35}$$

where Φ is CDF of the normal distribution and σ is the standard deviation of the natural log transformation of the original data.

To measure the quality of the log-normal fit shown, we once again plot the residual values for the same parameter settings (c_1, c_2) and show them in Fig. 3.21(c). The residual values are contained within a narrow range and lack any systematic deviation. We can thus comment that the log-normal distribution serves as a sufficiently good fit.

3.7 Visualization of the Search Process

The visualization of the search process enhances our understanding and intuition into the workings of PSO. Specifically, through visualizing the movement of particles, we obtain understanding about the influence of the parameters c_1 and c_2 on the search process. But firstly, visualizing the search that takes place in a high-dimensional domain requires performing a dimensionality-reduction, wherein we map particle positions from the high-dimensional space onto a low-dimensional space (2D), while minimizing distortions of the data. To reduce the dimensionality, we adopt a technique known as *Sammon's Mapping* [102].

Sammon's mapping – We discuss here a technique that maps a dataset from a high-dimensional (say, m-dimensional) input space onto a low-dimensional (say, d-dimensional) output space (with $d < m$). The idea is

to arrange all the points in the output space such that the inter-distance relationship among data points is preserved, i.e. the distances between the data points in the output space resembles the distances in the input space as closely as possible. To do so, Sammon's mapping aims to minimize an error function that quantifies differences of distances between points in the input space and the output space. This error function is often referred as *Sammon's stress* or *Sammon's error*.

Consider a dataset of n data points. If the distance between two points x_i and x_j in the input space is denoted by δ_{ij} and the distance between the corresponding mapped x'_i and x'_j in the output space is denoted by d_{ij}, then SammonâĂŹs stress measure, S is defined as follows:

$$S = \frac{1}{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\delta_{ij}} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{(\delta_{ij}-d_{ij})^2}{\delta_{ij}}. \tag{3.36}$$

The range of S is [0,1] with 0 indicating a lossless mapping. To minimize the stress measure S, any standard gradient based algorithm can be deployed. We used *gradient descent* algorithm.

We will now implement Sammon's mapping to visualize the initial population $\{x_i\}$, i = 1, 2, . . . 50. The transformed dataset is given by $\{x'_i\}$. $\{x'_i\}$ is first initialized in the 2D plane by performing principal component analysis (PCA) on $\{x_i\}$. We show the mapped points in Fig. 3.22(a). An arbitrary, random initialization is also sufficient although PCA leads to some performance improvement [103]. The initial value of Sammon's stress S is 0.255. As, S is minimized, the evolving positions of the particles is shown in the Fig. 3.22 (b-d) for iterations 4, 16 and 64 respectively of the gradient descent procedure. We show the decay of S in Fig. 3.22(e) and observe that S decays down to a residual of ≈ 0.066, below which the loss in the mapping cannot be minimized. Visually, the mapped positions do not alter much after enough iterations and Fig. 3.22(d) effectively shows the Sammon's mapping of the initial population and interestingly illustrates that the initial population $\{x_i\}$ is distributed uniformly. Using the similar procedure, we can map the position of the particles onto the 2D plane for every generation and use this technique to visualize the PSO search procedure.

We now demonstrate the influence of different (c_1, c_2) parameter settings on the PSO search process through the help of four distinct cases. We remind that ω is kept fixed at 0.25.. For each of these cases, we map the

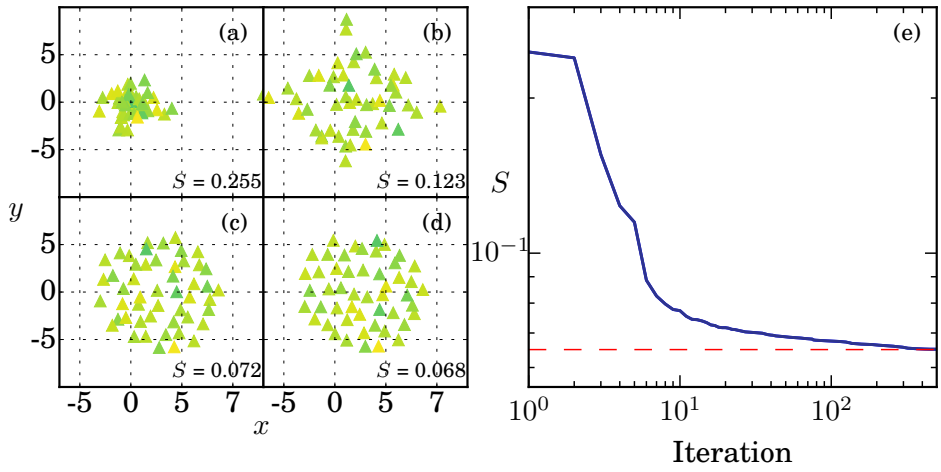


Figure 3.22: The initial population used to initialize PSO mapped onto a 2D space using Sammon's Mapping. (a-d) Snapshots of the mapped positions for iterations 0, 4, 16 and 64 of the gradient descent procedure. Color of the markers denotes the value of $\boldsymbol{f}$ and is based on the colorbar shown in Fig. 3.19. (e) Decay of S versus the gradient descent iterations. The residual value of S is recorded to be 0.065.

particle positions at generations 0, 4, 16 and 64 is shown in Fig. 3.23. In the figure, the color of the data points represents the objective function value $\boldsymbol{f}$, which can be decoded from the colorbar shown in Fig. 3.19. The four cases are discussed below:

(i) $c_1 = 0.25, c_2 = 0.25$ [Fig. 3.23 (a)]: We can observe that the swarm quickly loses diversity and shrinks continuously through generations. With this setting, the swarm lacks global exploration capability and converges at best to a poor local optimum. Thus, only a local exploration of the search space takes place with this parameter setting i.e. low values of both c_1 and c_2.

(ii) $c_1 = 0.0, c_2 = 3.0$ [Fig. 3.23 (b)]: Compared to the previous case, we observe a much broader exploration of the search space taking place before the particles ultimately converge towards a solution. The enhanced global search capability improves the quality of final solution with the pa-

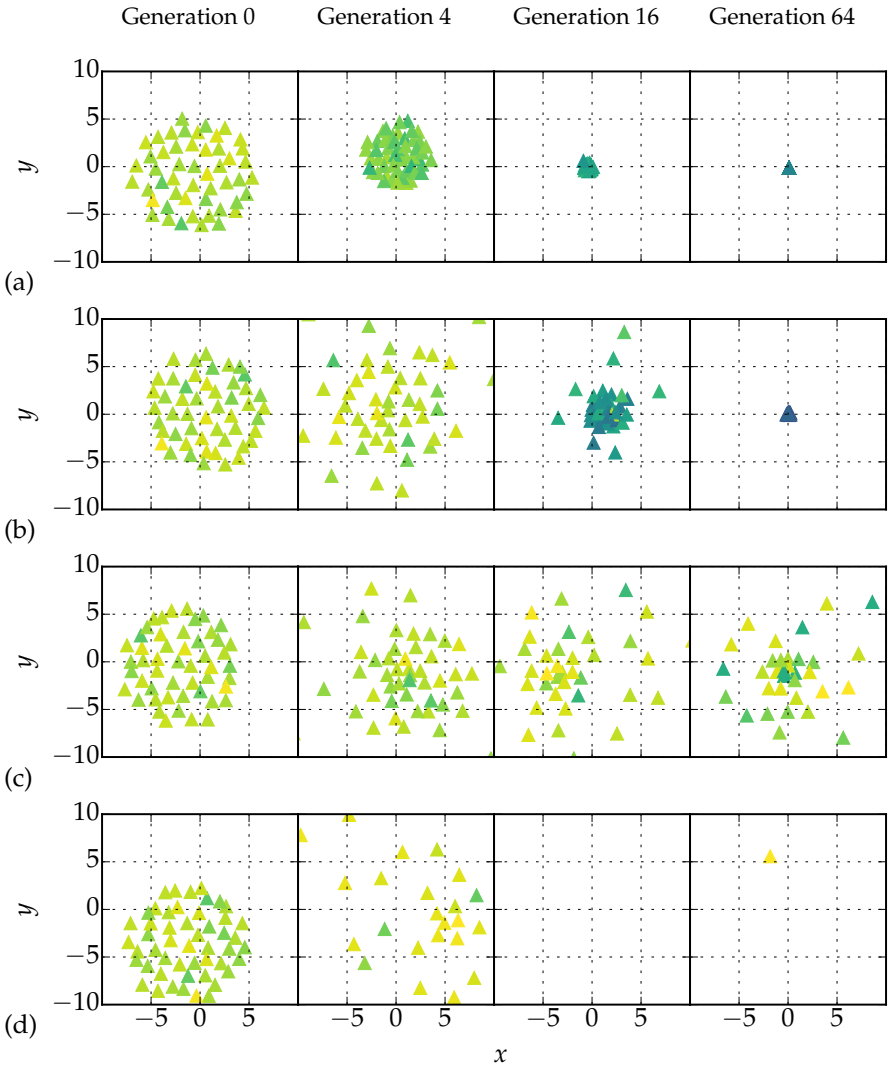


Figure 3.23: Sample PSO runs for different $\{0.25, c_1, c_2\}$ settings visualized through Sammon's mapping. (a) {0.25, 0.25, 0.25} - local search, (b) {0.25, 0.0, 3.0} - global search, (c) {0.25, 2.00, 2.00} - random search and (d) {0.25, 2.25, 2.25} - divergent search. The particle positions are shown for generations 0, 4, 16 and 64 of PSO. Color of the markers denotes the value of $\boldsymbol{f}$ and is based on the colorbar shown in Fig. 3.19.

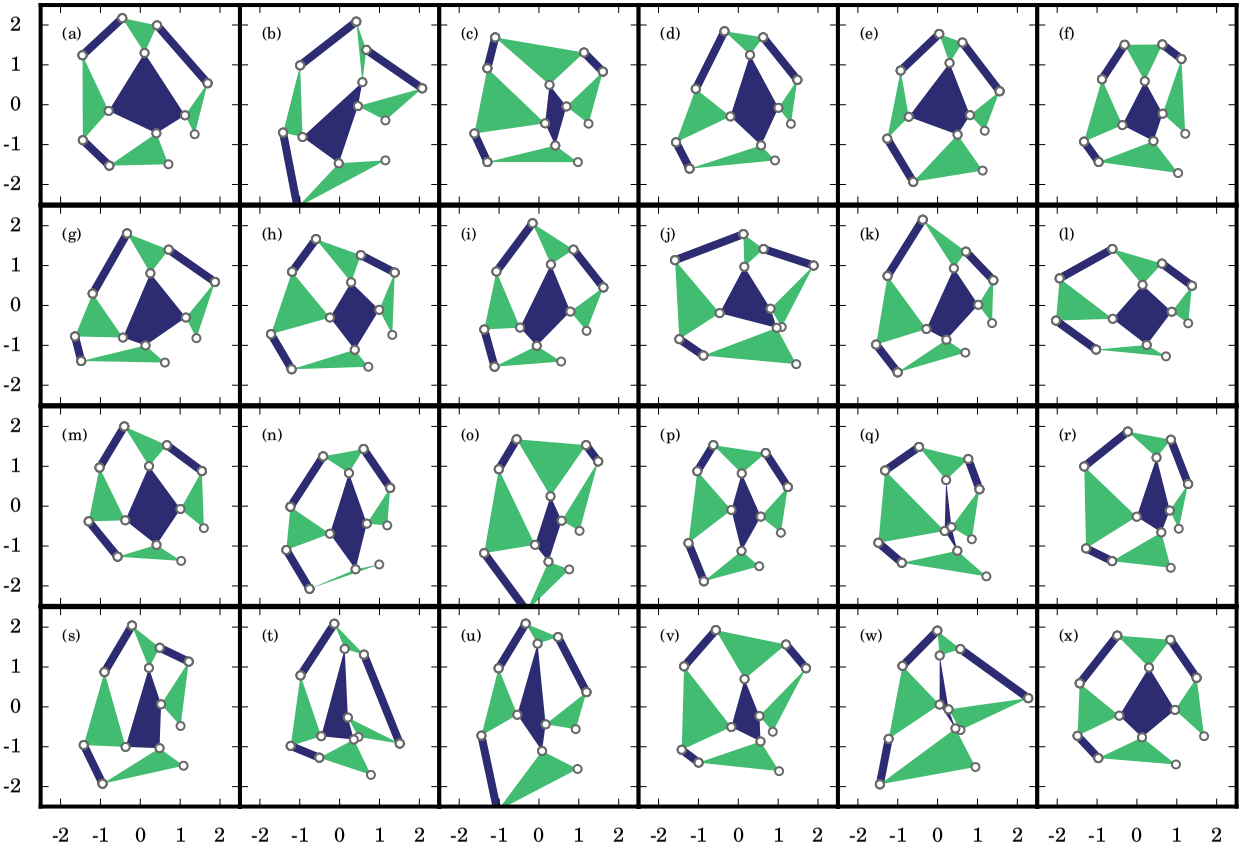


Figure 3.24: (a) - (x): A gallery of top solutions for the target curve $D_t(\theta) = 1.0$ out of an ensemble of 22,500 independent PSO runs. The final objective function value $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ for these solutions ranges from 1.6×10^{-7} to 1.3×10^{-6}.

rameter settings where c_1 takes on low values and c_2 takes on high values.

(iii) $c_1 = 2.0, c_2 = 2.0$ [Fig. 3.23 (c)]: Intermediately high values of both c_1 and c_2 make PSO a random search procedure. We observe that the particles fly randomly through the search space without converging towards a solution.

(iv) $c_1 = 2.25, c_2 = 2.25$ [Fig. 3.23 (d)]: With a further increase in the values of c_1 and c_2, the swarm 'collapses' and the particles positions diverge, making the particles escape-out the search space.

The visualization the search process provide us with the evidence that with the optimal parameter setting [Eq. (3.29)], the swarm carries out a broad exploration of the search space before converging to a solution.

3.8 Results

We have demonstrated the entire machinery to carry out the computer-aided design of mechanisms that meet the target curve $D_t(\theta)$: $D_t = 1.0$ (§3.4 - §3.6). From an ensemble of 2.25×10^{-4} independent PSO runs, the data for which was previously summarized in Fig. 3.18, we extract the top 24 solutions based on the final objective function value $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ and display them in Fig. 3.24. $[\boldsymbol{f}(x_i^k)]_{i,k_{max}}$ for these solutions ranges from 1.6×10^{-7} to 1.3×10^{-6}. We obtained these solutions for distinct $\{\omega = 0.25,$ $c_1,$ $c_2\}$ settings, however, all of them lye within the best-performing (c_1, c_2) subspace, previously defined by the Eq. 3.29. The richness of the objective function landscape in terms of number of distinct possible approximate solutions is reflected in the diversity of the mechanism design.

We pick the solution shown in Fig. 3.24(a) and show how it 'evolves' with generations, k in Fig. 3.25. The mechanisms shown in the figures (a)-(d) are the best solutions (out of the total 50 population members) found up to $k = 0$, 4, 16 and 64. The objective function values are labeled and denoted through $\boldsymbol{f}$. The corresponding $D(\theta)$ curves are shown in the figures (e)-(h). This particular solution was discovered with the parameter settings $\{\omega, c_1, c_2\} = \{0.25,\ 0.0,\ 3.0\}$. The improvement in the performance of the swarm over 100 generations is illustrated in Fig. 3.26 by the decay in the mean objective function value $< \boldsymbol{f}(x_i^k) >_i$ with generation k in (a) and the best objective function value $[\boldsymbol{f}(x_i^k)]_{i,k}$ discovered up to generation k in (b).

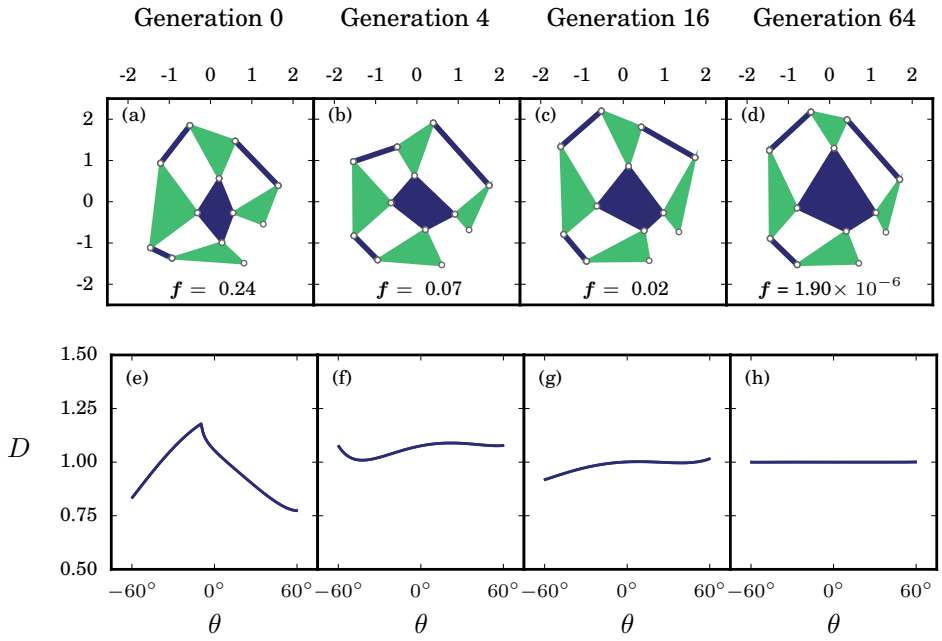


Figure 3.25: Evolution of the final solution shown in Fig. 3.24(a). For generation, $k = 0$, 4, 16 and 64, the successive figures (a)-(d) show the best solutions and (e)-(h) show their corresponding $D(\theta)$ curves. In (a)-(d), $\boldsymbol{f}$ denotes the objective function values.

3.8.1 Validation of the Local Minimum

Finding the global optimum is difficult to guaranteed in a high-dimensional space; nevertheless PSO allows to discover many approximate solutions with very low objective function values. A natural question is: where in the objective function landscape are these solutions 'sitting' or specifically, do the approximate solutions correspond to local minima in the objective function landscape? In general there is not strict guarantee that this is the case. This subsection is aimed at answering the above raised question in details through the help of statistics. In particular, we present evidences that a good PSO search that leads to a good quality final solution ends in a local minimum while this is not true for a poor search.

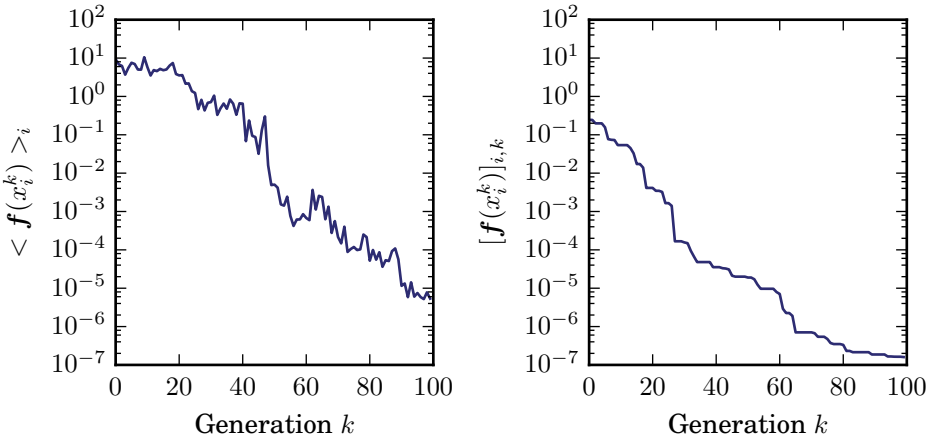


Figure 3.26: The decrease in (a) the mean objective function value $< \boldsymbol{f}(x_i^k) >_i$ with generation k and (b) the best objective function value $[\boldsymbol{f}(x_i^k)]_{i,k}$ discovered up to the generation k as a function of k for the example shown Fig. 3.25.

For a good PSO search – To illustrate our approach, we choose the final solution: $[x_i^k]_{i,k_{max}}$ shown in Fig. 3.24(a) and would like to test if this is a local minimum. Since in this subsection we are mainly interested about the final solutions only, we denote $[x_i^k]_{i,k_{max}}$ with $\boldsymbol{x}$ to avoid unnecessary complexity in the notations although we will return to the original notation if necessary. The numerical procedure for testing if $\boldsymbol{x}$ corresponds to a local minimum is stated on the next page in a procedural form:

For simplicity in notations, we denote $\boldsymbol{f}(x)$ with $\boldsymbol{f}$ and $\boldsymbol{f}(\boldsymbol{x} + \epsilon\hat{d})$ with $\boldsymbol{f}_p(\epsilon)$. Following the procedure described above, we show the variation in $\boldsymbol{f}_p(\epsilon)$ for five different choices of $\hat{d}$ in Fig. 3.27(a). ϵ covers the range [0, 10^{-2}] with the smallest stepsize [6] of 1×10^{-4}. Firstly, we observe a near quadratic increase in $\boldsymbol{f}_p$ with ϵ only around the immediate vicinity of $\epsilon = 0.0$ (see inset). Considering the really small scale of $\boldsymbol{f}_p$ compared with ϵ, this seems justified. To be able to see a clear parabolic increase, we need to zoom in further into ϵ, the choice of which is limited by numerical precision.

[6] Our estimate for the smallest value of stepsize such that the numerical noise is avoided. We discuss this in more detail towards the end of this subsection.

Procedure – Test if a final solution $\boldsymbol{x}$ corresponds to a minimum in the objective function landscape.

Inputs :

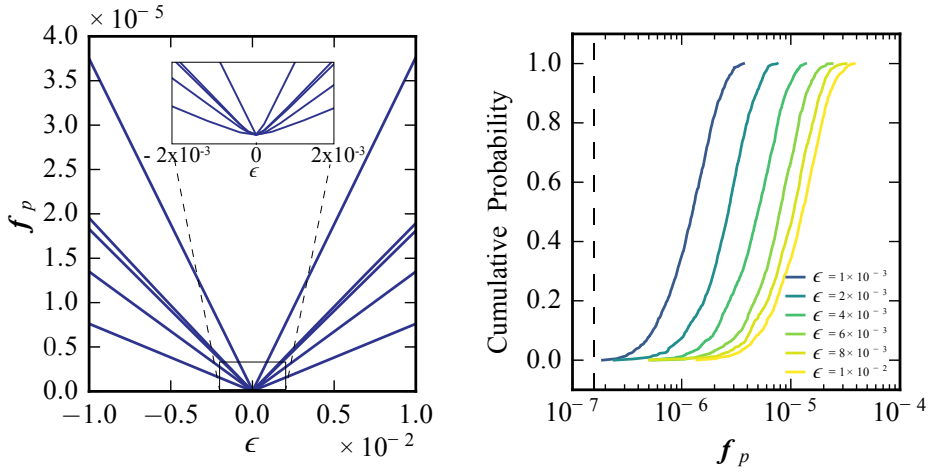
1: Final solution $\boldsymbol{x}$.
2: Stepsize range ϵ.
3: Total number of perturbed directions n_{pert}.
4: Unique perturbed direction $\hat{d}$ (and -$\hat{d}$) for each n_{pert}, where $\hat{d}$ is a 24 dimensional unit vector consisting of random numbers sampled from a uniform distribution.

Result : $\boldsymbol{x}$ corresponds to a local minimum if $\boldsymbol{f}(\boldsymbol{x} + \epsilon\hat{d})$ increases (approximately quadratically) with ϵ for all the n_{pert} directions.

Secondly, we observe that for larger values of ϵ, $\boldsymbol{f}_p$ increase linearly, the reason for which is the large difference in the scales of $\boldsymbol{f}_p$ and ϵ. The results are however are consistent with $\boldsymbol{x}$ being a local minimum.

Five random directions $\hat{d}$ cannot be sufficient to establish that $\boldsymbol{x}$ is indeed a local minimum. A stronger test is: we keep ϵ fixed, and calculate the values of $\boldsymbol{f}_p$ for 1000 random choices of $\hat{d}$ i.e $n_{pert} = 1000$. This we do for six separate values of ϵ: 1.0 x 10^{-3}, 2.0 x 10^{-3}, 4.0 x 10^{-3}, 6.0 x 10^{-3}, 8.0 x 10^{-3} and 10^{-2}. The distribution of $\boldsymbol{f}_p(\epsilon)$ is summarized through cumulative distribution functions (CDF's) in Fig. 3.27(b). There, the vertical dashed line represents $\boldsymbol{f}$, the objective function value of the original solution $\boldsymbol{x}$, and our data shows that all $\boldsymbol{f}_p$ values lie to the right of this line. The horizontal shift between these curves shows that the typical values of $\boldsymbol{f}_p(\epsilon)$ increases with ϵ. Our data presents strong evidence that the solution shown in Fig. 3.24(a) corresponds to a local minimum.

For a poor PSO search – Similarly, we can examine the variation of $\boldsymbol{f}_p(\epsilon)$ in the vicinity of final solution $\boldsymbol{x}$ resulting from a 'poor' PSO search. Based on the final objective function value, we identify one such search from our main ensemble of solutions. The parameter settings corresponding to this search are $\{\omega, c_1, c_2\} = \{0.25, 0.50, 0.50\}$. By performing the previously aforementioned procedure, we show in Fig. 3.28(a) the plot $\boldsymbol{f}_p$ as a function of ϵ. We observe that: *(i)* some of the perturbations $\hat{d}$ lead to a decrease in $\boldsymbol{f}_p$, we call such perturbations as *negative perturbations*, and *(ii)* we no longer have the *local parabolas* observed for the previous case and instead

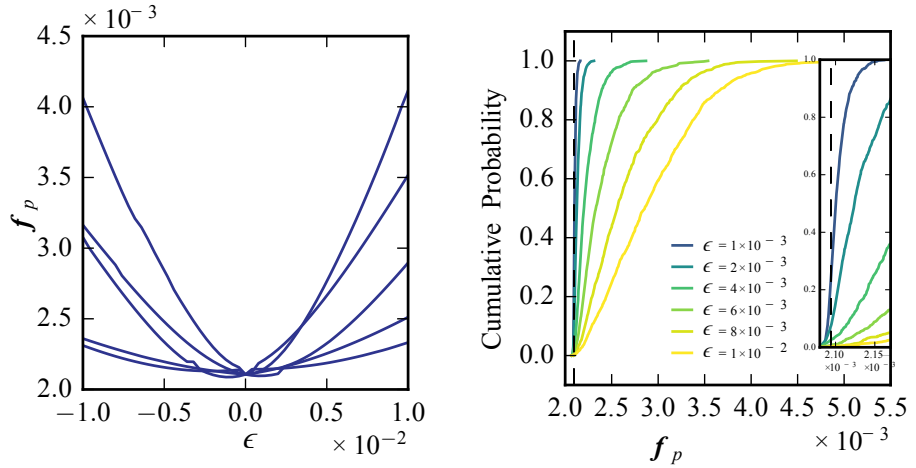


(a) **(b)**

Figure 3.27: (a) For five different choices of $\hat{d}$, $\boldsymbol{f}_p$ vs ϵ for a 'good' solution (Fig. 3.24(a), $\boldsymbol{f} = 1.6 \times 10^{-7}$). Inset figure shows a near quadratic increase in $\boldsymbol{f}_p$ for very low ϵ values. (b) CDF's of $\boldsymbol{f}_p(\epsilon)$ for six different values of ϵ, for each of which $n_{pert} = 1000$. The objective function value $\boldsymbol{f}$ of the unperturbed solution in (a) corresponds to the value of $\boldsymbol{f}_p$ for $\epsilon = 0.0$ and is marked by the dashed black line in (b) w.r.t. which all the values of $\boldsymbol{f}_p$ are higher, thereby proving strong evidence that the solution is a local minimum in the objective function terrain.

find that the variation in $\boldsymbol{f}$ with ϵ depends on the direction in which $\boldsymbol{x}$ is perturbed, implying that $\boldsymbol{x}$ cannot be a local minimum.We further confirm this argument by providing the CDF's of $\boldsymbol{f}_p(\epsilon)$ for the same six values of ϵ as before, with n_{pert} set to 1000 for every value of ϵ. In Fig. 3.28(b), the inset reveals the presence of negative perturbations for all values of ϵ.

Resolutions of PSO – In the previous discussion we remarked that the smallest choice of ϵ might be sensitive to the finite precision with which $\boldsymbol{x}$ is determined and as a guess chose it to be 1×10^{-4}. This worked fine as we did not notice any noise for the data shown in Fig. 3.27. By going much further below in ϵ, here, we explore this topic in detail: the effect of small values of ϵ on the perturbed solutions. By varying ϵ and calculating the



(a) **(b)**

Figure 3.28: (a) For five different choices of $\hat{d}$, $\boldsymbol{f}_p$ vs ϵ for a 'bad' solution $\boldsymbol{x}$. The objective function value of $\boldsymbol{x}$ is $\boldsymbol{f} = 2.09 \times 10^{-3}$. In the vicinity of $\boldsymbol{x}$, $\boldsymbol{f}_p < \boldsymbol{f}$, which implies that $\boldsymbol{x}$ cannot correspond to a local minimum. This, we confirm statistically. (b) CDF's of $\boldsymbol{f}_p(\epsilon)$ for six different values of ϵ, for each of which $n_{pert} = 1000$. The objective function value $\boldsymbol{f}$ of the unperturbed solution in (a) corresponds to the value of $\boldsymbol{f}_p$ for $\epsilon = 0.0$ and is marked by the dashed black line in (b) w.r.t. which some of the values of $\boldsymbol{f}_p$ are actually lower, thus confirming that $\boldsymbol{x}$ is not a local minimum.

statistics of $\boldsymbol{f}_p$, we can calculate the quality of solution $\boldsymbol{x}$ as well as getting an idea about the numerical precision which the local minima are obtained.

We stick with the solution shown in Fig. 3.24(a) and demonstrate the effect of small values of ϵ on the objective function value of the perturbed solutions. To quantify the statistics, for a fixed ϵ, we performed 1000 perturbations with different choices of $\hat{d}$ i.e $n_{pert} = 1000$. Fig. 3.29(a) shows the CDF's of $\boldsymbol{f}_p(\epsilon)$. The colorbar on the right indicates ϵ. The objective function value of the unperturbed solution $\boldsymbol{f}$ is marked by a red colored dashed vertical line. We observe a systematic shift in the CDF's towards the left as ϵ decreases. For these values of ϵ, we can observe the increasing presence of negative perturbations as ϵ becomes smaller. The fact that these values are still extremely close to $\boldsymbol{f}$ indicates that we run into the numerical

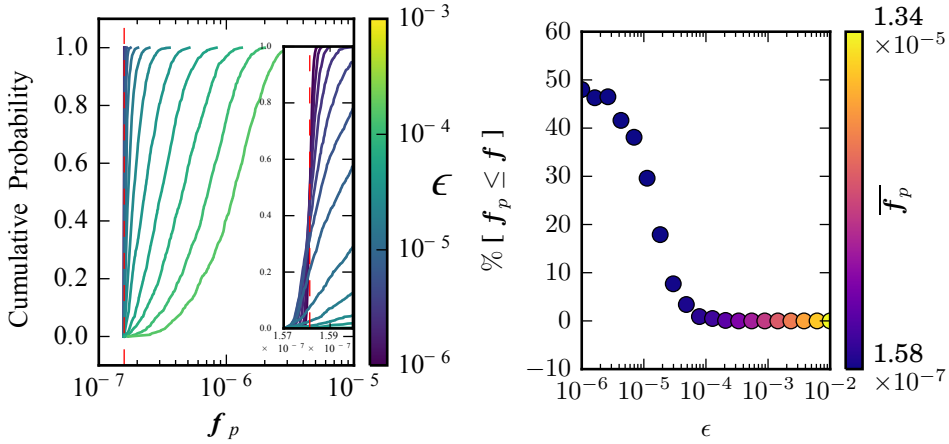


Figure 3.29: (a) CDF's of $\boldsymbol{f}_p(\epsilon)$ for small values of ϵ (see colorbar). Original solution $\boldsymbol{x}$ is shown in Fig. 3.24(a). The inset reveals the presence of negative perturbations within a very narrow range, that decrease with ϵ. (b) Percentage of negative perturbations displays an asymptotic decrease with ϵ, thereby providing with a meaningful bound on correct choice of ϵ. The color of the data points represents the mean value of 1000 perturbations, $\overline{\boldsymbol{f}_p(\epsilon)}$.

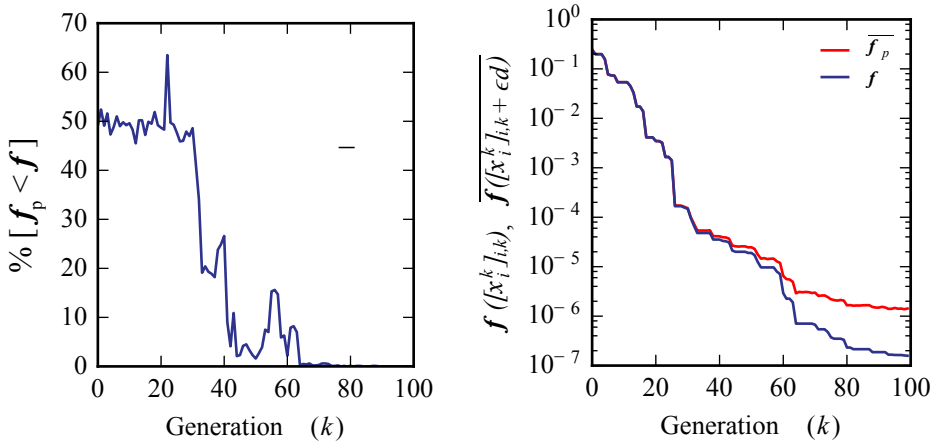
precision problems. To avoid repetition, we do not show CDF's for values of $\epsilon \geq 1 \times 10^{-3}$. They were previously shown in Fig. 3.27(b).

To be precise, for each value of ϵ, we can measure the percentage of negative perturbations. This time we cover a complete range from 10^{-6} to 10^{-2} in ϵ. The results are shown in Fig. 3.29(b), where the colorbar on the right indicates the mean of n_{pert} perturbations, $\overline{\boldsymbol{f}_p(\epsilon)}$. We observe that the percentage of negative perturbations decreases asymptotically with ϵ and reduces to zero as ϵ approaches $\sim$ 1x10^{-3}. Hence, this suggests that numerical precision with which PSO determines this local minima is of the order of 10^{-3}. This graph simultaneously provides us with a good estimate for a safe choice of the step-size ϵ to check if some solution $\boldsymbol{x}$ (within numerical accuracy) is a local minima. To perturb solutions in the coming analysis, we will keep the smallest value of ϵ to 10^{-3} while the largest value will never exceed 10^{-2}.

Can PSO escape a local minimum ? – Now that we have a numerical procedure to check whether $\boldsymbol{x}$ plausibly corresponds to a local minimum, we can ask the following question: How capable is PSO to jump out of local (near-local) minima ? We pick the sample search shown previously in Fig. 3.25 and extract out the best solutions $[x_i^k]_{i,k}$ found till the k^{th} generation for $k = 0, 1, 2.....100$. We then perturb these solutions and by measuring the percentage of negative perturbations, we can make an educated estimate of the local objective function landscape. We keep $\epsilon = 10^{-3}$ and $n_{pert} = 1000$. Fig. 3.30(a) shows the percentage of negative perturbations as a function of k and Fig. 3.30(b) shows the mean value of $\boldsymbol{f}([x_i^k]_{i,k} + \epsilon d)$, $\overline{\boldsymbol{f}_p}$ in red and $\boldsymbol{f}([x_i^k]_{i,k})$ (labeled as $\boldsymbol{f}$ in blue), both as a function of generation k. From Fig. 3.30(a), we observe that the percentage of negative perturbations fluctuates around 50% up until $k \sim 25$, after which this percentage starts to decrease. It is as if PSO is 'sliding down' on a surface in hyperdimensional space in search of a minimum. The curves in Fig. 3.30(b) further support this picture: values of $\boldsymbol{f}([x_i^k]_{i,k})$ and $\boldsymbol{f}([x_i^k]_{i,k} + \epsilon d)$ are extremely close to one another till $k \sim 25$. We then notice a sudden drop in the percentage of negative perturbations: PSO finds itself in a neighborhood where $\boldsymbol{x}$ corresponds to a minimum in $\sim 80\%$. Because we observe sharp fluctuations in the curve shown in Fig. 3.30(a), where it recovers even after dropping to very low values (k > 42), PSO intermittently escapes such regions in the search space that correspond to minima in many of the dimensions. This capability diminishes at larger generation count. Eventually, in this case PSO finds local minimum and $\boldsymbol{f}$ and $\boldsymbol{f}_p$ converge to their respective final values.

3.8.2 Validating Across Large Statistics

We have so far demonstrated the procedure for conducting a local minimum test for one good and one bad solution type each. In this section we explore the relation between the final objective function value and the fraction of negative perturbations across larger statistics. For this purpose, we carried out more PSO simulations. The implementations details of the PSO remain the same as previous. We fix ω at 0.25 and for the (c_1, c_2) parameters, we narrowed down to the good-performing set given by Eq. 3.29. This was done to ascertain that we discover a large number of solutions that are good. For each of the 36 different (c_1, c_2) pairs, 3000 PSO simulations were carried out, which provided 1.08×10^5 new data points.



(a) **(b)**

Figure 3.30: (a) For the search shown in Fig. 3.25, percentage of negative perturbations as a function generation k. The value of ϵ is 1x10^{-3}. (b) Mean objective function value of 1000 perturbed solutions: $\overline{\boldsymbol{f}([x_i^k]_{i,k} + \epsilon d)}$ and the objective function value of the original unperturbed solution: $\boldsymbol{f}([x_i^k]_{i,k})$ both as a function of k.

We now discuss the issue of linkages close to becoming parallelogram linkages. We briefly discussed this in §3.2 and mainly mentioned that the 3x3 network forms a perfect mechanism when all the four internal four-bar linkages are simultaneously parallelograms. We seek to answer several new interesting questions: how does the quality of solutions correlate with the proximity of the linkages to a parallelogram? Does PSO has some bias towards discovering these solutions? We address these question in the following discussion.

We first define an order parameter to quantify the proximity of a generic four-bar linkage to a parallelogram. Consider a generic four-bar linkage Λ_i with bar lengths a_1, a_2, a_3 and a_4 in the cyclic order [Fig. 3.6]. We define a variable s_i to measure the proximity of Λ to parallelogram linkages:

$$s_i = \frac{(a_1 - a_3)^2 + (a_2 - a_4)^2}{\sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}} \tag{3.37}$$

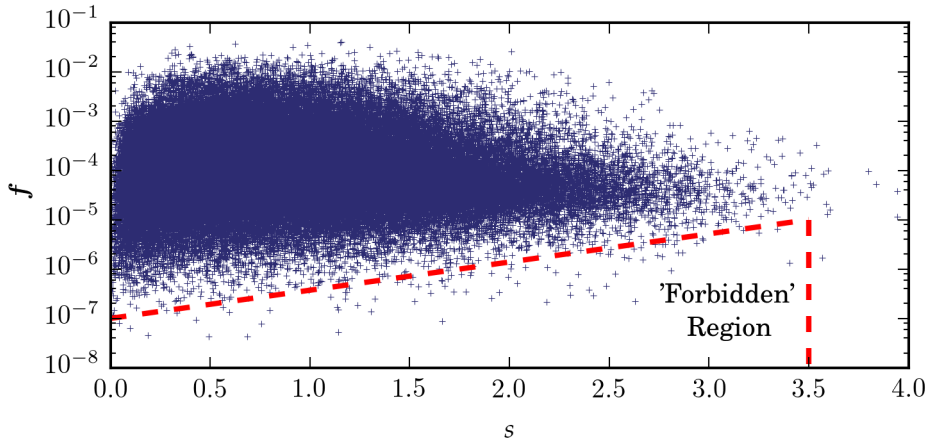


Figure 3.31: A scatter plot of objective function value $\boldsymbol{f}$ vs the order-parameter $\boldsymbol{s}$. 'Forbidden region' corresponds to the $\boldsymbol{f}$-$\boldsymbol{s}$ space where very few solutions are found.

The denominator in the above expression normalizes linkages consisting of varying bar lengths to a common scale. Clearly, the lower the value of s_i, the closer a linkage is to a parallelogram linkage. The order parameter $\boldsymbol{s}$ that measures the proximity of all the four linkages in a 3x3 system is defined as:

$$\boldsymbol{s} = \sum_{i=1}^{4} s_i \tag{3.38}$$

In Fig. 3.31, we show a scatter plot of the objective function value $\boldsymbol{f}$ and order-parameter $\boldsymbol{s}$ for our ensemble of 1.08×10^5 solutions. We observe a wide distribution for both of these variables. Importantly, we note that PSO displays a decline in its ability to discover solutions with high values of $\boldsymbol{s}$. We speculate that this may either be due to poor exploration of the higher $\boldsymbol{s}$ space or it may be possible that *search space with lower* $\boldsymbol{s}$ *values consists of larger density of deep local minima* wherein the algorithm displays a natural tendency to pursue search. This leads to the formation of a 'forbidden' region for higher $\boldsymbol{s}$ values, where PSO did not discover many solutions. It is interesting to notice that the height of the forbidden region on the *log y-axis* appears to increase almost linearly with $\boldsymbol{s}$ suggesting an exponential relation.

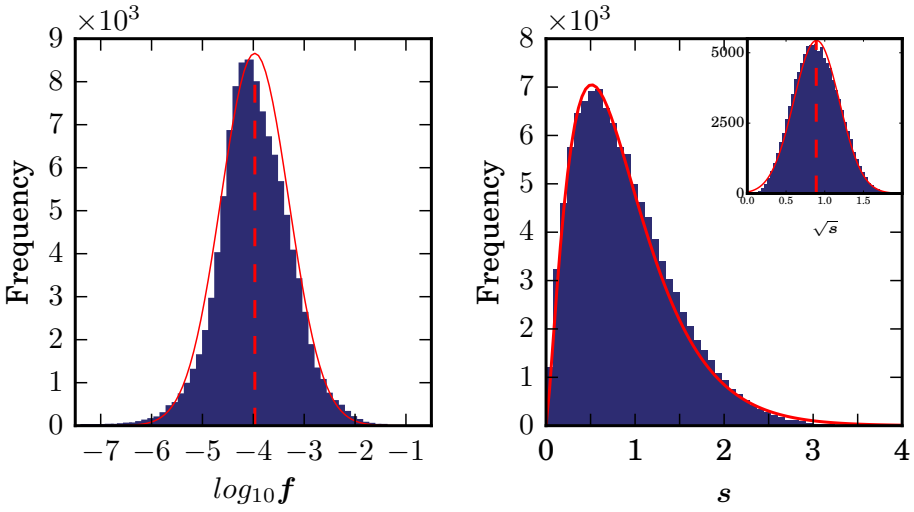


Figure 3.32: For the data shown in Fig. 3.31: (a) Histogram plot of $log_{10}\boldsymbol{f}$. The curve in red shows the PDF of a normal distribution with $\mu \sim$ -3.97 (marked by dashed vertical line) and $\sigma \sim 0.66$. This implies that $\boldsymbol{f}$ is log-normally distributed. (b) Histogram plot of $\boldsymbol{s}$. The curve in red shows the PDF of a gamma distribution implying that $\boldsymbol{s}$ is gamma distributed. Inset shows that $\sqrt{s}$ is normally distributed with $\mu \sim 0.89$ (marked by dashed vertical line) and $\sigma \sim 0.29$.

Distributions of $\boldsymbol{f}$ ($\log_{10}$ transformed) and $\boldsymbol{s}$ are shown separately through histogram plots in Fig. 3.32(a,b) respectively. For both of these plots we chose 50 bins of equal width to represent the data with $\boldsymbol{f}$ ranging from 1×10^{-8} to 1×10^{-1}, and $\boldsymbol{s}$ ranging from 3.31×10^{-3} to 3.94. The histogram shape of $log_{10}(\boldsymbol{f})$ appears to be bell-shaped like a normal distribution indicating that $\boldsymbol{f}$ is log-normally distributed; a normal distribution, shown in red, is able to fit the data well (except near the peak). The approximate fit parameters for the normally distributed $log_{10}(\boldsymbol{f})$ are: mean, $\mu \sim$ -3.97 and standard deviation, $\sigma \sim 0.66$. The log-normal distribution type of $\boldsymbol{f}$ is consistent with §3.6.2 [Fig. 3.21]. A gamma distribution fits the histogram of $\boldsymbol{s}$, shown in red. Through the fit we found out that the shape and scale parameters (usually denoted by k and θ [??]) are 2.36 and 0.37

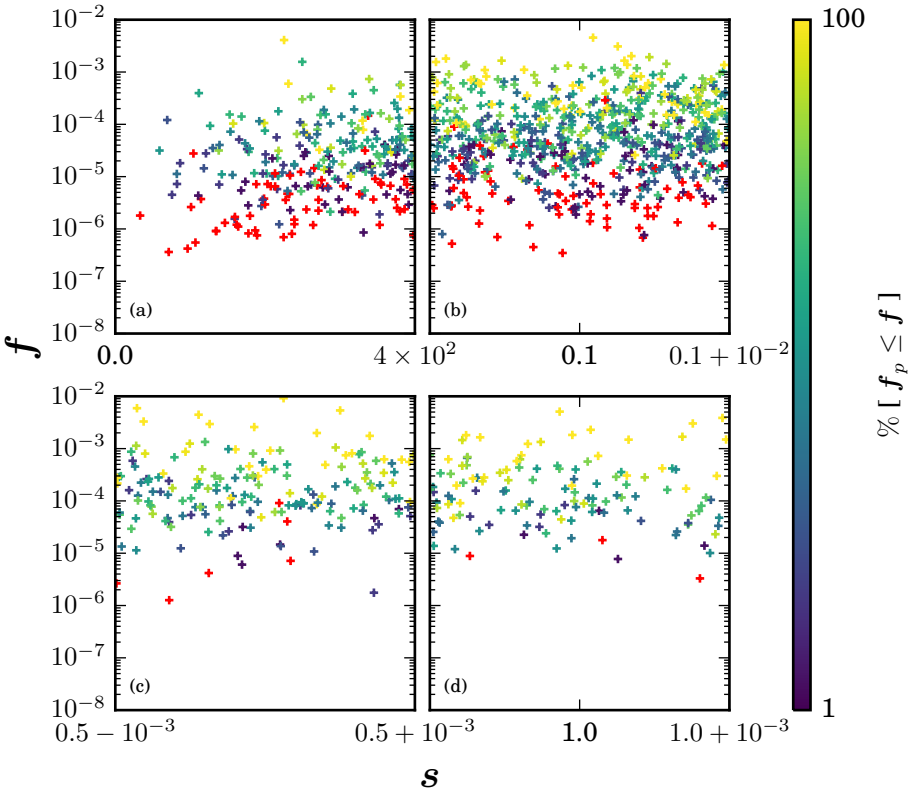


Figure 3.33: Quality of final solutions measured through the percentage of negative perturbations (see colorbar) for (a) $\boldsymbol{s} \leq 0.04$, (b) $0.1\text{-}10^{-2} \leq \boldsymbol{s} \leq 0.1\text{+}10^{-2}$, (c) $0.5\text{-}10^{-3} \leq \boldsymbol{s} \leq 0.5\text{+}10^{-3}$, and (d) $1\text{-}10^{-3} \leq \boldsymbol{s} \leq 1\text{+}10^{-3}$. Red color is used to separately mark only those data points for which none out of the 500 perturbations were found to be negative, implying that such points are strong candidates for a true local minimum. The appearing shared *x-axis* tick label belong to the left figures.

respectively. We carried out a square-root transformation of $\boldsymbol{s}$ and found that the shape of histogram is well approximated by a normal distribution [Fig 1.32(b), inset], shown in red) with $\mu \sim 0.89$ and $\sigma \sim 0.29$.

We now investigate whether there is a correlation between the values of $\boldsymbol{f}$ and the fraction of negative perturbations across the different intervals

of the order parameter $\boldsymbol{s}$. We first select four intervals in $\boldsymbol{s}$: *(i)* $0.0 \geq \boldsymbol{s} \leq 0.04$, *(ii)* $0.1\text{-}10^{-2} \leq \boldsymbol{s} \leq 0.1{+}10^{-2}$, *(iii)* $0.5\text{-}10^{-3} \leq \boldsymbol{s} \leq 0.5{+}10^{-3}$, and *(iv)* $1\text{-}10^{-3} \leq \boldsymbol{s} \leq 1{+}10^{-3}$, and extract the solutions falling within that range. For each solution, we then repeat the previously explained procedure, i.e. perturbing the solutions by a fixed step-size ϵ in the n_{pert} directions. We choose $\epsilon = 10^{-3}$ and $n_{pert} = 500$. Due to a large number of points to check, this is done to half the computational time. We remind that previously we set n_{pert} to 1000. Primarily, we are interested to find out the percentage of negative perturbations, if none, we infer that particular solution as a local minimum

We show the results in Fig 3.33. There, the colorbar denotes the percentage of negative perturbations. In the scatter plot, color red is used to separately mark only those data points for which none out of the 500 perturbations were found negative. These points are prime candidates for true local minima.We draw several conclusions: *(i)* substantial fraction of local minima correspond to values of $\boldsymbol{f}$ of order 10^{-5} or less. This is consistent across the figures (a)-(d). The number of these points decrease with an increase in $\boldsymbol{s}$, and *(ii)* We observe an appreciable and systematic increase in the percentage of negative perturbations for solutions with increasing values of $\boldsymbol{f}$. In general solutions with lower $\boldsymbol{f}$ values have lower percentage of negative perturbations. For increasing values of $\boldsymbol{f}$, most solutions are not strict local minimum. Considering that our implementation of PSO only used a population size of 50, it is not surprising that such solutions are found when exploring a high dimensional search space.

Variability in $\boldsymbol{f_p}$ – For the data points shown in Fig. 3.33(a-d) grouped on the basis of $\boldsymbol{s}$, we measure the variability in $\boldsymbol{f}_p$ by calculating the values of coefficients of variation (CV), which measures how large is the standard deviation with respect to the mean. This we do because of large scales of difference across the values of $\boldsymbol{f}$ and therefore the mean values of $\boldsymbol{f}_p$. Essentially, we expect that the solutions representing minimum or near-minimum should have higher values of $CV(\boldsymbol{f}_p)$ than the solutions that do not. We report the results in Fig. 3.34. There, the data sets in (a)-(d) are the same as previously categorized based on the value of $\boldsymbol{s}$ in Fig. 3.33. The color-code of the data points is also same. We observe significant differences in the values of $CV(\boldsymbol{f}_p)$. Typically, across $\boldsymbol{s}$, the values are lower for data points that are not local minima than the ones that are more likely to be.

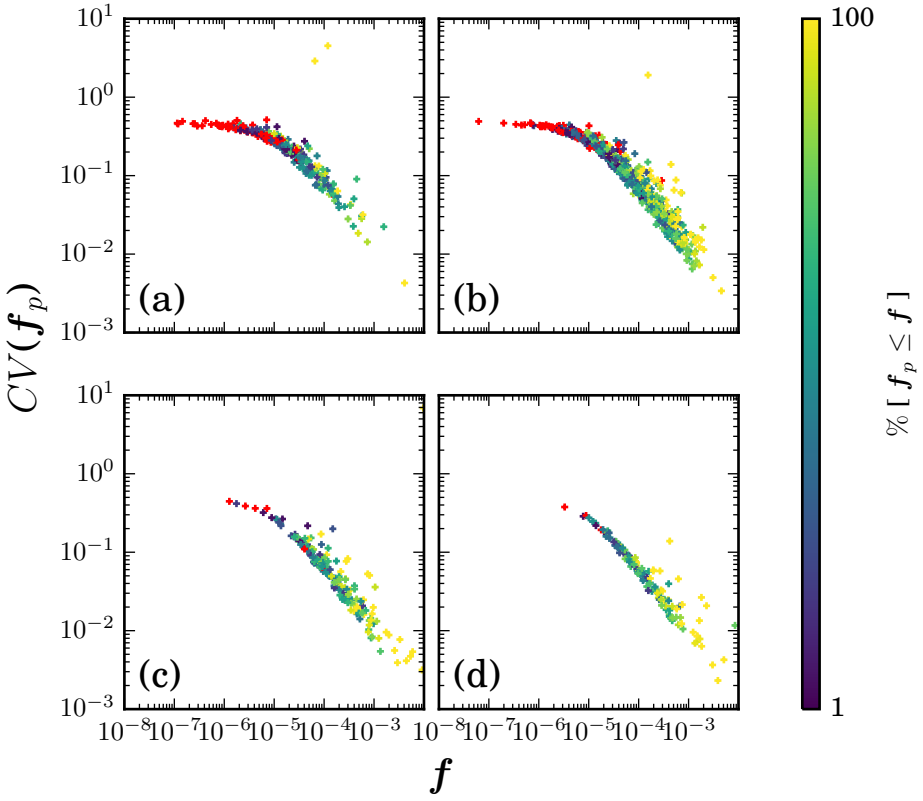


Figure 3.34: For the same data in Fig. 3.33(a-d) grouped on the basis of $\boldsymbol{s}$, $CV(\boldsymbol{f}_p)$ vs $\boldsymbol{f}$. We observe that across all the four intervals of $\boldsymbol{s}$, the values of $CV(\boldsymbol{f}_p)$ are lower for the data points that are not local minimum than the ones that are more likely to be.

Additionally, we notice that $CV(\boldsymbol{f}_p)$ for these two groups of data points follow different power law relations.

3.9 A Proof of Concept with 3D Printing

In this section, we present 3D printed flexible unit cells and tilings based on the computer-designed mechanisms. 3D printing begins with the creation of a 3D CAD design of the mechanisms in CAD software. While doing this,

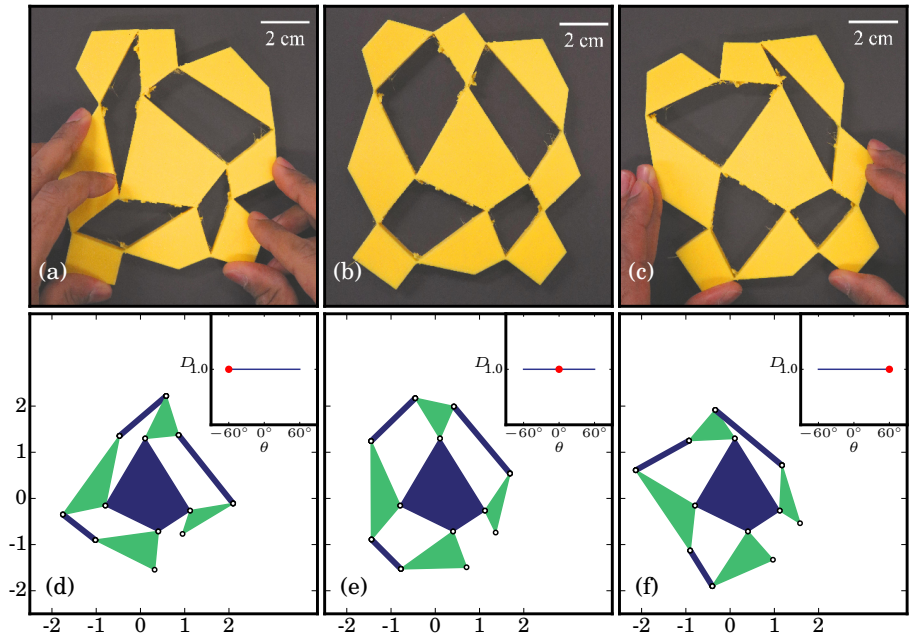


Figure 3.35: A 3D printed flexible unit cell in the states of (a) $\theta = -60°$, (b) $\theta = 0°$ and (c) $\theta = 60°$ of the base mechanism, which is shown in these states in (d-f) respectively. Inset shows the theoretical $D(\theta)$ curve. We observe good agreements between experiments and simulations, thus successfully modeling the soft deformation mode.

we pay special attention to the connecting-hinge thickness and keep it to a small value, determined by the resolution of the printer. These designs are then fed to a printer, which builds up a three-dimensional object by successively adding the printing material layer by layer. We use an elastic material for 3D printing, known by its commercial name *Filaflex* and print it with our *Felix Pro 1* printer.

3.9.1 Unit Cells

The design of a 3x3 flexible unit cell is simple to create. We start from a *precursor* mechanism generated by PSO in the neutral state ($\theta = 0°$) and complete the 3x3 network by 'fitting-in' the 'missing' polygon P_9 [Fig.

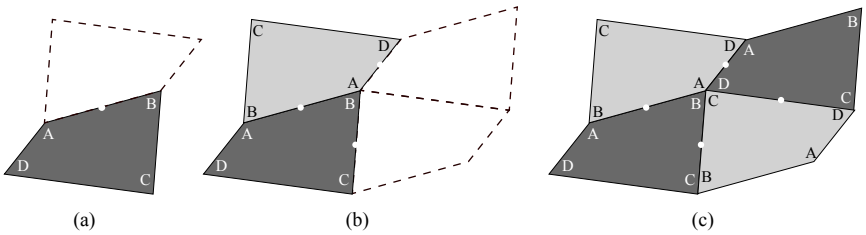


Figure 3.36: Quadrilaterals (simple) tessellate the plane.

3.12], joining the polygons P_6 and P_8. The corresponding edge length of P_9 being 1.0. The polygonal shape of the rigid elements at the *boundary* of the unit cell is not specified by the design, and we augment the shape of each element to a quadrilateral. This is done such that the quadrilaterals at the boundary do not collide throughout the internal hinging motion within the range of interest. In order to mimic the low energy deformation mode, we keep the designed minimum thickness of the connections joining two quadrilaterals together to be ≈ 0.45 mm. The resolution of our printer is ≈ 0.40 mm. Note that the final 3D printed samples have connections with thickness slightly greater than ≈ 0.45 mm.

An example of a 3D printed unit cell is shown in Fig. 3.35(b). There the scale of the sample is labeled. The out-of-plane thickness of the sample is ≈ 10 mm. The precursor mechanism is shown in the figure (e). The value of $\boldsymbol{f}$ is 4.5×10^{-7}. Fig. 3.35(a),(c) show the sample deformed in the $\theta = -60^\circ$ and $\theta = 60^\circ$ states, which are shown in the figure (d),(f) for the base mechanism. We observe a good agreement between the deformation patterns of 3D printed sample and the underlying mechanism.

3.9.2 Metatilings

We demonstrate a strategy to tile any arbitrary flexible unit cell into a regular tessellation, while preserving the internal soft mode. Firs, we note that any simple quadrilateral tesselates the plane [Fig. 3.36]. To obtain quadrilaterals, we augment the bar linkages P_1, P_3, P_7 and P_9 of the unit cell into triangles [Fig. 3.37(a,b)], whose free vertices form a virtual quadrilateral boundary. We then create a regular tesselation of this *augmented unit cell* by tiling it.The extreme deformed states of the tesselation i.e. $\theta = -60^\circ$ and $\theta = 60^\circ$ are shown in Fig. 3.37(c,d) respectively. Clearly, the tesselation retains

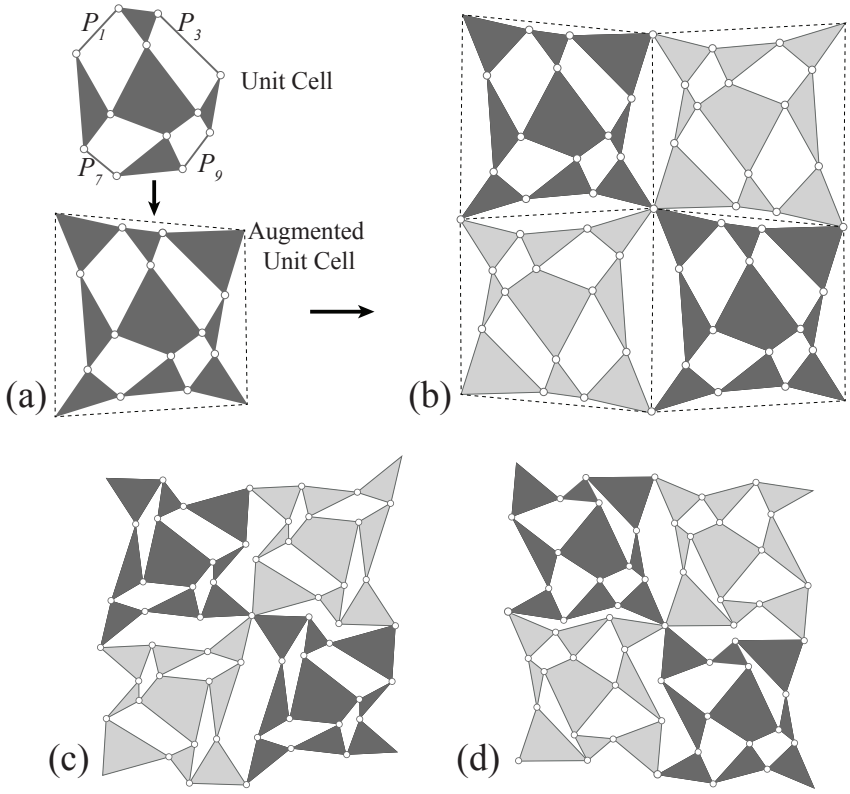
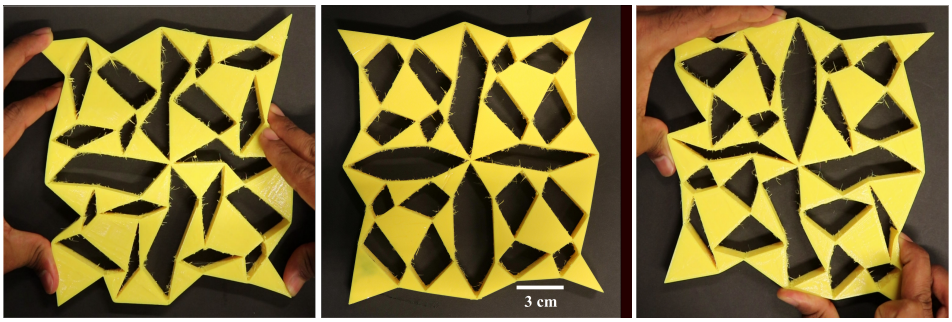


Figure 3.37: A general procedure to tile an arbitrary flexible 3x3 unit cell into a periodic tesselation, that we call a *Metatiling*. Note that the unit cell shown in this example is the same as in Fig. 3.35(b). We augment a 3x3 unit cell by transforming the bar linkages (labeled P_1, P_3, P_7 and P_9) into triangular elements. The virtual edges connecting the free corners of the augmented unit cell are shown in dashed lines. These edges form a quadrilateral tile. (b) A periodic tesselation consisting of the 3x3 augmented unit cell and copies rotated by 180° in-plane (shown in a lighter shade). The deformed states of the metatiling corresponding to: (c) $\theta = -60°$ and (d) $\theta = 60°$ configurations of the unit cell.

the deformation modes of its unit cell. We call such flexible tesselations *metatilings*.

By following this procedure, we design the metatiling of the unit cell



(a) **(b)** **(c)**

Figure 3.38: 3D printed metatiling of the unit cell shown in Fig. 3.35. The metatiling in its neutral state is in (b). In (a, c), we show the extreme deformed states of it, respectively, in which, the internal unit cell is at the $\theta = -60°$ and $\theta = 60°$ configurations. We can observe that the deformations in the real sample closely conform the ideal scenario [Fig. 3.37(c,d)].

shown in Fig. 3.35 and 3D print it. The tiling in the neutral state is shown in Fig. 3.38(b). The length scale of the sample is labeled. The designed thickness the sample (plane-normal) is 10 mm. The extreme deformed states are shown in Fig. 3.40(a,c). We observe a decent agreement with the deformed states expected for ideal hinges [Fig. 3.37(c,d)]

3.10 Discussion and Conclusion

In this chapter, we demonstrated an optimization-based methodology to design small generic yet flexible systems that exhibit low-energy deformations. The main strategy is to optimize the design of the underlying mechanism such that a curve that characterizes the internal motion, the $D(\theta)$ curve [§3.1] matches the desired target response $D_t(\theta)$. We began with formulating the optimization problem with the aim of obtaining a mathematical definition for the objective function in terms of the design variables that includes the penalized constraint handling as well [§3.4], and employed PSO to optimize for $D_t(\theta)$ [§3.5].

We focused on the optimal parameter setting of PSO in §3.6 and showed the effect of the parameters (ω, c_1, c_2) on the search behavior and distribution of the final solutions. In particular, we found out that for the current

problem, at $\omega = 0.25$, high values of c_2 and low values of c_1 are conditional [Eq. 3.29, §3.6.1], with the best-performing (c_1, c_2) set displaying a vertical downward shift with an increase in ω. We then demonstrated that the best-performing (c_1, c_2) set results in a log-normal distribution of the final solutions, whereas the worst-performing set results in a normal distribution [§3.6.2].

In §3.7, we utilized Sammon's mapping in order to effectively reduce the dimensionality of the search space from 24 dimensions and map the candidate solutions onto a 2D plane. The visualization of PSO done in this manner assisted us in verifying the main arguments of §3.6.1: suboptimal (c_1, c_2) settings can lead to premature convergence, divergence or no convergence at all, whereas correct settings impart a much higher global search capability to PSO.

In the main results section [§3.8], we showed the optimized designs and found a huge diversity in terms of the distinct possible solutions that satisfy our target criteria and are still generic, although we observe that the quality of solutions suffers as the genericness of the solutions increases (§3.8.2). We characterized PSO and showed that a 'good' PSO search results in the swarm particles getting trapped inside a deep local minimum while this is not the case with a 'bad' search (§3.8.1).

Finally in §3.9, we bring these computer designed structures to real life through 3D printing with flexible material and slender connections. We are able to closely model the ideal low-energy deformation mode. Additionally, we showed how to tile these flexible unit cells into a regular tessellation while retaining the original soft-mode.

Acknowledgments

The large batch of PSO simulations were performed availing the high-performance computing facilities offered by SURFsara, Amsterdam. We are especially thankful for their excellent technical support to facilitate efficient computing and data transfer. The work presented in this chapter is part of an industrial partnership programme (IPP) *'Computational Sciences for Energy Research (CSER)'* started in 2012, and funded by the Shell Global Solutions International B.V., the Netherlands Organization for Scientific Research (NWO) and the Foundation for Fundamental Research on Matter (FOM).