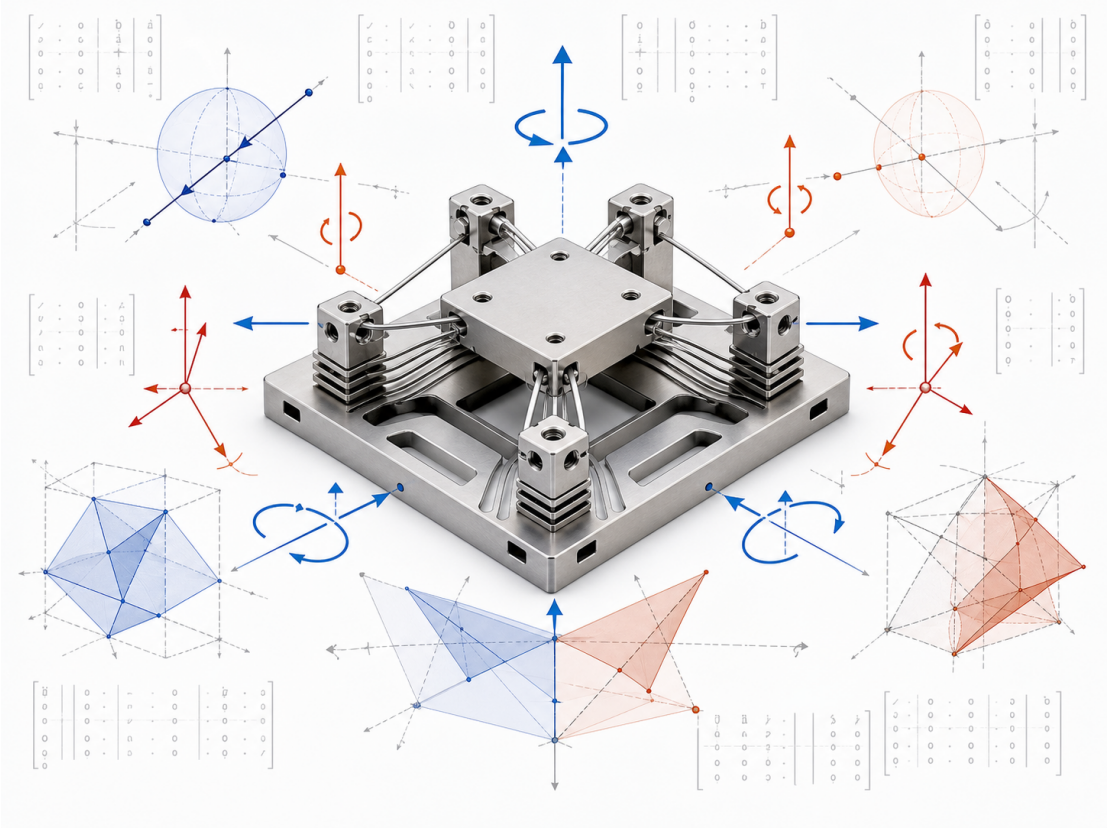
# Screw Theory Based Design of Compliant Mechanisms



Hai-Jun Su

July 2026

Draft text on screw-theory-based compliant mechanism design

# Preface

This book is written as a graduate text on a screw-theory-based design approach for compliant mechanisms. Its central premise is that compliant mechanism design can be organized around twist spaces, wrench spaces, reciprocity, coordinate transformations, rank calculations, and compliance matrices. A student should finish the book able to state a desired motion as a twist space, compute its reciprocal wrench space, recognize whether ordinary flexure primitives can realize the required constraints, assemble serial and parallel compliant modules, and derive the compliance or stiffness matrix of the resulting mechanism.

The material draws on exact constraint design, screw theory, compliant mechanism synthesis, symbolic compliance models, and computational examples. In particular, the text uses exact-constraint background material [2], flexure design guidance [4], screw-theory synthesis and mobility studies [8, 9, 10, 12, 14], and symbolic compliance and compliance-mobility criteria [11, 13]. The Freedom and Constraint Topology literature is cited as a graphical representation of the same reciprocal screw-system relationships [5, 6, 7]; the primary language of the book is the algebraic screw-theory formulation. The formulas and examples use a single screw-coordinate convention throughout unless a remark explicitly states otherwise.

The book is accompanied by computational and instructional materials. The project folder includes Wolfram/Mathematica scripts and notebooks for the algebraic calculations, while the course website at `https://haijunsu-osu.github.io/kinematics_ug/compliant_mechanisms_app/` provides an online entry point for the screw-theory-based compliant-mechanism design material [15]. Lecture videos and lecture slides are also available as companion teaching materials; the local draft folder includes NotebookLM-generated slides in `notebooklm_slides`. These materials are not a replacement for the derivations in the book. They are intended to let the reader rerun calculations, inspect intermediate matrices, and connect the written derivations to the course lectures.

Some figures in this draft are adapted or cropped from cited source materials to keep the derivations visually connected to the original mechanisms. Permissions should be checked before public redistribution.

# Contents

# Chapter 1

# Introduction

Compliant mechanisms achieve motion by elastic deformation rather than by sliding, rolling, or pin-jointed motion. This single fact changes the designer's problem. A conventional mechanism designer can often begin by counting rigid links and joints, because motion is concentrated in ideal pairs. A compliant mechanism designer must instead ask which deformations are intended, which deformations are forbidden, and which load paths make those distinctions reliable. The design task is therefore not only to obtain motion, but also to distribute compliance and stiffness so that the desired motion is easy and parasitic motion is difficult.

This distinction explains why exact constraint design and screw theory are natural foundations for compliant mechanism design. Exact constraint design asks a simple question: how many independent constraints are needed to locate a body without overconstraint? Blanding's treatment of exact constraint design makes this question concrete through patterns of constraints, freedoms, and reciprocal patterns [2]. Henein's flexure tutorial makes the same question practical for flexure mechanisms: flexures can deliver high precision, no backlash, no wear, and monolithic manufacture, but only when their compliant and stiff directions are used deliberately [4]. Screw theory supplies the algebra that joins these design instincts.

Graphical freedom-and-constraint charts are useful because they draw the same reciprocal screw-system relationships in a form that can be inspected visually [5, 6]. In this book they are treated as a geometric representation of the screw-theory approach rather than as a separate design theory. The screw-theory formulation is the working language: visual topologies become vector spaces, complementary patterns become reciprocal subspaces, and design synthesis becomes a sequence of rank and null-space calculations [8, 9, 10, 12].

The purpose of this book is to teach that algebraic screw-theory version. The goal is not to replace geometric intuition. The goal is to make the intuition checkable. Qualitative design statements become basis choices, rank calculations, reciprocal subspaces, and compliance matrices. The payoff is that a topology can be reasoned about geometrically and then checked algebraically.

## 1.1 Why Mobility Is Not Just a Count

For a rigid body in space, six small motions are possible: three rotations and three translations. In a rigid-body mechanism course, one often begins with mobility formulas such as the Chebyshev–Gruebler count. Those formulas are useful when all links are rigid, all joints are ideal, and all constraints are independent. Compliant mechanisms violate the spirit of those assumptions. A flexure is not an ideal joint; it is a deformable body with large stiffness in some directions and small stiffness in others. A stage might move mostly as intended even though every direction

has finite compliance. A redundant blade may improve load capacity without changing intended mobility. A symmetric set of flexures may introduce constraints that are algebraically dependent but mechanically valuable.

For that reason, mobility in compliant mechanisms should be treated first as a subspace and only second as an integer. The integer is the dimension of the dominant motion space, but the design meaning lives in the basis directions, the reciprocal constraints, and the relative stiffnesses. The screw-algebra mobility framework follows this view by building a library of flexure elements, deriving their twist and wrench matrices, decomposing a mechanism into modules, and then assembling mobility from the bottom up [10]. Zhang et al. add a quantitative mobility criterion by decomposing compliance matrices into eigentwists and eigenwrenches, then comparing scaled eigencompliances to decide which directions should be regarded as mobile [13]. These two views are complementary: screw algebra is the cleanest conceptual tool, while compliance decomposition is the bridge from ideal mobility to finite stiffness.

## 1.2 A Running View of Design

Throughout the book, a compliant mechanism is treated as a system connecting a stage to ground. The stage is the body whose motion matters. The ground is the body taken as fixed. Between them are flexure primitives, flexure joints, serial chains, parallel chains, and hybrid modules. Each primitive contributes either a motion description, a constraint description, or a compliance description. The designer's task is to assemble these descriptions into a mechanism-level model.

The formal algebra begins in Chapter 3, so this introduction deliberately stays at the design level. For now, it is enough to keep the main picture in mind: compliant mechanism design alternates between the freedom a stage should retain and the constraints that make all other motions expensive.

## 1.3 Organization of the Book

The chapters follow the design workflow. First, Chapter 3 builds the screw algebra: twists, wrenches, line screws, coordinate transformations, reciprocal spaces, and the rule of complementary patterns. This is the mathematical core of the book. Second, Chapter 4 uses the algebra to analyze serial, parallel, and hybrid compliant mechanisms. Serial mechanisms are most naturally assembled by adding motion spaces, while parallel mechanisms are most naturally assembled by adding constraint spaces. Third, Chapter 5 turns the direction of reasoning around. Instead of asking what a given mechanism can do, it asks what mechanism can realize a desired motion space. Fourth, Chapter 6 moves from ideal mobility to finite stiffness by deriving compliance and stiffness matrices and showing how they compose across topology.

This order is important. A student who begins directly with stiffness matrices may know how much a stage moves under a load but may not know why the matrix has that structure. A student who begins only from sketches may see attractive topologies but may not know how to automate or verify them. Screw algebra sits between geometric intuition and quantitative stiffness analysis. It tells us which directions should be compliant, which directions should be stiff, and which matrix entries have design meaning.

# Chapter 2

# The Constraint-Based Design Approach

Constraint-based design begins from a simple but powerful idea: a compliant mechanism is not defined by the motion it lacks, but by the motions it is deliberately prevented from taking. In that sense, the designer does not merely ask how to make a stage move. The designer asks which directions should remain free, which directions should be suppressed, and how to distribute those suppressions so that the intended motion is robust rather than accidental. This viewpoint is the geometric core of exact constraint design and flexure-mechanism design practice [2, 4].

A rigid body in space has six infinitesimal motions, but the useful question for a compliant mechanism is not how many of those motions exist in principle. The useful question is which of those six directions should be allowed to survive the design process. A stage that is meant to translate along one axis is not characterized by one free motion alone; it is characterized by five independent directions that must be blocked in a manner compatible with that translation. This is why exact constraint design is geometric before it is algebraic. One first imagines the motion space and the blocked space, and only then asks for a mechanism that realizes that picture.

## 2.1 Mobility of Compliant Mechanisms

The most intuitive way to read the design problem is to treat freedom as a geometric shape in the six-dimensional motion space of a rigid body. In the plane, the relevant picture has three small motions: one rotation and two translations. In space, the picture expands to three rotations and three translations. A compliant mechanism does not remove the need to think in these terms. It makes the thinking more urgent, because a flexure is never an ideal joint. It has preferred directions of compliance, directions that are strongly resisted, and directions that are only approximately suppressed. The designer therefore has to decide not only what the mechanism should do, but also which directions should carry the load and which directions should remain slack enough to permit motion.

It is useful to contrast this viewpoint with the standard mobility count for rigid-body mechanisms. In an ideal rigid-body mechanism, the links are treated as perfectly rigid and the joints are treated as exact kinematic constraints. If a mechanism has $n$ links including ground, $j$ joints, and the $i$th joint permits $f_i$ independent relative motions, the Gruebler–Kutzbach criterion gives the nominal mobility

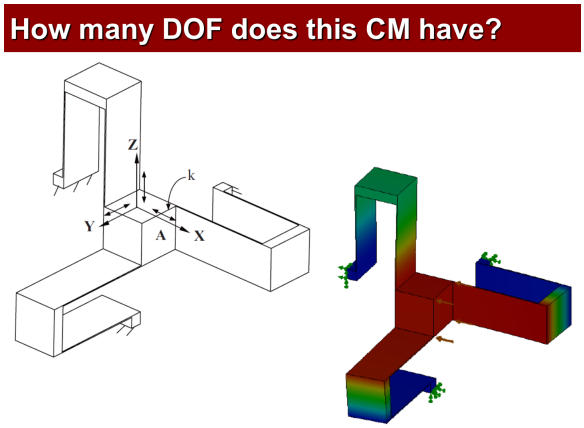
$$M = \lambda(n - 1 - j) + \sum_{i=1}^{j} f_i, \tag{2.1}$$

where $\lambda = 3$ for a planar mechanism and $\lambda = 6$ for a spatial mechanism [3]. Within the assumptions of ideal joints and independent constraints, the result is deterministic: the mobility is a property of the link-joint structure and its geometry, not a matter of how much force is applied.

Compliant mechanisms require a different interpretation. A compliant mechanism does not possess perfectly free and perfectly blocked directions in the same literal sense as an ideal linkage, because every elastic member has finite stiffness in every direction. Mobility is therefore a relative statement about the stiffness landscape of the mechanism. A direction with low stiffness is a direction in which the moving body can be displaced easily, so the designer treats it as a freedom. A direction with high stiffness is a direction in which motion is strongly resisted, so the designer treats it as constrained. The boundary between mobility and constraint is therefore set by stiffness ratios, loading requirements, allowable parasitic motion, and the precision target of the device. Constraint-based design keeps the geometric discipline of rigid-body kinematics, but replaces the binary language of ideal joints with the practical language of dominant compliant directions and dominant constrained directions.

That is the central advantage of constraint-based design. It turns mobility from an abstract count into a geometric choice. If the desired motion is a one-degree-of-freedom translation, then the stage is being designed around a five-dimensional complement of constraints. If the desired motion is a pure rotation about a prescribed center, then the constraint directions must be arranged so that the motion is compatible with that center and incompatible with parasitic translations. Figure 2.1 frames the contrast: a rigid-body mechanism invites a deterministic mobility count, while a compliant mechanism must first be read by the constraints and stiffness directions imposed by its flexures. The idealization is strict, but the design intuition is durable: the mechanism works because the constraints are placed where they do not fight the intended motion.

**(a)** Rigid-body mechanism.




**(b)** Compliant mechanism.

Figure 2.1: Mobility interpretation for rigid-body and compliant mechanisms. A rigid-body platform is analyzed by counting ideal links and joints, while a compliant mechanism should be analyzed by identifying which constraints and stiffness directions its flexures impose on the moving body.

## 2.2 Flexures as Intentional Constraints

Constraint-based design treats flexures as constraint elements first and motion elements second. This is the right order of thought for synthesis. A wire flexure is the clearest example. Its dominant mechanical effect is to resist axial extension, so in the local idealization it behaves like a single

translational constraint. The flexure is useful not because it is weak, but because it is selective. It resists one direction strongly while leaving other small motions available through bending and rotation.

An ideal sheet flexure is richer. It can be idealized as providing two dominant translational constraints and one dominant rotational constraint, while remaining compliant in the out-of-plane translation and two bending/torsional rotations. Figure 2.2 contrasts this primitive with the simpler wire constraint. The useful idealization is binary rather than moderate: in the directions assigned to constraint, the sheet is treated as rigid; in the complementary directions, it is treated as compliant. The important point is that a sheet flexure is not just a softer wire. It occupies a different place in the constraint taxonomy, because its geometry suppresses a different pattern of motion and therefore contributes a different complement to the stage freedom.

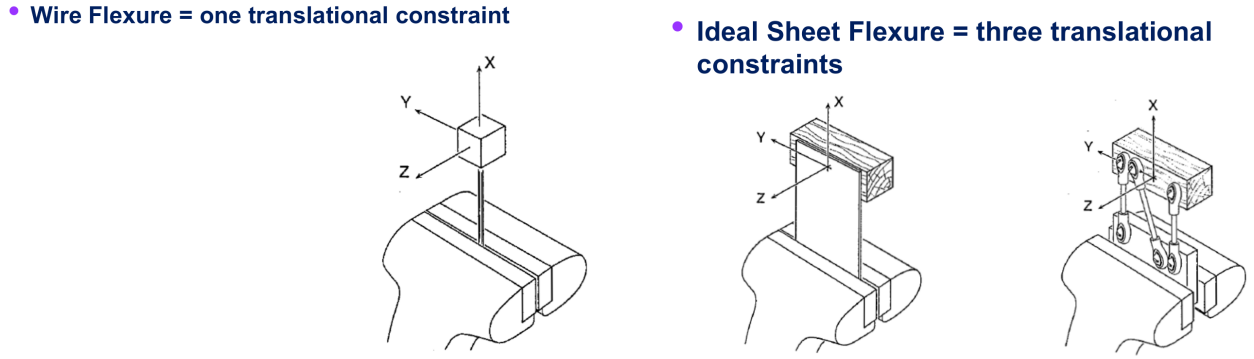



Figure 2.2: Basic flexure primitives interpreted as constraint elements. A wire flexure contributes one dominant translational constraint, while an ideal sheet flexure contributes two dominant translational constraints and one dominant rotational constraint.

The mechanics behind this classification can be seen from elementary beam estimates for a thin sheet flexure [2]. Figure 2.3 shows the geometric parameters used in the estimate.

Let the sheet have length $l$, width $w$, thickness $t$, Young's modulus $E$, and shear modulus $G$, with the local $x$-axis along the sheet length, $y$ along the in-plane width, and $z$ through the thickness. The geometric message of the sketch is simple: in-plane loading stretches or bends a comparatively large section, while out-of-plane loading bends the thin dimension. Rotation about the sheet normal is also resisted strongly because the sheet must deform in its broad plane. These geometric separations are what make the ideal sheet flexure a useful constraint primitive. The cross-sectional area and second moments used in the estimate are defined in (2.2).

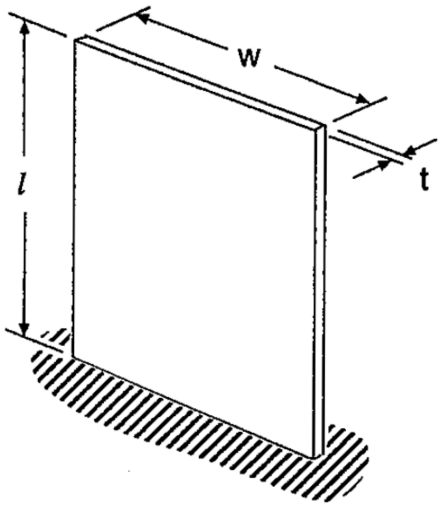



Figure 2.3: Dimensioned sheet flexure for the beam-theory stiffness estimate.

$$A = wt, \qquad I_y = \frac{wt^3}{12}, \qquad I_z = \frac{tw^3}{12}. \tag{2.2}$$

For small end deflections, the three translational stiffnesses in (2.3) show the strong axial stiffness and the much smaller bending stiffnesses.

$$k_x \simeq \frac{EA}{l} = \frac{Ewt}{l}, \qquad k_y \simeq \frac{3EI_z}{l^3} = \frac{Etw^3}{4l^3}, \qquad k_z \simeq \frac{3EI_y}{l^3} = \frac{Ewt^3}{4l^3}. \tag{2.3}$$

The corresponding end rotational stiffnesses in (2.4) complete the six-direction stiffness picture.

$$k_{\theta_x} \simeq \frac{GJ}{l} \approx \frac{\eta G w t^3}{l}, \qquad k_{\theta_y} \simeq \frac{EI_y}{l} = \frac{Ewt^3}{12l}, \qquad k_{\theta_z} \simeq \frac{EI_z}{l} = \frac{Etw^3}{12l}, \tag{2.4}$$

where $J \approx \eta w t^3$ is the torsion constant of a thin rectangular section and $\eta$ is a geometric constant of order one. The numerical constants depend on the exact end condition, but the stiffness separation does not. The key ratios are collected in (2.5).

$$\frac{k_x}{k_z} = 4\left(\frac{l}{t}\right)^2, \qquad \frac{k_y}{k_z} = \left(\frac{w}{t}\right)^2, \qquad \frac{k_{\theta_z}}{k_{\theta_y}} = \left(\frac{w}{t}\right)^2. \tag{2.5}$$

For the slender proportions typical of sheet flexures, $l/t$ and $w/t$ are large. For example, (2.5) gives $k_x/k_z \approx 3.6 \times 10^5$ when $l/t \approx 300$. This ratio does not depend on $E$; changing the sheet material changes both stiffnesses, but it does not change the geometric separation caused by stretching in the plane of the sheet versus bending out of that plane. Thus the sheet is mechanically stiff in $x$, $y$, and $\theta_z$, and compliant in $z$, $\theta_x$, and $\theta_y$, as summarized in Table 2.1. This is the stiffness justification for treating the ideal sheet flexure as a constraint pattern that blocks two translations and one rotation while leaving the complementary three motions available.

Table 2.1: Binary stiffness idealization of a sheet flexure in six small-motion directions. The sheet is stiff for in-plane translations and rotation about the sheet normal, and compliant for out-of-plane translation and the two rotations that bend or twist the thin sheet [2].

| Direction | Stiff | Compliant |
|---|---|---|
| $x$ | ✓ | |
| $y$ | ✓ | |
| $z$ | | ✓ |
| $\theta_x$ | | ✓ |
| $\theta_y$ | | ✓ |
| $\theta_z$ | ✓ | |

Once the primitives are understood this way, the design logic becomes much clearer. Flexures are not catalogued merely by shape. They are catalogued by the constraint directions they contribute. A mechanism architecture is then built by combining those directions so that the collection of constraints matches the intended motion space and no more. This is the point where the language of exact constraint design becomes practically valuable. A good topology is one in which every constraint has a geometric purpose.

## 2.3 Counting Mobility from Constraint Patterns

The planar exact-constraint examples are deliberately simple because they teach the geometric habit that later becomes screw-system algebra. In planar exact-constraint design, a rigid body has three infinitesimal freedoms: rotation about $z$ and translations in $x$ and $y$. A translational constraint is drawn as a line of constraint acting on the moving body. If the constrained direction is $\boldsymbol{u} = (u_x, u_y)^\mathsf{T}$ and the line passes through $\boldsymbol{r} = (x, y)^\mathsf{T}$, the small velocity of that point in the constrained direction is

$$\boldsymbol{u}^\mathsf{T}\left(\begin{bmatrix}\delta_x\\ \delta_y\end{bmatrix} + \theta_z\begin{bmatrix}-y\\ x\end{bmatrix}\right) = \theta_z(xu_y - yu_x) + \delta_x u_x + \delta_y u_y. \tag{2.6}$$

The constraint condition is that this scalar vanish. Thus a single translational constraint removes one planar freedom and leaves a two-dimensional family of allowed small motions, as visualized in Fig. 2.4.


Figure 2.4: Planar translational constraints in exact-constraint design. A single line constraint blocks one velocity component of the moving body, so the mobility count starts from the constraint pattern rather than from a joint name.

**Example 2.1** (One translational constraint in the plane)**.** *If the constraint line passes through the origin and blocks velocity along $x$, the constraint condition is $\delta_x = 0$. The body may still translate in $y$, and it may rotate about the origin because the constrained point has no $x$-velocity under that rotation. The mobility count is therefore $m = 2$, but the geometric content is more specific than the number: the remaining freedom is the family generated by $y$-translation and rotation about a point on the constraint line.*

Two translational constraints reduce the planar body from three freedoms to one freedom when their lines are independent. If the two constraint lines intersect, the surviving motion is a rotation about their intersection. Place the intersection at the origin and choose two orthogonal constraints. The first removes the $x$-velocity of the origin and the second removes the $y$-velocity of the origin, so the two conditions are

$$\delta_x = 0, \qquad \delta_y = 0. \tag{2.7}$$

The only remaining small motion is rotation about the intersection point.

**Example 2.2** (A planar rotational freedom)**.** *The two intersecting constraints in Fig. 2.5 are independent. The mobility count is $m = 3 - 2 = 1$, but the important conclusion is the location of the one freedom. The body is not merely one-DOF; it is one-DOF about the intersection of the two constraint lines. Moving either line changes the instant center and therefore changes the design intent.*

If the two translational constraints are parallel rather than intersecting, their intersection moves to infinity and the surviving motion becomes a translation. With two horizontal constraint lines located at $y = h$ and $y = -h$, both blocking $x$-velocity, the two point-velocity conditions are

$$\delta_x - h\theta_z = 0, \qquad \delta_x + h\theta_z = 0. \tag{2.8}$$

For $h \neq 0$, these conditions imply $\theta_z = 0$ and $\delta_x = 0$, leaving only $\delta_y$. In the exact-constraint picture, parallel constraint lines create an instant center at infinity; geometrically, that is a pure translation perpendicular to the constraint lines.

**Example 2.3** (A planar translational freedom)**.** *The parallel-constraint pattern in Fig. 2.6 also has two independent constraints and therefore also has mobility one. It is nevertheless a different one-DOF mechanism from the intersecting case. The count alone cannot distinguish a rotation about a finite point from a translation; the geometry of the constraint lines supplies that distinction.*

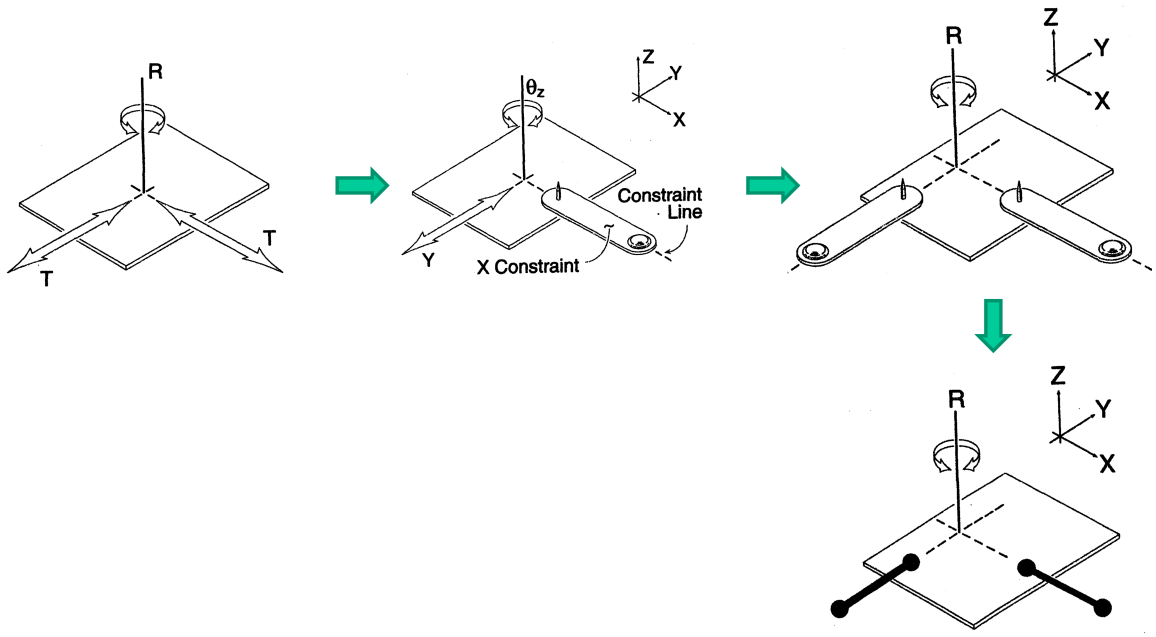



Figure 2.5: Designing a planar rotation with two intersecting translational constraints. The two constraint lines meet at the instant center, so the only allowed motion is rotation about that point.

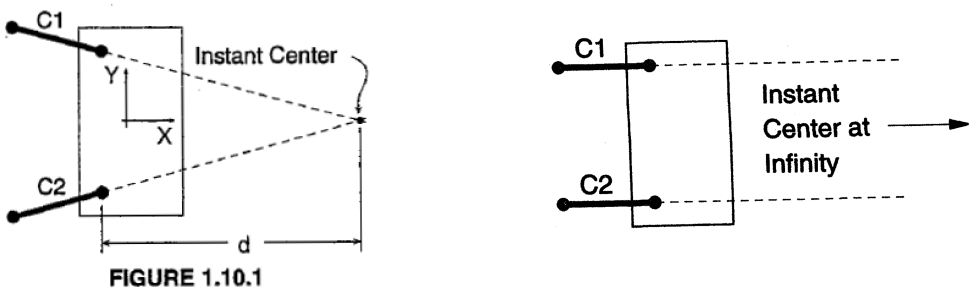



Figure 2.6: Designing a planar translation with two parallel translational constraints. Parallel constraint lines meet at an instant center at infinity, so the allowed one-DOF motion is translation.

The same reasoning can be organized as a small pattern table. No line constraint leaves all three planar freedoms. One line constraint leaves two freedoms. Two independent line constraints leave one freedom, whose geometric type depends on whether the lines intersect at a finite point or at infinity. Three independent line constraints fully locate the planar body. Degenerate cases are not exceptions to the method; they are geometric dependencies in the constraint pattern.

Serial composition gives the other basic design move. Two one-DOF translation modules can be cascaded so that their motion spaces add. In the drafting-head example, one parallelogram-like module supplies one translation of an intermediate body, while a second module supplies a transverse translation of the final body. If the first module contributes $\text{span}\{[0,1,0]^{\mathsf{T}}\}$ and the second contributes $\text{span}\{[0,0,1]^{\mathsf{T}}\}$, then the final planar motion space is

$$\mathcal{F}_{\text{serial}} = \text{span}\left\{ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}. \tag{2.9}$$

The orientation is still constrained by each module, but the two translational freedoms accumulate through the serial chain.

**Example 2.4** (Cascaded translations as a serial chain)**.** *The cascaded device in Fig. 2.7 should not be analyzed as a single mysterious two-DOF joint. It is a serial composition of two one-DOF translation modules. The first module moves the intermediate body, and the second module moves the drafting head relative to that intermediate body. The final output has two independent translations even though each stage individually has only one.*

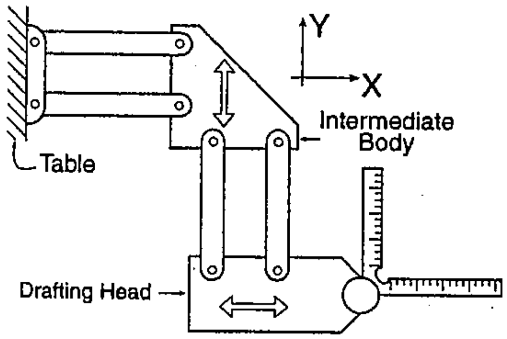



Figure 2.7: Cascading two translational mechanisms to obtain planar $x$-$y$ translation.

The spatial versions of these examples use the same logic with six body freedoms instead of three. The left panel of Fig. 2.8 shows this spatial reading of constraint devices and wire-supported stages.

A single wire-like translational constraint removes one instantaneous velocity component and leaves a five-dimensional family of small motions. A set of six independent line constraints can locate a rigid body completely. If some lines intersect, become parallel, or form a symmetric dependent pattern, the body retains mobility. This is the spatial version of the same visual constraint-counting logic.

**Example 2.5** (Spatial constraint counting)**.** *For a spatial stage, the ideal mobility is $m = 6 - c$, where $c$ is the number of independent constraint directions. Four independent wire constraints leave a nominal two-dimensional motion family. Adding more wires may reduce the mobility, but only if the added constraints are independent of the constraints already present. The geometric caution is the same as in the plane: six physical constraints do not necessarily mean zero mobility unless their line constraints are independent. Figure 2.8 illustrates why a six-constraint spatial layout still has to be read as a pattern of independent lines.*

These examples are intentionally elementary, but they establish the geometric foundation of the whole book. Constraint-based design first asks what line constraints are present, which constraints are independent, and what complementary motion pattern remains. Screw theory will express the same question as a reciprocal-space calculation.

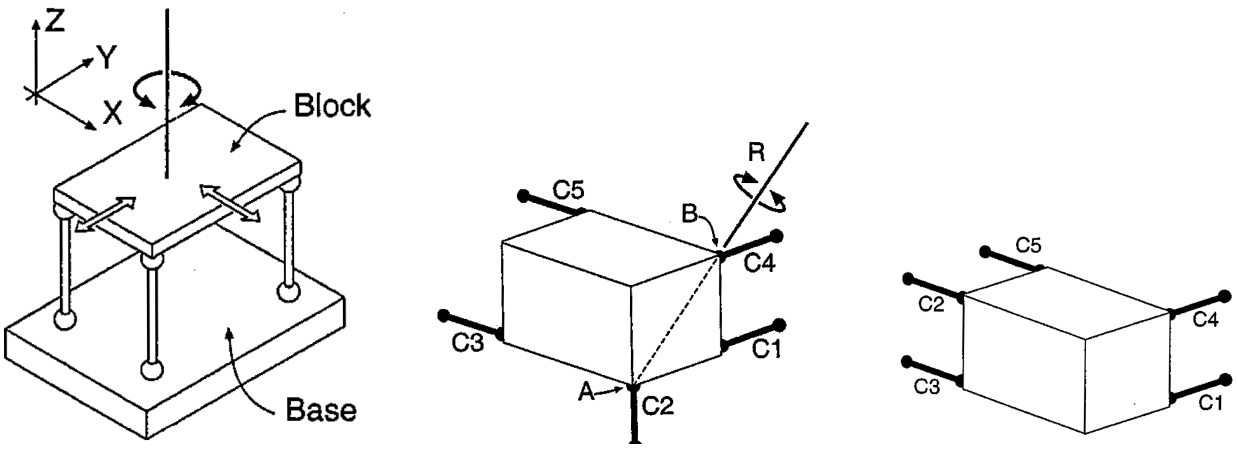



Figure 2.8: Spatial constraint-counting examples. The left panel shows a wire-supported stage whose wire-like elements are read as translational constraint lines; the middle and right panels show how visual exact-constraint counting extends from planar line constraints to spatial line-constraint arrangements.

## 2.4 Graphical Representations of Screw Systems

Graphical screw-system representations make the complementary relation between freedoms and constraints visible before the algebra is written. The Freedom and Constraint Topology or FACT approach is one well-known form of this graphical representation [5, 6]. In the present book, it is used only in that role: as a geometric picture of the same twist and wrench spaces that screw theory computes.

The chart in Fig. 2.9 organizes graphical representations of screw systems by degree-of-freedom level. Its value here is pedagogical: it shows, in a compact visual form, the same reciprocal structure that Chapter 3 writes as linear subspaces.

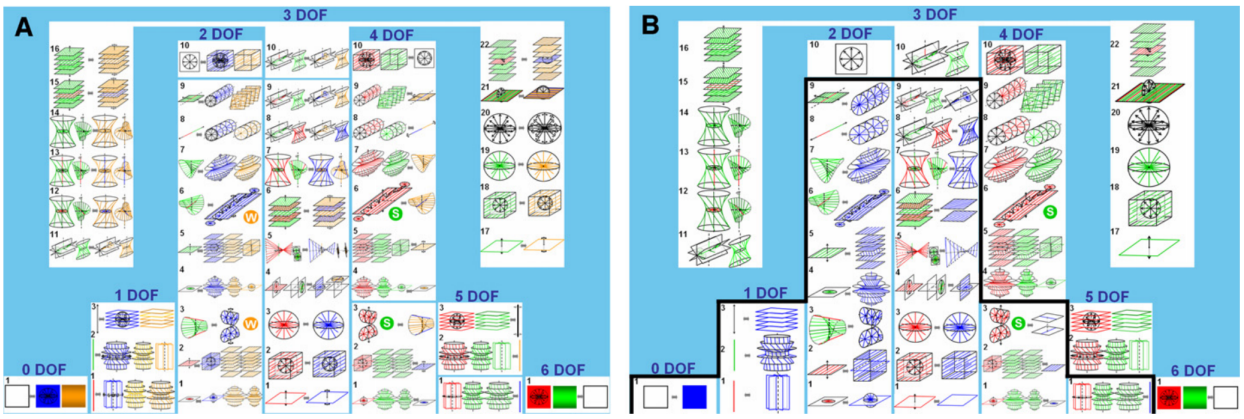



Figure 2.9: Graphical chart for screw-system representations [7]. The columns organize screw-system types by degree-of-freedom level, making complementary freedom and constraint spaces visible before the reciprocal-space calculation is written.

The appeal of the graphical view is that it makes the reciprocal relation between motion and constraint visible. A designer can see, in a single sketch, whether a set of flexure branches is likely


Figure 2.10: A constraint-space picture and a freedom-space picture describe the same design from opposite sides. Screw theory writes this complement as reciprocal twist and wrench spaces.

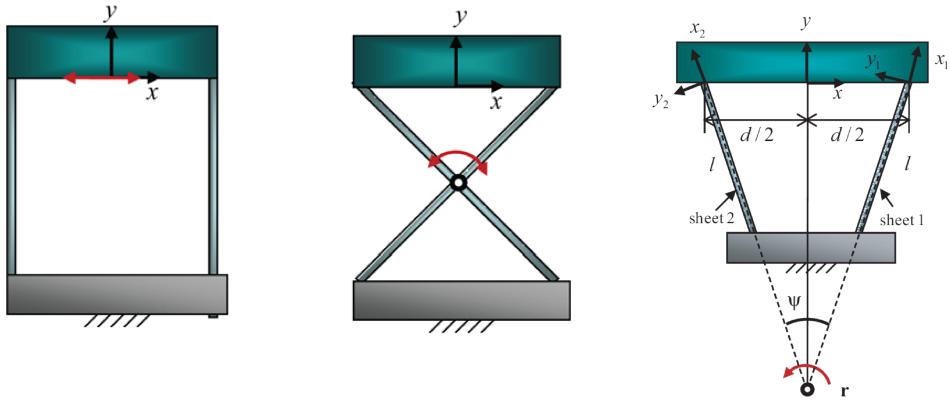



Figure 2.11: Parallel-connected flexures. In a parallel arrangement, constraints from separate branches act on the same moving body, so the stage freedom is the complement of the combined branch constraints. This geometric idea becomes the wrench-space rank calculation in later chapters.

to block the intended motion or whether it is arranged to support it. Figure 2.10 gives the same lesson in the elementary language of constraint and freedom spaces. That visual reasoning is useful, but it is not the final analysis. The screw-theory calculation supplies the reciprocal-space and rank checks that decide whether the sketch is correct.

Once flexure elements are placed in parallel, the global stage behavior is governed by how their constraint directions combine. This is why the design process starts to look like a problem of pattern matching. One asks whether the branch constraints are complementary, whether they are redundant in a useful way, and whether they leave the desired mobility untouched. Figure 2.11 is the geometric prototype for the wrench-space rank calculation used later.

The algebraic version begins in Chapter 3. The present chapter has stayed deliberately geometric: it asks the reader to see constraint patterns, complementary freedoms, and the rank consequences of line arrangements before introducing screw coordinates.

## 2.5 Reading Designs from Their Constraints

The practical skill that follows from this chapter is the ability to read a mechanism backwards. A designer should be able to look at a proposed compliant stage and identify the intended freedom space, the constraint space that supports it, and the likely failure modes if either space has been chosen poorly. If a design is supposed to translate but shows a constraint branch that directly blocks that translation, the geometry already signals a problem. If a design is supposed to rotate

about a point but the constraints are arranged as though the stage were a rigid prism, the reciprocal picture is again wrong. The advantage of constraint-based design is that such errors are visible before detailed stiffness calculations are attempted.

The deeper lesson is that compliant mechanism synthesis is not a search for flexibility in the ordinary sense. It is a search for the right complement of flexibility and resistance. Exact constraint design supplies the geometric discipline for that search, and screw theory supplies the algebra that makes it computable. The purpose is to make the intended motion obvious and the unintended motion expensive.

# Chapter 3

# Screw Theory and Algebra

Chapter 2 built the geometric picture first. Constraint-based design asks what motions should remain free and what constraints should make all other motions expensive. Screw theory enters precisely at the point where this geometric intuition must become checkable. It is the algebraic form of the design approach: a twist records an infinitesimal motion, a wrench records a constraint or load, and reciprocity records whether the constraint does virtual work on that motion.

This reciprocal relation is what allows a designer to move cleanly from a sketch to a synthesis calculation. In the algebraic formulation used here, freedom patterns, constraint patterns, and flexure-element libraries are represented with screw coordinates and assembled through linear algebra [8, 10]. The conceptual move is important. Screw theory does not replace geometric reasoning. It records the geometry in a form that can be computed, transformed, and checked.

This is why the statement that screw theory is the algebraic form of constraint-based design is literal rather than metaphorical. A graphical chart can help the designer see a candidate freedom or constraint pattern, but screw theory tells the designer whether that pattern is reciprocal to the desired motion and whether the chosen flexure primitives can realize it. Geometry gives intuition; screw algebra gives proof. The paired viewpoint is simple: a general freedom is represented by a twist, and a general constraint is represented by a wrench. This chapter develops those definitions once, then uses them as the algebraic foundation for mobility analysis, mobility synthesis, and stiffness modeling.

## 3.1 Screw Coordinates for Twists and Wrenches

In the coordinate convention used throughout this book, a twist is the six-vector

$$\hat{T} = \begin{bmatrix} \boldsymbol{\Omega} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \theta_x & \theta_y & \theta_z & \delta_x & \delta_y & \delta_z \end{bmatrix}^{\mathsf{T}}, \tag{3.1}$$

where the upper vector is the infinitesimal angular part and the lower vector is the linear velocity of the chosen reference point. A wrench is written in the dual force–moment form

$$\hat{W} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{M} \end{bmatrix} = \begin{bmatrix} F_x & F_y & F_z & M_x & M_y & M_z \end{bmatrix}^{\mathsf{T}}, \tag{3.2}$$

where $\boldsymbol{F}$ is force and $\boldsymbol{M}$ is moment about the same reference point.

The axis form of a screw is often the clearest way to connect the algebra to geometry. We use the column-vector convention adopted throughout this book [8]. Let a twist axis have unit direction $\boldsymbol{s}$, let $\boldsymbol{c}$ be the position vector of any point on the twist axis, let $\omega$ be the angular-speed magnitude, and let $v$ be the translational velocity along the axis. The twist pitch is

$$p = \frac{v}{\omega}, \tag{3.3}$$

when $\omega \neq 0$, and the twist is written as

$$\hat{T} = \begin{bmatrix} \boldsymbol{\Omega} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} \omega \boldsymbol{s} \\ \boldsymbol{c} \times (\omega \boldsymbol{s}) + v \boldsymbol{s} \end{bmatrix}. \tag{3.4}$$

The term $v\boldsymbol{s} = p\boldsymbol{\Omega}$ is the translational velocity along the screw axis. A pure rotation has $p = 0$, while a pure translation is the limiting case $\omega = 0$ and $p = \infty$, interpreted by keeping the translational part finite.

Similarly, let a wrench axis have unit direction $\boldsymbol{u}$, let $\boldsymbol{r}$ be the position vector of any point on the wrench axis, let $f$ be the force magnitude, and let $m$ be the partial moment along the axis. The wrench pitch is

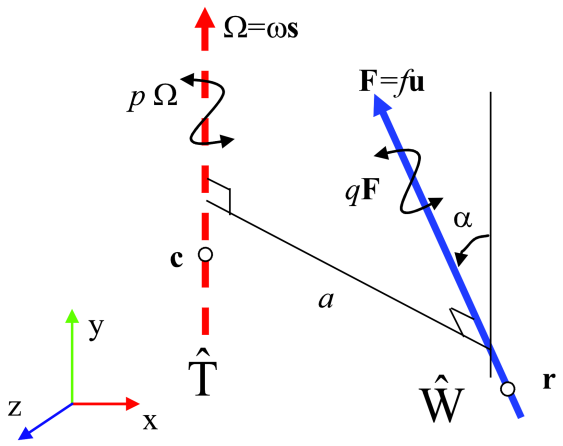



Figure 3.1: Screw parameters following the convention in [8]. The twist $\hat{T} = (\boldsymbol{\Omega} \mid \boldsymbol{V})$ uses axis direction $\boldsymbol{s}$, point $\boldsymbol{c}$, angular speed $\omega$, axial velocity $v$, and pitch $p = v/\omega$. The wrench $\hat{W} = (\boldsymbol{F} \mid \boldsymbol{M})$ uses axis direction $\boldsymbol{u}$, point $\boldsymbol{r}$, force magnitude $f$, axial moment magnitude $m$, and pitch $q = m/f$. The axes have normal distance $a$ and skew angle $\alpha$.

$$q = \frac{m}{f}, \tag{3.5}$$

when $f \neq 0$, and the wrench is written as

$$\hat{W} = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{M} \end{bmatrix} = \begin{bmatrix} f \boldsymbol{u} \\ \boldsymbol{r} \times (f \boldsymbol{u}) + m \boldsymbol{u} \end{bmatrix}. \tag{3.6}$$

The term $m\boldsymbol{u} = q\boldsymbol{F}$ is the free couple along the wrench axis. A pure force has $q = 0$, while a pure couple is the limiting case $f = 0$ and $q = \infty$, interpreted by keeping the moment part finite. Figure 3.1 collects these twist and wrench parameters in one geometric diagram.

In compliant mechanism design, the most common basis twists are the principal rotations and translations

$$\hat{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \quad \hat{R}_y = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \quad \hat{R}_z = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \tag{3.7}$$

and

$$\hat{P}_x = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \quad \hat{P}_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{\mathsf{T}}, \quad \hat{P}_z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}}. \tag{3.8}$$

The notation $R$ and $P$ echoes the usual rotational and prismatic freedom terminology.

**Definition 3.1** (Freedom space)**.** *A freedom space $\mathcal{F}$ is a linear subspace of $\mathbb{R}^6$ spanned by allowable twists. If $\boldsymbol{T} = [\hat{T}_1 \ \hat{T}_2 \ \cdots \ \hat{T}_n]$ has independent columns, then*

$$\mathcal{F} = \operatorname{range}(\boldsymbol{T}), \qquad \dim \mathcal{F} = \operatorname{rank}(\boldsymbol{T}) = n. \tag{3.9}$$

*Any small motion allowed by the design model can be written as $\hat{T} = \boldsymbol{T}\boldsymbol{a}$ for some coordinate vector $\boldsymbol{a}$.*

**Definition 3.2** (Constraint space)**.** *A constraint space $\mathcal{C}$ is a linear subspace of $\mathbb{R}^6$ spanned by constraint wrenches. If $\boldsymbol{W} = [\hat{W}_1\ \hat{W}_2\ \cdots\ \hat{W}_m]$ has independent columns, then*

$$\mathcal{C} = \operatorname{range}(\boldsymbol{W}), \qquad \dim \mathcal{C} = \operatorname{rank}(\boldsymbol{W}) = m. \tag{3.10}$$

*The columns of $\boldsymbol{W}$ are the independent virtual-work restrictions imposed on the stage.*

In compliant mechanisms, flexure primitives enter these spaces after an idealization step. A wire is usually modeled as one axial line-force constraint, whereas a sheet or blade may contribute several independent constraints. Figure 3.2 shows the corresponding wire and sheet idealizations. The algebraic calculation starts only after that mechanics judgment has been made.

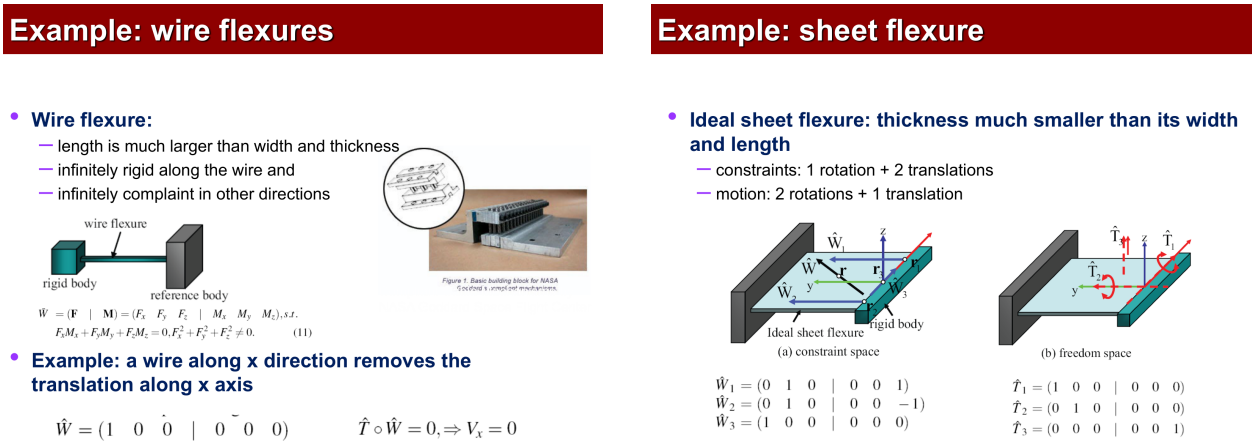



Figure 3.2: Idealized models for wire and sheet flexures. The wire example emphasizes that a slender member removes axial translation and is modeled algebraically by one line-force wrench. The sheet-flexure example emphasizes that a primitive's apparent geometry must be converted into a precise motion or constraint basis before analysis.

## 3.2 Reciprocity and Complementary Patterns

The reciprocal product between a twist and a wrench is

$$\hat{T} \circ \hat{W} = \boldsymbol{V}^{\mathsf{T}}\boldsymbol{F} + \boldsymbol{\Omega}^{\mathsf{T}}\boldsymbol{M} = \hat{T}^{\mathsf{T}}\boldsymbol{\Delta}\hat{W}, \qquad \boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}. \tag{3.11}$$

If this scalar is zero, the wrench is reciprocal to the twist. In virtual-work language, the wrench does no work on that infinitesimal motion. In design language, the constraint does not prevent that freedom.

For two nonzero screw axes with angular magnitude $\omega$, force magnitude $f$, pitches $p$ and $q$, skew angle $\alpha$, and normal distance $a$, substituting the axis definitions gives

$$\begin{aligned} \hat{T} \circ \hat{W} &= (\boldsymbol{F}^{\mathsf{T}}\boldsymbol{V} + \boldsymbol{M}^{\mathsf{T}}\boldsymbol{\Omega}) \\ &= (fv + m\omega)\, \boldsymbol{s}^{\mathsf{T}}\boldsymbol{u} + f\omega(\boldsymbol{c} - \boldsymbol{r})^{\mathsf{T}}(\boldsymbol{s} \times \boldsymbol{u}) \\ &= f\omega\left[(p + q)\cos\alpha - a\sin\alpha\right]. \end{aligned} \tag{3.12}$$

This expression classifies the main reciprocity cases. A pure translation is always reciprocal to a pure couple, because neither screw has the component needed to do virtual work on the other. If

one screw is pure and the other is not, reciprocity requires the directions to be perpendicular. When both screws have finite nonzero axes, the bracketed expression controls the condition. If $p+q=0$, including the common zero-pitch case $p=q=0$, reciprocity requires the axes to be coplanar, which appears geometrically as intersecting axes or parallel axes. If the axes are perpendicular, $\cos\alpha=0$, reciprocity can occur only when $a=0$, so the axes must intersect. Figure 3.3 turns this reciprocal condition into the complementary-pattern picture used throughout the book.

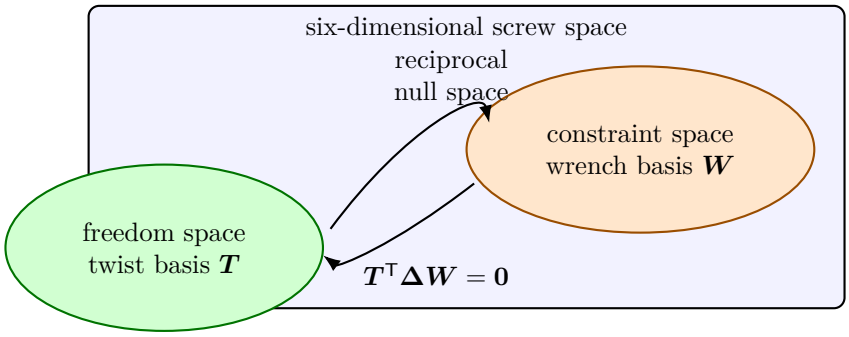



Figure 3.3: The algebraic form of complementary patterns. A freedom space is a span of twists, a constraint space is a span of wrenches, and the two are complementary when all reciprocal products vanish.

**Principle 3.1** (Rule of complementary patterns)**.** *For an ideal rigid stage in spatial motion, the freedom space and constraint space are complementary reciprocal spaces. If* $\mathcal{F}=\text{range}(\boldsymbol{T})$*, then the compatible constraint space is*

$$\mathcal{C}=\{\,\hat{W}\in\mathbb{R}^6 \mid \boldsymbol{T}^{\mathsf{T}}\boldsymbol{\Delta}\hat{W}=\mathbf{0}\,\}. \tag{3.13}$$

*If* $\mathcal{C}=\text{range}(\boldsymbol{W})$*, then the compatible freedom space is*

$$\mathcal{F}=\{\,\hat{T}\in\mathbb{R}^6 \mid \boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Delta}\hat{T}=\mathbf{0}\,\}. \tag{3.14}$$

*When the mechanism is exactly constrained in the ideal model, the dimensions satisfy* $\dim\mathcal{F}+\dim\mathcal{C}=6$.

This rule is the algebraic version of complementary freedom and constraint patterns. Graphical charts can present the same relationship visually, but screw algebra computes it directly with null spaces. The matrix calculation is therefore the primary design test: a candidate constraint set is acceptable only when it lies in the reciprocal space of the desired twists and has the required rank.

**Example 3.1** (Reciprocal space of a pure translation)**.** *Let the desired freedom space be one pure translation along* $x$*,*

$$\boldsymbol{T}=\hat{P}_x=\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}^{\mathsf{T}}. \tag{3.15}$$

*A reciprocal wrench* $\hat{W}=[F_x,F_y,F_z,M_x,M_y,M_z]^{\mathsf{T}}$ *must satisfy*

$$\hat{P}_x^{\mathsf{T}}\boldsymbol{\Delta}\hat{W}=F_x=0. \tag{3.16}$$

*Therefore the reciprocal constraint space consists of all wrenches with no* $x$*-force component. A mechanism that is intended to translate in* $x$ *must not include a dominant constraint wrench with* $F_x\neq 0$*. To realize the full five-dimensional reciprocal constraint space with physical flexures, the designer must choose line forces, couples, or flexure limbs whose wrenches span five independent directions inside this subspace.*

## 3.3 Line Screws and Wire Flexures

Line screws are central to wire-flexure synthesis because the ideal wire flexure contributes a pure force along its axis. To state the realizability question precisely, it is useful to distinguish a single line screw from a screw system that can be spanned by line screws. A general screw

$$\hat{S} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} \tag{3.17}$$

is a line screw when it can be written as

$$\hat{S} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{p} \times \boldsymbol{U} \end{bmatrix} \tag{3.18}$$

for some point $\boldsymbol{p}$ on the line. The vector $\boldsymbol{U}$ gives the line direction and $\boldsymbol{p} \times \boldsymbol{U}$ is the moment of that directed line about the origin. Equivalently,

$$\boldsymbol{U}^{\mathsf{T}}\boldsymbol{V} = 0, \qquad \boldsymbol{U} \neq \boldsymbol{0}. \tag{3.19}$$

This definition covers the two cases most often used in compliant mechanism design: a pure-force wrench along a wire axis and a zero-pitch rotational twist about a revolute axis.

Now let $\mathcal{S}$ be a rank-$m$ screw system with basis matrix

$$\boldsymbol{S} = \begin{bmatrix} \hat{S}_1 & \hat{S}_2 & \cdots & \hat{S}_m \end{bmatrix} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix}, \qquad \mathcal{S} = \operatorname{range}(\boldsymbol{S}). \tag{3.20}$$

Following the terminology in [9], the line variety of this system is the subset of all screws in $\mathcal{S}$ that also satisfy the line-screw condition:

$$\mathcal{L}(\mathcal{S}) = \left\{ \hat{S} = \boldsymbol{S}\boldsymbol{a} = \begin{bmatrix} \boldsymbol{U}_a \\ \boldsymbol{V}_a \end{bmatrix} \;\middle|\; \boldsymbol{U}_a^{\mathsf{T}}\boldsymbol{V}_a = 0,\ \boldsymbol{U}_a \neq \boldsymbol{0},\ \boldsymbol{a} \in \mathbb{R}^m \right\}. \tag{3.21}$$

The rank of $\mathcal{L}(\mathcal{S})$ is the maximum number of linearly independent line screws that can be selected from this subset. The system $\mathcal{S}$ is called a line screw system when this rank is $m$. In design language, this means that the whole $m$-dimensional screw system can be generated by $m$ independent directed lines.

For computation, the same idea can be checked through the self-reciprocal product of the basis matrix,

$$\boldsymbol{Q}_S = \boldsymbol{S}^{\mathsf{T}}\boldsymbol{\Delta}\boldsymbol{S} = \boldsymbol{U}^{\mathsf{T}}\boldsymbol{V} + \boldsymbol{V}^{\mathsf{T}}\boldsymbol{U}, \qquad \boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}. \tag{3.22}$$

Necessary and sufficient eigenvalue criteria based on $\boldsymbol{Q}_S$ decide whether a candidate $m$-system is a line screw system [9]. The important design message is that not every reciprocal wrench space is realizable by wires alone. It may be a mathematically valid constraint space but fail the line-screw-system test, in which case the designer must introduce other flexure primitives or use a hybrid architecture.

**Example 3.2** (Wire flexure as a line-force constraint)**.** *Consider a long, slender wire flexure connected between ground and a stage. In the idealized kinematic model, the wire is much stiffer in axial extension than in the motions associated with end rotations and transverse bending. It therefore supplies one dominant translational constraint along its axis. If the unit direction of the wire is* $\boldsymbol{u}$ *and a point on its line is* $\boldsymbol{r}$*, the unit constraint wrench is*

$$\hat{W}_w = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{r} \times \boldsymbol{u} \end{bmatrix}. \tag{3.23}$$

*This is a line screw because its moment part is the moment of a force about the origin. If the stage twist is* $\hat{T} = [\boldsymbol{\theta}^{\mathsf{T}}, \boldsymbol{\delta}^{\mathsf{T}}]^{\mathsf{T}}$*, reciprocity gives*

$$\hat{T} \circ \hat{W}_w = \boldsymbol{\delta}^{\mathsf{T}} \boldsymbol{u} + \boldsymbol{\theta}^{\mathsf{T}} (\boldsymbol{r} \times \boldsymbol{u}) = 0. \tag{3.24}$$

*The first term is the translational motion along the wire, and the second term is the axial motion induced at the wire attachment point by rotation of the stage. The wire forbids the sum of those axial motions. This one equation is the algebraic form of the visual exact-constraint statement that a wire blocks extension along its own line.*

The same construction scales to mechanisms with many flexures. In a parallel flexure stage, each blade, wire, notch, or limb contributes one or more wrench columns. The stage mobility is found by stacking those columns, reducing their rank, and computing the reciprocal twist space. In a serial chain, each element contributes motion twists, and the chain mobility is found by stacking transformed twist columns.

**Example 3.3** (A vertical wire at an offset)**.** *Let a wire be parallel to* $z$ *and pass through* $\boldsymbol{r} = [a, 0, 0]^{\mathsf{T}}$*. With unit force magnitude, its wrench is*

$$\hat{W} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -a \\ 0 \end{bmatrix}. \tag{3.25}$$

*The reciprocal condition for a stage twist is*

$$\hat{T} \circ \hat{W} = \delta_z - a\theta_y = 0. \tag{3.26}$$

*The wire does not simply constrain vertical translation. Because it is offset from the origin, a rotation about* $y$ *creates axial displacement of the wire attachment point. This is why the same wire direction can contribute different moment components when moved to different locations.*

## 3.4 Coordinate Transformations

Screws are coordinate-dependent representations of coordinate-independent geometric objects. A change of frame must therefore transform both twists and wrenches. Suppose the coordinate transformation from a local frame to a stage frame is represented by a rotation $\boldsymbol{R}$ and translation vector $\boldsymbol{d}$. Let

$$[\boldsymbol{d}]_\times = \begin{bmatrix} 0 & -d_z & d_y \\ d_z & 0 & -d_x \\ -d_y & d_x & 0 \end{bmatrix}. \tag{3.27}$$

In the convention used throughout this book and in the symbolic compliance formulation, the adjoint matrix is

$$\mathrm{Ad}(\boldsymbol{R}, \boldsymbol{d}) = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ [\boldsymbol{d}]_\times \boldsymbol{R} & \boldsymbol{R} \end{bmatrix}. \tag{3.28}$$

The transformed coordinates of a twist are

$$\hat{T}' = \mathrm{Ad}(\boldsymbol{R}, \boldsymbol{d}) \hat{T}. \tag{3.29}$$

The symbolic-compliance formulation uses the same adjoint notation for wrench coordinates in its matrix transformations [11]. Some robotics texts use the dual adjoint for wrenches to enforce a particular power-invariance convention. The formulas in this book use one convention throughout so that the mobility, compliance, and computational examples remain internally consistent.

**Example 3.4** (Adjoint transformation in the notebooks)**.** *The Mathematica notebooks define a skew-symmetric matrix from* $\boldsymbol{d}$*, then build*

$$\mathrm{Ad} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{0} \\ [\boldsymbol{d}]_\times \boldsymbol{R} & \boldsymbol{R} \end{bmatrix}. \tag{3.30}$$

*They repeatedly use this matrix to transform local flexure primitives into the stage frame before concatenating screw bases or assembling compliance and stiffness matrices. This pattern is the computational backbone of the examples in Chapters 4 and 6.*

## 3.5 Computing Reciprocal Spaces

For a column-basis twist matrix $\boldsymbol{T}$, the reciprocal wrench basis is obtained by solving

$$\boldsymbol{T}^{\mathsf{T}} \boldsymbol{\Delta} \boldsymbol{W} = \boldsymbol{0}. \tag{3.31}$$

Equivalently, the columns of $\boldsymbol{W}$ form a basis for

$$\mathrm{null}(\boldsymbol{T}^{\mathsf{T}} \boldsymbol{\Delta}). \tag{3.32}$$

For a column-basis wrench matrix $\boldsymbol{W}$, the reciprocal twist basis is obtained by solving

$$\boldsymbol{W}^{\mathsf{T}} \boldsymbol{\Delta} \boldsymbol{T} = \boldsymbol{0}. \tag{3.33}$$

In a symbolic system such as Mathematica, this is a null-space computation followed by column reduction. The implementation detail is less important than the invariant design idea: reciprocal spaces are found by linear algebra, not by visual guesswork.

**Example 3.5** (Four-wire calculation)**.** *Consider four ideal wires whose line-force wrenches in the stage frame are*

$$\hat{W}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{W}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{W}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -a \\ 0 \end{bmatrix}, \quad \hat{W}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ b \\ 0 \\ 0 \end{bmatrix}. \tag{3.34}$$

*For nonzero* $a$ *and* $b$*, the constraint matrix* $\boldsymbol{W} = [\hat{W}_1\ \hat{W}_2\ \hat{W}_3\ \hat{W}_4]$ *has rank four. The compatible freedom space is the two-dimensional null space of* $\boldsymbol{W}^{\mathsf{T}} \boldsymbol{\Delta}$*. Writing* $\hat{T} = [\theta_x, \theta_y, \theta_z, \delta_x, \delta_y, \delta_z]^{\mathsf{T}}$*, the four reciprocal equations are*

$$\delta_x = 0, \qquad \delta_y = 0, \qquad \delta_z - a\theta_y = 0, \qquad \delta_z + b\theta_x = 0. \tag{3.35}$$

*The remaining two parameters may be taken as* $\theta_y$ *and* $\theta_z$*, with* $\theta_x = -(a/b)\theta_y$ *and* $\delta_z = a\theta_y$*. A drawing of this mechanism may be hard to interpret at first glance, but the mobility follows directly from the rank and null-space calculation. Figure 3.4 shows the corresponding parallel-connected flexure picture.*

The reciprocal-space script in Appendix A constructs line-force wrenches, computes reciprocal spaces with `NullSpace`, and reduces the resulting bases with `RowReduce`. It verifies the rank-four, two-dimensional reciprocal space of the four-wire example and recovers $\mathrm{span}\{\hat{P}_x\}$ for the five-wire pure-translation design. The accompanying notebook provides the interactive workflow.

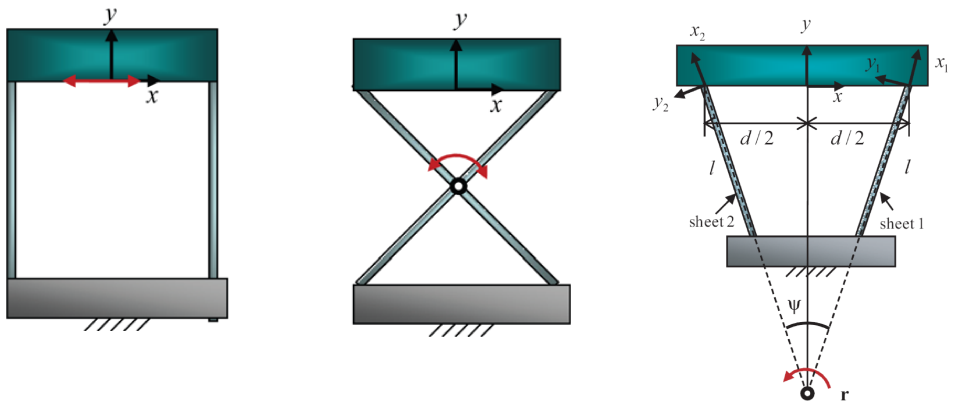



Figure 3.4: Parallel-connected flexures. The four-wire calculation is the algebraic version of this picture: each branch contributes a line constraint to the same moving body, and the stage mobility is the reciprocal space of the combined constraints.

## 3.6 Example Screw Systems from Flexure Geometry

Two compact examples are worth studying before moving from algebra to mechanism design [8]. The first example shows how a freedom space is more than a count of degrees of freedom. The second shows how an ideal sheet flexure can be represented either by a constraint space or by its reciprocal freedom space.

**Example 3.6** (Two-dimensional freedom spaces generated by rotations)**.** *Consider two pure rotational twists. If the axes are parallel, with common angular vector* $\boldsymbol{\Omega}$ *and points* $\boldsymbol{c}_1$ *and* $\boldsymbol{c}_2$ *on the two axes, the two basis twists are*

$$\hat{T}_1 = \begin{bmatrix} \boldsymbol{\Omega} \\ \boldsymbol{c}_1 \times \boldsymbol{\Omega} \end{bmatrix}, \qquad \hat{T}_2 = \begin{bmatrix} \boldsymbol{\Omega} \\ \boldsymbol{c}_2 \times \boldsymbol{\Omega} \end{bmatrix}. \tag{3.36}$$

*Any motion in the two-dimensional freedom space is*

$$\hat{T} = k_1\hat{T}_1 + k_2\hat{T}_2 = \begin{bmatrix} (k_1+k_2)\boldsymbol{\Omega} \\ (k_1\boldsymbol{c}_1 + k_2\boldsymbol{c}_2) \times \boldsymbol{\Omega} \end{bmatrix}. \tag{3.37}$$

*When* $k_1 + k_2 = 0$*, the angular part vanishes and the motion becomes a pure translation normal to the plane of the two parallel axes. Thus two parallel revolute freedoms in series can generate an instantaneous translation when their angular speeds are equal and opposite.*

*If the two rotational axes intersect at* $\boldsymbol{c}$*, the twists have the form*

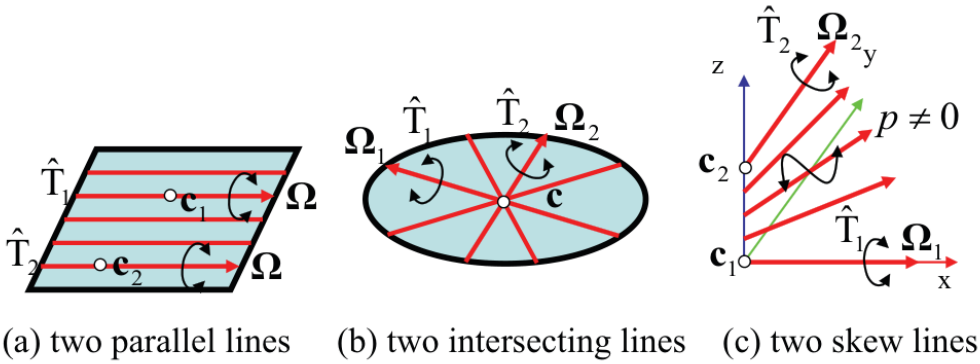
$$\hat{T}_i = \begin{bmatrix} \boldsymbol{\Omega}_i \\ \boldsymbol{c} \times \boldsymbol{\Omega}_i \end{bmatrix}, \qquad i = 1, 2. \tag{3.38}$$

*Every linear combination remains a pure rotation through the same point,*

$$\hat{T} = \begin{bmatrix} k_1\boldsymbol{\Omega}_1 + k_2\boldsymbol{\Omega}_2 \\ \boldsymbol{c} \times (k_1\boldsymbol{\Omega}_1 + k_2\boldsymbol{\Omega}_2) \end{bmatrix}. \tag{3.39}$$

*If the two axes are skew, the generic linear combination is instead a finite-pitch screw motion,*

$$\hat{T} = \begin{bmatrix} k_1\boldsymbol{\Omega}_1 + k_2\boldsymbol{\Omega}_2 \\ \boldsymbol{c}_1 \times k_1\boldsymbol{\Omega}_1 + \boldsymbol{c}_2 \times k_2\boldsymbol{\Omega}_2 \end{bmatrix}. \tag{3.40}$$




(a) two parallel lines (b) two intersecting lines (c) two skew lines

Figure 3.5: Two-dimensional freedom spaces generated by two rotational axes [8]. Parallel axes generate a plane of parallel rotations and can produce a pure translation by cancellation of angular parts. Intersecting axes generate a disk of rotations through one point. Skew axes generate a cylindroid containing finite-pitch screw motions.

*Figure 3.5 summarizes these three cases. A plane, disk, and cylindroid are all two-dimensional freedom spaces, but they have different geometric and design meanings.*

**Example 3.7** (Constraint and freedom space of an ideal sheet flexure)**.** *An ideal sheet flexure can be modeled as three independent ideal line-force constraints in the sheet plane. In the coordinate choice shown in Fig. 3.6, one convenient basis is*

$$\hat{W}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \hat{W}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}, \qquad \hat{W}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.41}$$

*These three wrenches represent two $y$-directed constraints through points on the $x$-axis and one $x$-directed constraint. A general line-force constraint in the sheet plane has force $\boldsymbol{F} = (F_x, F_y, 0)^{\mathsf{T}}$ and passes through $\boldsymbol{r} = (r_x, r_y, 0)^{\mathsf{T}}$, so*

$$\hat{W} = \begin{bmatrix} F_x \\ F_y \\ 0 \\ 0 \\ 0 \\ r_x F_y - r_y F_x \end{bmatrix}. \tag{3.42}$$

*It lies in the span of the three basis wrenches because*

$$\hat{W} = \frac{r_x F_y - r_y F_x + F_y}{2}\,\hat{W}_1 + \frac{F_y - r_x F_y + r_y F_x}{2}\,\hat{W}_2 + F_x\,\hat{W}_3. \tag{3.43}$$

*The reciprocal motion space is obtained by solving $\boldsymbol{W}_s^{\mathsf{T}} \boldsymbol{\Delta} \hat{T} = \boldsymbol{0}$, where $\boldsymbol{W}_s = [\hat{W}_1\ \hat{W}_2\ \hat{W}_3]$. Writing $\hat{T} = [\theta_x, \theta_y, \theta_z, \delta_x, \delta_y, \delta_z]^{\mathsf{T}}$, the reciprocal equations give*

$$\delta_y + \theta_z = 0, \qquad \delta_y - \theta_z = 0, \qquad \delta_x = 0. \tag{3.44}$$

*Therefore* $\delta_x = \delta_y = \theta_z = 0$*, and a convenient freedom basis is*

$$\hat{T}_1 = \hat{R}_x, \qquad \hat{T}_2 = \hat{R}_y, \qquad \hat{T}_3 = \hat{P}_z. \tag{3.45}$$

*The ideal sheet flexure therefore permits rotations about* $x$ *and* $y$ *and translation normal to the sheet, while constraining the two in-plane translations and the rotation about the sheet normal.*

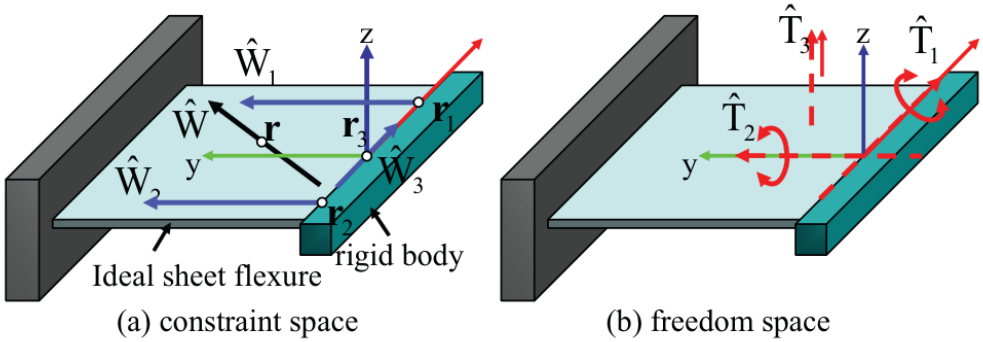



Figure 3.6: Constraint and freedom spaces of an ideal sheet flexure [8]. The left view shows a three-wrench constraint basis in the sheet plane; the right view shows the reciprocal freedom basis.

## 3.7 From Algebra to Design Judgment

The algebra in this chapter is exact only for the idealized model. Real flexures have finite stiffness in every direction, stress limits, manufacturing constraints, and nonlinearities at large motion. That does not weaken the value of screw theory. It clarifies its role. Screw theory is the conceptual design filter. It helps the designer choose a topology whose dominant compliant and stiff directions are correct before investing effort in sizing, stress analysis, finite element modeling, or optimization. The later stiffness chapter returns to finite compliance and shows how the same screw coordinates become the rows and columns of compliance and stiffness matrices.

# Chapter 4

# Mobility Analysis of Compliant Mechanisms

Mobility analysis asks what motions a given compliant mechanism allows. In screw-theory language, the answer is a freedom space. A mechanism drawing by itself does not give this space. The designer must decide which flexure primitives are being idealized, express each primitive as a twist or wrench basis, transform those bases into a common frame, and then combine them according to topology. The procedure is organized into a library of elements, a top-down decomposition of the mechanism, and a bottom-up assembly of motion and constraint spaces [10]. This chapter presents that method in a form suitable for hand calculation and computational notebooks.

## 4.1 Constraint Pattern Analysis

The rule of complementary patterns is the analysis bridge between a drawn constraint layout and its mobility. In a spatial ideal model, $c$ independent constraints leave $f = 6 - c$ degrees of freedom when every allowed twist is reciprocal to every applied wrench,

$$\hat{T}_j \circ \hat{W}_i = 0, \qquad i = 1, \ldots, c, \quad j = 1, \ldots, f, \quad c + f = 6. \tag{4.1}$$

The useful part of this rule is not merely the dimension count. The reciprocal calculation tells us what the remaining freedoms are. An ideal sheet flexure demonstrates the standard calculation: write the constraint wrenches, impose reciprocity with a general twist, solve the resulting linear equations, and choose an interpretable basis for the solution space [8].

**Example 4.1** (A sheet-flexure constraint pattern)**.** *For the sheet-flexure basis in Eq. (3.41), a general twist* $\hat{T} = [\theta_x, \theta_y, \theta_z, \delta_x, \delta_y, \delta_z]^\mathsf{T}$ *must satisfy*

$$\hat{T} \circ \hat{W}_1 = 0, \qquad \hat{T} \circ \hat{W}_2 = 0, \qquad \hat{T} \circ \hat{W}_3 = 0. \tag{4.2}$$

*The three equations are*

$$\delta_y + \theta_z = 0, \qquad \delta_y - \theta_z = 0, \qquad \delta_x = 0. \tag{4.3}$$

*Hence* $\delta_x = \delta_y = \theta_z = 0$*. A convenient basis of the reciprocal freedom space is*

$$\hat{R}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \hat{R}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \hat{P}_z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.4}$$

*This result is the ideal sheet-flexure interpretation: the element suppresses in-plane translations and rotation about the sheet normal, while allowing two bending rotations and one normal translation.*

It is tempting to think that any constraint pattern made from ideal line forces will leave only pure rotations and pure translations. The counterexample in Fig. 4.1 shows why that intuition is incomplete. Four ideal constraints are applied to a body. Two are $y$-directed constraints through $x = 1$ and $x = -1$, one is a $z$-directed constraint, and the fourth is a skew line in the plane $y = -1$ at 45° to the $x$ and $z$ axes. The corresponding wrench basis is

$$\hat{W}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \hat{W}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}, \qquad \hat{W}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{W}_4 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \end{bmatrix}. \tag{4.5}$$

Solving $\boldsymbol{W}^{\mathsf{T}} \boldsymbol{\Delta} \hat{T} = \mathbf{0}$ gives the two independent twists

$$\hat{T}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \hat{T}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{4.6}$$

The first is a pure rotation about $y$, with pitch zero. The second is a screw motion about the $x$-axis with pitch one. A general allowed motion is

$$\hat{T} = k_1 \hat{T}_1 + k_2 \hat{T}_2 = \begin{bmatrix} k_2 \\ k_1 \\ 0 \\ k_2 \\ 0 \\ 0 \end{bmatrix}. \tag{4.7}$$

If this general motion were a pure rotation, it would satisfy $\boldsymbol{\Omega}^{\mathsf{T}} \boldsymbol{V} = 0$, which here gives $k_2^2 = 0$. Thus $k_2 = 0$, and the only pure rotation in the space is the $\hat{T}_1$ direction. All other freedoms in this two-dimensional space are finite-pitch screw motions. The example is important because it separates mobility count from mobility type: $f = 2$, but the second freedom is not an ordinary revolute or translational motion.

## 4.2 Element Libraries

The first step in mobility analysis is to define the local model of each primitive. A wire flexure is most naturally modeled by a constraint wrench along its axis. A blade or sheet flexure can be

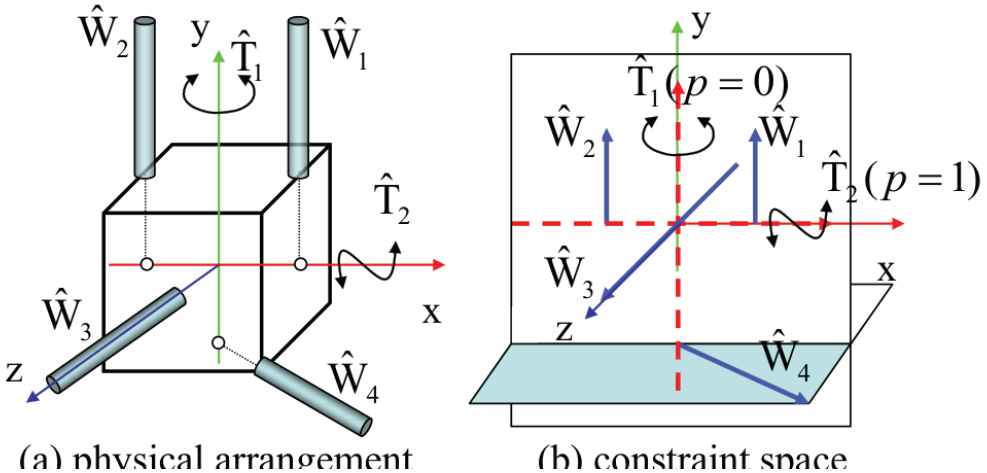



Figure 4.1: Constraint pattern whose reciprocal freedom space contains a finite-pitch screw motion [8]. Four ideal line constraints leave a two-dimensional freedom space spanned by one pure rotation and one screw motion.

modeled by a motion basis describing the relatively compliant deformations or by a constraint basis describing the relatively stiff directions. A spherical notch hinge, rotational hinge, or compound flexure chain can be modeled as a higher-level primitive after its own motion and constraint spaces have been derived. The library is not merely a table of names. It is a table of local screw bases with stated assumptions.

For example, a wire flexure is naturally modeled as one line-force constraint. An ideal sheet flexure may instead be modeled as a primitive with one dominant rotational freedom and two dominant translational freedoms, with the complementary constraints supplied by the stiff directions of the sheet. The exact basis chosen depends on whether the sheet is being used as an ideal blade, a leaf spring, a cross-strip pivot element, or a member of a larger chain. The analysis must state the chosen model before rank calculations can be meaningful.

**Definition 4.1** (Local primitive model)**.** *A local primitive model is a pair* $(\boldsymbol{T}_e, \boldsymbol{W}_e)$ *written in the primitive's local coordinate frame, where* $\boldsymbol{T}_e$ *spans the ideal motion space and* $\boldsymbol{W}_e$ *spans the ideal constraint space. The bases should satisfy* $\boldsymbol{T}_e^{\mathsf{T}} \boldsymbol{\Delta} \boldsymbol{W}_e = \boldsymbol{0}$ *in the ideal model, with* $\operatorname{rank}(\boldsymbol{T}_e) + \operatorname{rank}(\boldsymbol{W}_e) = 6$ *when the primitive model is exactly constrained.*

This definition is deliberately broad. Some flexure primitives are best specified by $\boldsymbol{W}_e$, after which $\boldsymbol{T}_e$ is computed as the reciprocal space. Others are best specified by $\boldsymbol{T}_e$, after which $\boldsymbol{W}_e$ is computed. The designer should choose the representation that is most natural for the primitive and topology. Figure 4.2 gives representative primitive models in this local-basis language.

## 4.3 Serial Composition

A serial flexure chain connects the stage to ground through intermediate bodies. Since the elements are connected in sequence, the stage motion is the superposition of the motions contributed by each element, after each local motion basis has been transformed into the stage frame. If the $j$-th element has local motion matrix $\boldsymbol{T}_j$ and adjoint transformation $\mathrm{Ad}_j$, the serial-chain motion matrix is

$$\boldsymbol{T}_{\mathrm{ser}} = \begin{bmatrix} \mathrm{Ad}_1 \boldsymbol{T}_1 & \mathrm{Ad}_2 \boldsymbol{T}_2 & \cdots & \mathrm{Ad}_m \boldsymbol{T}_m \end{bmatrix}. \tag{4.8}$$

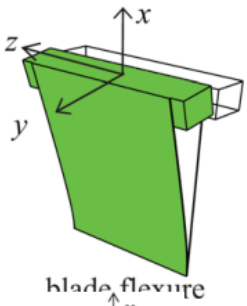
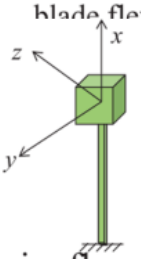
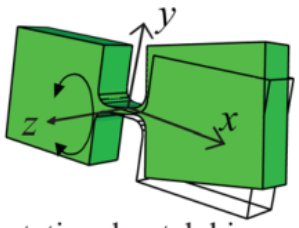
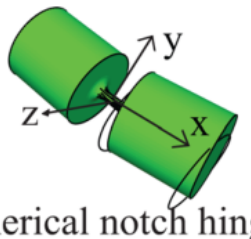
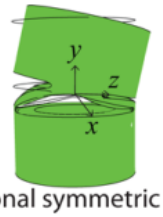
| Flexure | $[T]$ (motion) | $[W]$ (constraint) |
|---|---|---|
|  blade flexure | $[T_b] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ | $[W_b] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ |
|  wire flexure | $[T_w] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ | $[W_w] = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ |
|  rotational notch hinge | $[T_r] = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ | $[W_r] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ |
|  spherical notch hinge | $[T_s] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ | $[W_s] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |
|  rotational symmetric cylinder | $[T_c] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ | $[W_c] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ |

Figure 4.2: Representative flexure primitives and their screw-system models. Every primitive must enter the analysis as a stated local twist or wrench basis, not only as a drawing [10].

The serial-chain mobility is

$$f_{\text{ser}} = \operatorname{rank}(\boldsymbol{T}_{\text{ser}}). \tag{4.9}$$

A reduced basis for the freedom space is obtained by column reduction, and the reciprocal constraint basis is obtained from $\operatorname{null}(\boldsymbol{T}_{\text{ser}}^{\mathsf{T}}\boldsymbol{\Delta})$.

### 4.3.1 Serial Chain Example: Two Perpendicular Blade Flexures

A serial chain of two identical blade flexures is a useful teaching example [10]. The first blade is transformed by a translation, while the second is transformed by a rotation about the $z$-axis. The motion basis of the chain is not obtained by adding scalar degrees of freedom. It is obtained by concatenating the transformed twist bases. Some columns may become dependent after transformation, so the final mobility is the rank of the concatenated matrix, not the sum of the nominal mobilities of the two blades.

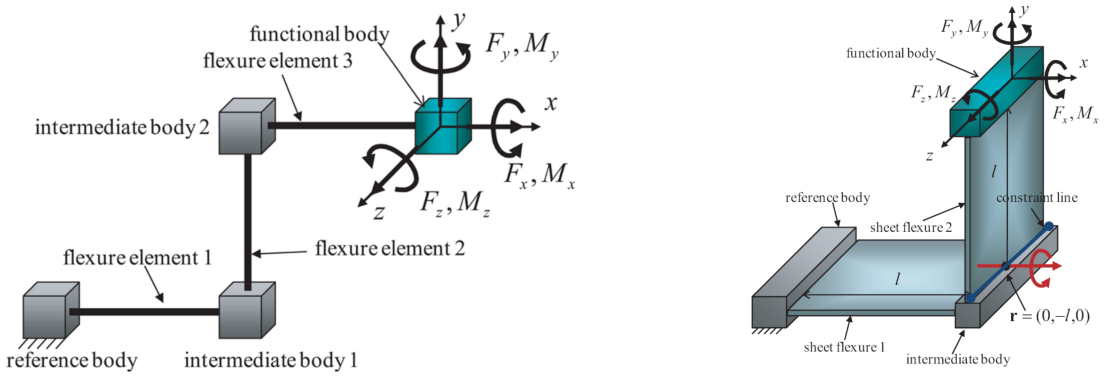
For this example the calculation is

$$\boldsymbol{T}_{bb} = \begin{bmatrix} \operatorname{Ad}_1 \boldsymbol{T}_b & \operatorname{Ad}_2 \boldsymbol{T}_b \end{bmatrix}, \qquad f_{bb} = \operatorname{rank}(\boldsymbol{T}_{bb}), \tag{4.10}$$

where $\boldsymbol{T}_b$ is the local blade-flexure motion basis. The column-reduced matrix $\bar{\boldsymbol{T}}_{bb}$ is the actual serial-chain freedom basis, and

$$\boldsymbol{W}_{bb} = \operatorname{null}(\boldsymbol{T}_{bb}^{\mathsf{T}}\boldsymbol{\Delta}) \tag{4.11}$$

is the remaining chain constraint space. This is the computation performed in the mobility notebooks: define the two adjoint transformations, multiply them into the local blade basis, concatenate the two transformed bases, and then apply `RowReduce`, `MatrixRank`, and `NullSpace`. Figure 4.3(a) shows the generic serial-chain geometry, while Fig. 4.3(b) shows the specific two-blade realization used below.




(a) Generic serial chain.

(b) Two perpendicular blade flexures.

Figure 4.3: Serial-chain flexure examples. (a) A generic serial chain of flexures, analyzed by transforming each local flexure basis into a common frame and concatenating the resulting twist columns. (b) A concrete two-perpendicular-blade-flexure example connected in series through an intermediate body.

Figure 4.3(b) shows a concrete version of this calculation: two perpendicular blade flexures connected through an intermediate body. The first blade is offset from the functional body by the blade length $l$, while the second blade is rotated by $-\pi/2$ about the $z$-axis. In the book's

twist-coordinate order $[\theta_x, \theta_y, \theta_z, \delta_x, \delta_y, \delta_z]^\mathsf{T}$, an ideal blade-flexure motion basis may be written as

$$\boldsymbol{T}_b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.12}$$

For the two perpendicular blades in Fig. 4.3(b), the twist transformations are

$$\mathrm{Ad}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & l & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ -l & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathrm{Ad}_2 = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.13}$$

Column-wise assembly gives

$$\boldsymbol{T}_{bb} = \begin{bmatrix} \mathrm{Ad}_1 \boldsymbol{T}_b & \mathrm{Ad}_2 \boldsymbol{T}_b \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ l & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -l & 0 & 0 & 0 \end{bmatrix}. \tag{4.14}$$

Reducing the columns and deleting the zero column gives a rank-five freedom basis,

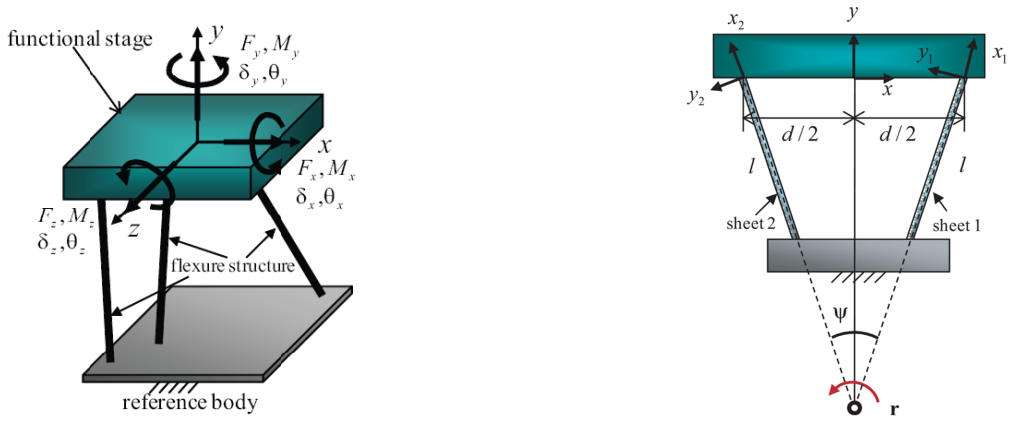
$$\bar{\boldsymbol{T}}_{bb} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -l & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad f_{bb} = 5. \tag{4.15}$$

The first basis twist is not a pure rotation about the $x$-axis at the chosen reference point; it carries the translational component $-l$ in the $\delta_z$ row. This is the geometric effect of carrying the first blade's local freedom through the intermediate body to the functional body.

This rank statement is one of the most important habits in compliant mechanism analysis. A designer may assemble two flexure elements and expect their freedoms to add. They add only as vector spaces, and vector spaces can overlap. Redundancy is not a bookkeeping nuisance; it is often the difference between a mechanism that actually has the intended motion and one that only appears to have it.

## 4.4 Parallel Composition

A parallel flexure mechanism connects the stage to ground through two or more branches. Since all branches act on the same stage, their constraints are superimposed. Figure 4.4(a) shows the physical topology: each flexure branch connects the same functional stage to the reference body, so




(a) Generic parallel chain.

(b) Parallel blade-flexure pivot.

Figure 4.4: Parallel-chain flexure examples. (a) Multiple flexure structures connect the same functional stage to the reference body, so their branch constraints are assembled in the common stage frame. (b) A two-blade parallel flexure pivot used for the wrench-matrix derivation.

all branch wrenches constrain one common stage twist. Figure 4.4(b) previews the two-blade pivot used in the worked calculation of Section 4.4.1.

If the $j$-th branch has local constraint matrix $\boldsymbol{W}_j$ and adjoint transformation $\mathrm{Ad}_j$, the parallel constraint matrix is

$$\boldsymbol{W}_{\mathrm{par}} = \begin{bmatrix} \mathrm{Ad}_1 \boldsymbol{W}_1 & \mathrm{Ad}_2 \boldsymbol{W}_2 & \cdots & \mathrm{Ad}_m \boldsymbol{W}_m \end{bmatrix}. \tag{4.16}$$

The degree of constraint is

$$c_{\mathrm{par}} = \mathrm{rank}(\boldsymbol{W}_{\mathrm{par}}). \tag{4.17}$$

The compatible freedom basis is obtained from

$$\mathcal{F}_{\mathrm{par}} = \mathrm{null}(\boldsymbol{W}_{\mathrm{par}}^{\mathsf{T}} \boldsymbol{\Delta}). \tag{4.18}$$

When the ideal parallel mechanism is exactly constrained, the mobility is $6 - c_{\mathrm{par}}$.

This rule is best used as a hierarchical calculation rather than as a drawing-level classification [10]. The top-down part is the modeling step: divide the mechanism into modules, then divide each module until the leaves are primitives or simple chains whose local twist or wrench matrices are known. The bottom-up part is the algebraic assembly step: analyze the leaves first, replace each solved submodule by its reduced motion or constraint matrix, and pass that matrix upward to the parent module.

For a serial module $\mathcal{M}$, the quantity passed upward is a motion space. If the children have local twist matrices $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_m$, and if $\mathrm{Ad}_j$ maps the $j$-th child frame into the module frame, then

$$\boldsymbol{T}_{\mathcal{M}} = \begin{bmatrix} \mathrm{Ad}_1 \boldsymbol{T}_1 & \mathrm{Ad}_2 \boldsymbol{T}_2 & \cdots & \mathrm{Ad}_m \boldsymbol{T}_m \end{bmatrix}, \qquad f_{\mathcal{M}} = \mathrm{rank}(\boldsymbol{T}_{\mathcal{M}}). \tag{4.19}$$

After column reduction, $\boldsymbol{T}_{\mathcal{M}}$ is the module freedom basis. Any complementary constraint basis can be recovered from

$$\boldsymbol{W}_{\mathcal{M}} = \mathrm{null}(\boldsymbol{T}_{\mathcal{M}}^{\mathsf{T}} \boldsymbol{\Delta}). \tag{4.20}$$

Thus a serial chain contributes transformed twist columns to its parent; its mobility is the rank of those columns, not the sum of the nominal degrees of freedom of the individual flexures.

For a parallel module $\mathcal{M}$, the quantity passed upward is first a constraint space. If the children have local wrench matrices $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_m$, the assembled module constraint matrix is

$$\boldsymbol{W}_{\mathcal{M}} = \begin{bmatrix} \mathrm{Ad}_1 \boldsymbol{W}_1 & \mathrm{Ad}_2 \boldsymbol{W}_2 & \cdots & \mathrm{Ad}_m \boldsymbol{W}_m \end{bmatrix}, \qquad c_{\mathcal{M}} = \mathrm{rank}(\boldsymbol{W}_{\mathcal{M}}). \tag{4.21}$$

The module freedom basis is then the reciprocal twist space

$$\mathcal{F}_{\mathcal{M}} = \mathrm{null}(\boldsymbol{W}_{\mathcal{M}}^{\mathsf{T}} \boldsymbol{\Delta}), \qquad f_{\mathcal{M}} = 6 - c_{\mathcal{M}} \tag{4.22}$$

for an exactly constrained ideal spatial stage. A parallel branch therefore contributes transformed wrench columns; only after rank reduction and reciprocity does the mobility space appear.

### 4.4.1 Parallel Chain Example: Two Blade Flexures

A trapezoidal or cross-strip flexure pivot can be idealized as two blade flexures assembled in parallel. Each blade contributes a constraint space in its own local frame. The stage constraint space is obtained by transforming both blade constraint matrices and concatenating them. The reciprocal twist basis then reveals the dominant rotation of the pivot. When the blades are symmetric, some constraint columns become redundant, but that redundancy may still be useful mechanically because it increases stiffness, load capacity, or symmetry.

The angled two-flexure calculation gives a useful concrete result. With the notation used here, the two blade frames are obtained from the stage frame by rotations about $z$ and opposite offsets [10],

$$\boldsymbol{R}_1 = Z\left(\frac{\pi - \psi}{2}\right), \qquad \boldsymbol{d}_1 = \begin{bmatrix} d/2 & 0 & 0 \end{bmatrix}^{\mathsf{T}}, \qquad \boldsymbol{R}_2 = Z\left(\frac{\pi + \psi}{2}\right), \qquad \boldsymbol{d}_2 = \begin{bmatrix} -d/2 & 0 & 0 \end{bmatrix}^{\mathsf{T}}. \tag{4.23}$$

If $\boldsymbol{W}_b$ is the local blade-flexure constraint basis and $\mathrm{Ad}_i$ is the corresponding wrench transformation, the parallel assembly is

$$\boldsymbol{W}_t = \begin{bmatrix} \mathrm{Ad}_1 \boldsymbol{W}_b & \mathrm{Ad}_2 \boldsymbol{W}_b \end{bmatrix}. \tag{4.24}$$

For the two symmetric blades, the assembled wrench matrix written in the common stage frame is

$$\boldsymbol{W}_t = \begin{bmatrix} 0 & \sin(\psi/2) & 0 & 0 & -\sin(\psi/2) & 0 \\ 0 & \cos(\psi/2) & 0 & 0 & \cos(\psi/2) & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -\cos(\psi/2) & 0 & 0 & -\cos(\psi/2) \\ -d/2 & 0 & \sin(\psi/2) & d/2 & 0 & -\sin(\psi/2) \\ 0 & \frac{d}{2}\cos(\psi/2) & 0 & 0 & -\frac{d}{2}\cos(\psi/2) & 0 \end{bmatrix}. \tag{4.25}$$

Column reduction of this matrix gives the compact basis used by the Mathematica script,

$$\bar{\boldsymbol{W}}_{\angle} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{d}{2}\cot(\psi/2) & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.26}$$

For generic nonzero $d$ and nondegenerate $\psi$, this matrix has rank five. The reciprocal twist basis printed by the Wolfram script is

$$\hat{T}_{\angle} = \begin{bmatrix} 0 & 0 & 1 & -\frac{d}{2}\cot(\psi/2) & 0 & 0 \end{bmatrix}^{\mathsf{T}}. \tag{4.27}$$

The result has the form of a rotation about $z$ coupled to an $x$-translation of the reference point. Geometrically, the pivot is not merely "one DOF"; its one DOF is a particular screw whose center is set by the flexure spacing and angle. Figure 4.4(b) shows the parallel flexure pivot used for this wrench-matrix derivation.

## 4.5 Hybrid Mechanisms

Most useful flexure mechanisms are neither purely serial nor purely parallel. They are hybrid assemblies of modules. The practical mobility workflow is therefore hierarchical. First divide the mechanism into modules whose topology is easy to recognize. Then analyze each low-level module using the serial or parallel rule. Replace each analyzed module by its reduced motion or constraint basis. Continue upward until the stage-level freedom space is obtained.

An $xy$ flexure stage illustrates the idea [10]. A stage may be divided into symmetric modules. Each module may contain parallelograms. Each parallelogram may be formed by two serial chains assembled in parallel. Each serial chain may be formed by rotational hinges. The analysis begins at the hinge level and proceeds upward. The resulting mobility is not guessed from the final drawing; it is built from transformed spaces.

**Principle 4.1** (Top-down decomposition and bottom-up assembly)**.** *For a general compliant mechanism, first decompose the geometry into known flexure elements, joints, chains, and modules. Then assemble each module's screw space from the bottom up. Serial submodules are assembled through transformed twist spaces, parallel submodules through transformed wrench spaces, and hybrid modules by recursively applying the appropriate rule at each level.*

The principle also gives a useful debugging procedure. If the final mobility is not what was intended, do not immediately redesign the entire mechanism. Inspect the rank at each module boundary. A lost freedom usually appears where two expected independent twist columns become dependent. An unwanted freedom usually appears where a constraint matrix has lower rank than expected.

### 4.5.1 Hybrid Chain Example: Spherical-Notch Chains

The notebooks use a two-spherical-notch serial chain, written informally as an SS chain, and then assemble three such chains in parallel. A single spherical joint allows three rotations and constrains three translations at its center. When two spherical joints are connected in series by an intermediate body, the chain can be interpreted as a limb whose motion space is obtained by concatenating the transformed rotational twist bases of the two joints. The reciprocal constraint space of the chain is then computed from that serial motion space.

When three SS chains are connected in parallel between the stage and ground, each chain contributes its constraint basis to the stage. The stage constraint matrix is the concatenation of the three transformed chain wrench matrices. The stage mobility follows from the reciprocal twist space. This example is valuable pedagogically because it forces the student to use both rules. Each limb is a serial mobility problem. The platform is a parallel constraint problem. Figure 4.5 gives the corresponding three-limb platform architecture.

The Wolfram reproduction of the three SS-chain example reduces the stage constraint matrix

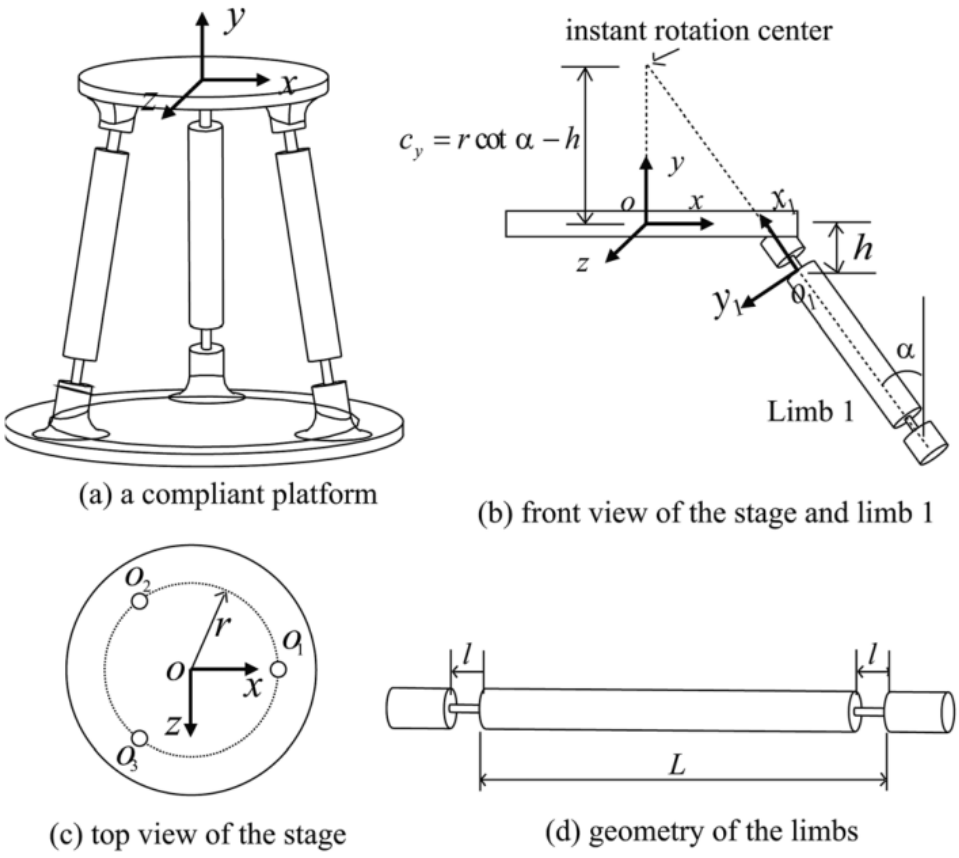



Figure 4.5: Three-limb platform from [11]. The spherical-notch-chain mobility example has the same hierarchical structure: a serial limb is first reduced to a module, and multiple limbs are then assembled in parallel at the moving platform.

to

$$\bar{\boldsymbol{W}}_{3SS} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -h + r\cot\alpha \\ 0 & 0 & 0 \\ h - r\cot\alpha & 0 & 0 \end{bmatrix}. \tag{4.28}$$

For a generic layout, the rank is three. The reciprocal motion basis is

$$\bar{\boldsymbol{T}}_{3SS} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ -h + r\cot\alpha & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & h - r\cot\alpha \end{bmatrix}. \tag{4.29}$$

Thus the platform has a three-dimensional freedom space. The parameters $h$, $r$, and $\alpha$ do not change the generic mobility count, but they do shift the translational components associated with the rotational basis twists. This is exactly the kind of information that is lost when one reports only the number of degrees of freedom.

The mobility-analysis script in Appendix A reproduces the rank-five angled-flexure result and the rank-three SS-chain result using `RowReduce` and `MatrixRank`. The accompanying notebook contains the broader mobility-analysis workspace.

## 4.6 Qualitative Mobility Versus Quantitative Mobility

The screw-algebra method gives qualitative mobility. It identifies the ideal twist directions that are allowed and the reciprocal wrench directions that are constrained. In a real compliant mechanism, every direction has finite compliance; a direction treated as constrained in the ideal model is simply much less compliant than a direction treated as free. The final design must therefore check the intended stiffness hierarchy as well as stress and range of motion. Chapter 6 develops the derivation, transformation, and assembly of stiffness and compliance matrices. This section uses a completed compliance matrix only to determine whether its effective mobility agrees with the ideal screw-space prediction.

### 4.6.1 Mobility from Compliance Decomposition

The compliance matrix also contains mobility information. The compliance-based procedure separates the deformation response into translational and rotational compliance axes, introduces a characteristic length to compare them, and then determines which directions are effectively mobile [13]. It complements the preceding screw-space analysis. The latter identifies ideal freedom and constraint spaces; the compliance decomposition asks whether the manufactured geometry preserves the required stiffness separation.

The procedure uses the twist order and wrench order introduced in Chapter 6. Define the selector

$$\boldsymbol{T}_1 = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}. \tag{4.30}$$

The translational eigenscrews are obtained from the generalized eigenvalue problem

$$\boldsymbol{C}\hat{W}_{\delta i} = c_{\delta i}\boldsymbol{T}_1\hat{W}_{\delta i}, \qquad \hat{T}_{\delta i} = \boldsymbol{T}_1\hat{W}_{\delta i}, \tag{4.31}$$

and the rotational eigenscrews from

$$\boldsymbol{C}\boldsymbol{T}_1\hat{T}_{\theta i} = c_{\theta i}\hat{T}_{\theta i}, \qquad \hat{W}_{\theta i} = \boldsymbol{T}_1\hat{T}_{\theta i}. \tag{4.32}$$

The first problem has three finite translational eigencompliances $c_{\delta i}$; the other generalized eigenvalues are infinite because $\boldsymbol{T}_1$ is singular. The second problem has three nonzero rotational eigencompliances $c_{\theta i}$. Their eigentwists locate the compliant motion axes, and the associated eigenwrenches identify the reciprocal loading directions. The eigencompliances are invariant under a change of reference frame even though the coordinate components of the eigenscrews transform with the adjoint operator [13].

Raw translational and rotational eigencompliances cannot yet be compared. Their units are respectively m/N and rad/(N m). Let $l_c$ be a characteristic length of the mechanism, and define

$$\widetilde{\hat{T}} = \underbrace{\begin{bmatrix} l_c\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{T}_2}\hat{T}, \qquad \widetilde{\hat{W}} = \underbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}/l_c \end{bmatrix}}_{\boldsymbol{T}_3}\hat{W}. \tag{4.33}$$

For a block-partitioned compliance matrix,

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix}, \tag{4.34}$$

the scaled matrix is

$$\widetilde{\boldsymbol{C}} = \boldsymbol{T}_2\boldsymbol{C}\boldsymbol{T}_3^{-1} = \begin{bmatrix} l_c\boldsymbol{C}_{11} & l_c^2\boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & l_c\boldsymbol{C}_{22} \end{bmatrix}. \tag{4.35}$$

Equivalently, the translational eigencompliances are unchanged and only the rotational eigencompliances are rescaled:

$$\widetilde{c}_{\delta i} = c_{\delta i}, \qquad \widetilde{c}_{\theta i} = l_c^2 c_{\theta i}. \tag{4.36}$$

All six values then have common units of length per force. The length unit used for $l_c$ must match the unit used in the compliance matrix.

For a closed-loop mechanism supported by two or more chains, a practical choice is the largest separation between any pair of stage–chain connection boundaries. If $B_i$ and $B_j$ are two such boundaries, define

$$d_{ij} = \max_{\boldsymbol{p}_i \in B_i,\, \boldsymbol{p}_j \in B_j} \|\boldsymbol{p}_i - \boldsymbol{p}_j\|, \qquad l_c = \max_{i \neq j} d_{ij}. \tag{4.37}$$

For an open serial chain, $l_c$ is instead the largest distance between the motion-stage boundary and the ground boundary. These topology-based choices are estimates rather than material constants. When the mobility classification changes strongly with dimensions, the characteristic length can be refined from the distance of each rotational eigentwist axis to the mechanism boundaries [13].

Let the six scaled eigencompliances be ordered as

$$\widetilde{c}_1 \geq \widetilde{c}_2 \geq \cdots \geq \widetilde{c}_6 \geq 0, \tag{4.38}$$

and define the compliance ratios

$$CR_i = \frac{\widetilde{c}_i}{\widetilde{c}_1}. \tag{4.39}$$

Let $m$ be the number of ratios satisfying $CR_i \leq \epsilon$, where values between $10^{-3}$ and $10^{-2}$ provide useful initial engineering thresholds. The threshold is not universal; it encodes the required stiffness separation for a particular precision, load level, and allowable parasitic motion.

**Principle 4.2** (Criterion 1: relative compliance separation)**.** *If $m \geq 1$, the $m$ small ratios represent effectively constrained directions, and the effective mobility is*

$$N = 6 - m. \tag{4.40}$$

*Criterion 1 covers $1 \leq N \leq 5$. The eigentwists associated with the $N$ largest scaled eigencompliances are the quantitative freedom directions.*

**Principle 4.3** (Criterion 2: no relative separation)**.** *If $m = 0$, all six directions have comparable compliance and a relative test cannot distinguish a mobility. Introduce an application-dependent absolute compliance $c_\epsilon$. Then*

$$N = \begin{cases} 0, & \widetilde{c}_1 \leq c_\epsilon, \\ 6, & \widetilde{c}_1 > c_\epsilon. \end{cases} \tag{4.41}$$

*Criterion 2 distinguishes a mechanism that is effectively rigid at the relevant load scale from one that is compliant in all six instantaneous directions.*

Figure 4.6 summarizes the decision after the scaled eigencompliances and ratios have been calculated.

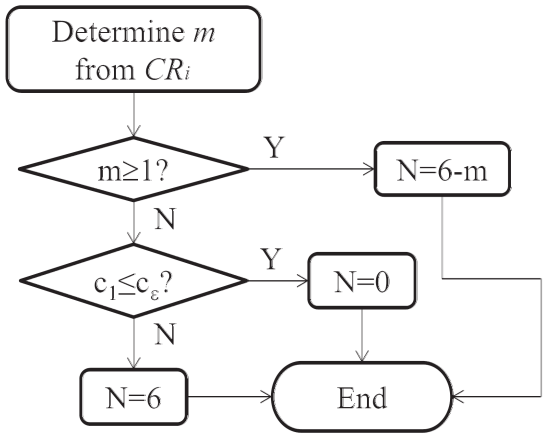



Figure 4.6: Decision sequence for the two effective-mobility criteria [13].

The calculations can be performed directly in Mathematica once a numerical $6 \times 6$ matrix is available. The compliance-mobility script in Appendix A calls `Eigensystem[{C,T1}]` for the translational eigenvalues and `Eigensystem[C.T1]` for the rotational eigenvalues. It scales the rotational set by $l_c^2$, computes the ratios in Equation (4.39), applies the two criteria, and prints the associated eigenscrew vectors.

**Example 4.2** (Blade flexure: Criterion 1)**.** *Consider the blade flexure in Figure 4.7, with length $l$, width $w$, thickness $t$, area $A = tw$, and principal second moments*

$$I_y = \frac{tw^3}{12}, \qquad I_z = \frac{wt^3}{12}. \tag{4.42}$$

*The principal-axis translational and rotational eigencompliances are*

$$\boldsymbol{c}_\delta = \begin{bmatrix} \dfrac{l}{EA} & \dfrac{l^3}{12EI_z} & \dfrac{l^3}{12EI_y} \end{bmatrix}^{\mathsf{T}}, \qquad \boldsymbol{c}_\theta = \begin{bmatrix} \dfrac{l}{Gkwt^3} & \dfrac{l}{EI_y} & \dfrac{l}{EI_z} \end{bmatrix}^{\mathsf{T}}, \tag{4.43}$$

*where $G$ is the shear modulus and $k$ is the torsional correction factor. The physical axis associated with each component follows the local coordinate definition of the blade; the formulas are most useful as a principal-axis spectrum before assembly. Figure 4.8 shows the characteristic-length robustness of this same eigencompliance calculation.*

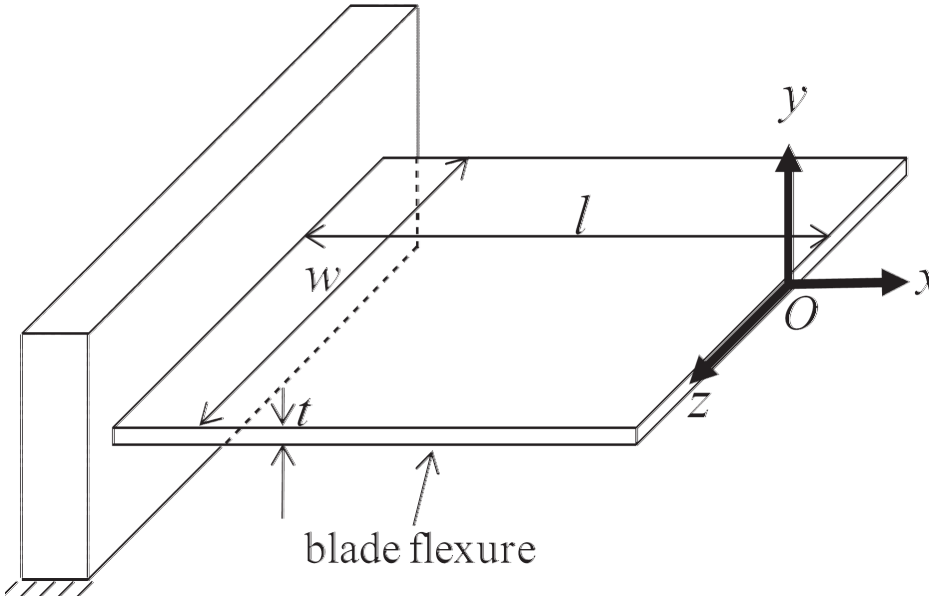



Figure 4.7: Blade flexure used to illustrate the scaled eigencompliance calculation [13].

With $t/w = 0.02$, $t/l = 0.01$, $\nu = 0.3$, $E/G = 2.6$, $k = 1/3$, $l_c = l$, and $\epsilon = 10^{-3}$, the ordered ratios reported for the blade are

$$\boldsymbol{CR} = \begin{bmatrix} 1 & 0.65 & 0.0833 & 4.0 \times 10^{-4} & 3.33 \times 10^{-5} & 8.33 \times 10^{-6} \end{bmatrix}^{\mathsf{T}}. \tag{4.44}$$

There are three ratios below the threshold, so Criterion 1 yields $m = 3$ and $N = 3$. The mechanism is therefore compliant in three directions: the two bending rotations and the transverse translation. If the width is reduced until $w = t = 0.01l$, the blade becomes wire-like. The corresponding ratios are

$$\boldsymbol{CR} = \begin{bmatrix} 1 & 0.65 & 0.0833 & 1 & 0.0833 & 8.33 \times 10^{-6} \end{bmatrix}^{\mathsf{T}}, \tag{4.45}$$

so only one direction is effectively constrained and $N = 5$. The result makes the design implication explicit: reducing the width does not merely change a stiffness value; it changes the effective mobility of the flexure.

The classification is robust to a useful interval of characteristic lengths. The plot confirms that the three-dimensional result remains unchanged over a broad range around $l_c = l$; this is expected because the spectral gap remains well separated from the threshold.

**Example 4.3** (Two wire flexures in series: Criterion 2)**.** *Figure 4.9 shows two identical wire flexures connected in series. For $l = 50\,\mathrm{mm}$, $t = 1\,\mathrm{mm}$, $w = 40\,\mathrm{mm}$, $E = 210\,\mathrm{GPa}$, $\nu = 0.28$, and $l_c = 70\sqrt{2}\,\mathrm{mm}$, serial compliance assembly followed by the eigenproblems in Equations (4.31) and (4.32) gives*

$$\widetilde{\boldsymbol{c}}_\delta = \begin{bmatrix} 6.3812 & 0.5955 & 8.1606 \end{bmatrix}^{\mathsf{T}}\ \mathrm{mm/N}, \qquad \widetilde{\boldsymbol{c}}_\theta = \begin{bmatrix} 70.4142 & 70.4142 & 56 \end{bmatrix}^{\mathsf{T}}\ \mathrm{mm/N}. \tag{4.46}$$

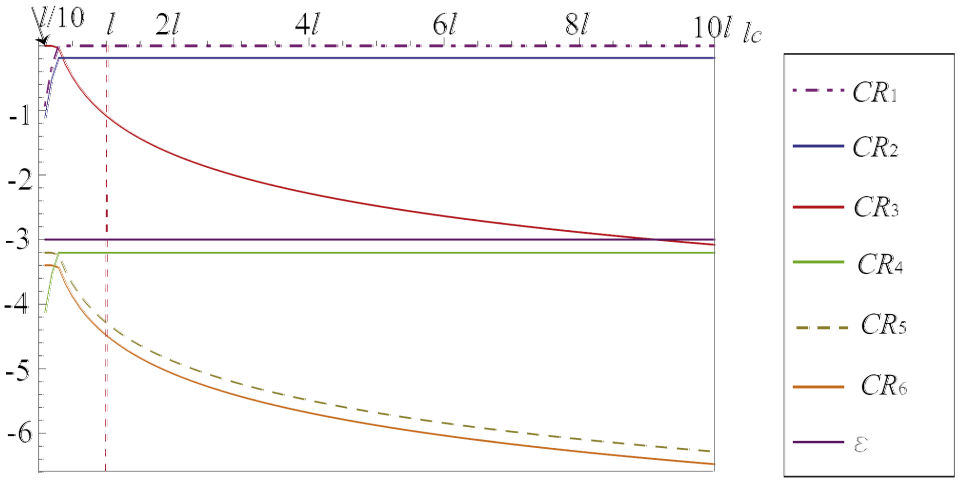



Figure 4.8: Compliance ratios of the blade example as the characteristic length is varied. The threshold identifies three effectively constrained directions over the central range of the plot [13].

*The ordered ratios are*

$$\boldsymbol{CR} = \begin{bmatrix} 1 & 1 & 0.7953 & 0.1159 & 0.09062 & 0.008457 \end{bmatrix}^{\mathsf{T}}. \tag{4.47}$$

*For* $\epsilon = 10^{-3}$*, none is below the relative threshold and* $m = 0$*. Choosing* $c_\epsilon = 0.01\,\mathrm{mm/N}$*, the largest compliance is much larger than the absolute limit, so Criterion 2 gives* $N = 6$*.*

Increasing the thickness to $t = 20\,\mathrm{mm}$, while holding the other dimensions fixed, gives

$$\widetilde{\boldsymbol{c}}_\delta = \begin{bmatrix} 4.0476 \times 10^{-5} \\ 4.3155 \times 10^{-6} \\ 5.1004 \times 10^{-5} \end{bmatrix} \mathrm{mm/N}, \tag{4.48}$$

$$\widetilde{\boldsymbol{c}}_\theta = \begin{bmatrix} 4.4009 \times 10^{-4} \\ 4.4009 \times 10^{-4} \\ 3.5 \times 10^{-4} \end{bmatrix} \mathrm{mm/N}. \tag{4.49}$$

Again $m = 0$, but now $\widetilde{c}_1 = 4.4009 \times 10^{-4}\,\mathrm{mm/N} < c_\epsilon$, so the same criterion gives $N = 0$. This pair of calculations shows why a relative spectral test alone is insufficient: it cannot distinguish a highly compliant body from an effectively rigid one when neither has a pronounced internal stiffness hierarchy.

**Example 4.4** (Three-limb spatial parallel mechanism)**.** *The spatial parallel flexure in Figure 4.10 has three identical limbs, each with two circular notch flexures. After adding the two transformed notch compliances in each limb, inverting to the limb stiffness, and adding the three transformed limb stiffnesses, the resulting stage compliance produces a three-dimensional mobility for the nominal geometry. With* $E = 210\,\mathrm{GPa}$*,* $\nu = 0.3$*,* $t = 2\,\mathrm{mm}$*,* $r = 10\,\mathrm{mm}$*,* $L = 200\,\mathrm{mm}$*,* $r_3 = 100\,\mathrm{mm}$*, and* $\beta = 30°$*, the geometry-based guide gives* $l_c = 100\sqrt{3}\,\mathrm{mm}$*. The scaled compliance ratios are*

$$\boldsymbol{CR} = \begin{bmatrix} 1 & 0.3514 & 0.3514 & 2.9881 \times 10^{-4} & 2.9881 \times 10^{-4} & 4.9812 \times 10^{-5} \end{bmatrix}^{\mathsf{T}}. \tag{4.50}$$

*Thus* $m = 3$ *and* $N = 3$*, with the three dominant eigentwists corresponding to rotations about the stage axes.*

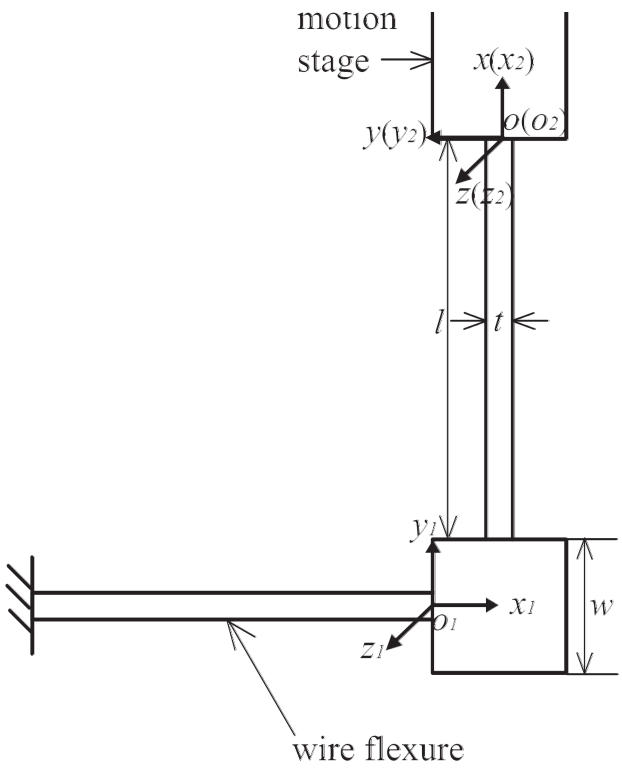
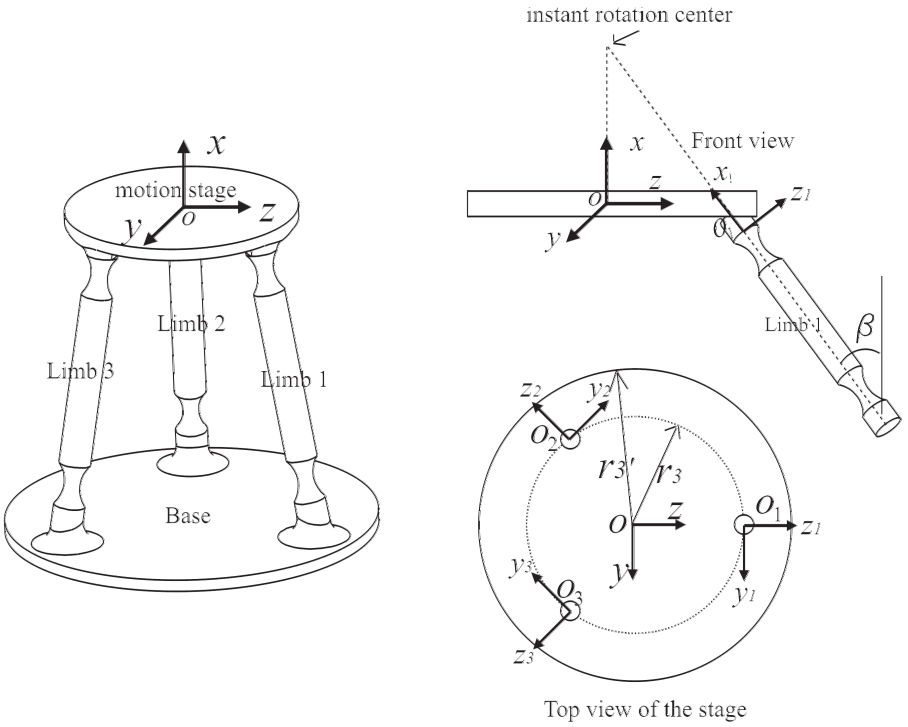
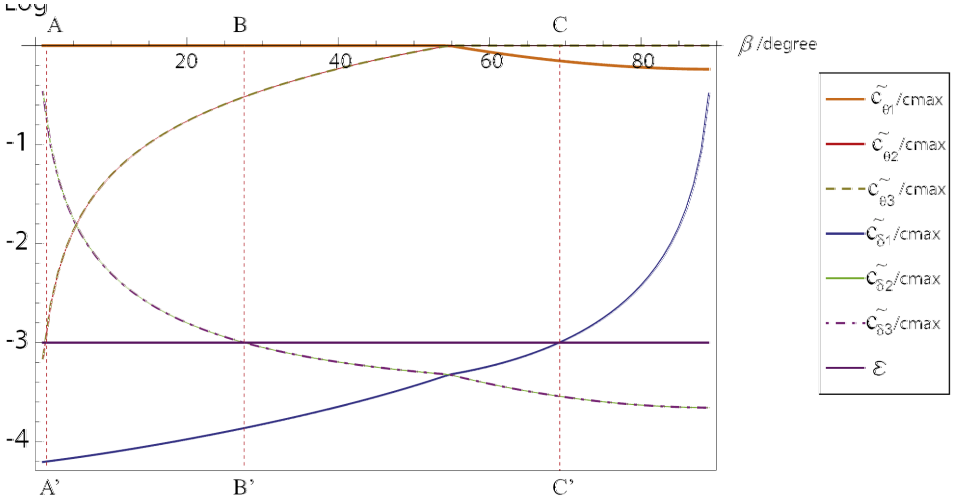



Figure 4.9: Serial mechanism formed by two wire flexures for the Criterion 2 calculation [13].

The same example illustrates how quantitative mobility can change as geometry changes. Figure 4.10(b) shows four regions separated by crossings of the relative threshold. From left to right, the reported mobilities are $N = 3$, $N = 5$, $N = 3$, and $N = 4$. The corresponding eigentwists identify whether the large-compliance directions are rotational or translational. This is the essential bridge back to screw theory: the decomposition provides a numerical basis for the actual freedom space, while the reciprocal screw construction provides the target space that the design should realize.




**(a)**




**(b)**

Figure 4.10: Spatial parallel-flexure case study: (a) mechanism geometry and (b) scaled compliance-ratio changes as the limb inclination changes [13].

# Chapter 5

# Mobility Synthesis of Compliant Mechanisms

Mobility synthesis reverses mobility analysis. In analysis, the mechanism is known and the freedom space is unknown. In synthesis, the desired freedom space is known and the mechanism is unknown. The algebraic workflow is therefore direct: specify the desired twist space, compute its reciprocal wrench space, select flexure primitives whose constraint wrenches span that space, and verify that the resulting physical arrangement has the intended rank. The design difficulty is not the null-space calculation. The difficulty is finding realizable flexure geometries whose wrenches or limb spaces match the algebra.

## 5.1 The Algebraic Synthesis Problem

Let the desired stage freedom space be

$$\mathcal{F}_d = \text{range}(\boldsymbol{T}_d), \tag{5.1}$$

where $\boldsymbol{T}_d \in \mathbb{R}^{6\times n}$ has rank $n$. The reciprocal constraint space is

$$\mathcal{C}_d = \text{null}(\boldsymbol{T}_d^{\mathsf{T}}\boldsymbol{\Delta}). \tag{5.2}$$

For an exactly constrained ideal design, $\dim \mathcal{C}_d = 6 - n$. A synthesized mechanism must provide a constraint matrix $\boldsymbol{W}$ such that

$$\text{range}(\boldsymbol{W}) = \mathcal{C}_d \tag{5.3}$$

or at least such that range($\boldsymbol{W}$) contains the required nonredundant constraints while any additional constraints remain compatible with the desired freedom space. The first condition is exact constraint. The second condition allows deliberate redundant constraints inside the same reciprocal space for symmetry, stiffness, thermal stability, or load capacity.

**Principle 5.1** (Synthesis through reciprocity)**.** *A desired motion is synthesized by constraining every reciprocal direction and by avoiding constraints that do virtual work on the desired twists. Algebraically, candidate constraints must satisfy* $\boldsymbol{T}_d^{\mathsf{T}}\boldsymbol{\Delta}\boldsymbol{W} = \boldsymbol{0}$*, and they must have sufficient rank to remove the undesired motions.*

### 5.1.1 General Synthesis Procedure

The synthesis procedure begins by fixing the stage frame and the functional point at which motion is prescribed. Step 1 is to write a full-rank desired twist matrix $\boldsymbol{T}_d$. The columns must describe the actual freedom screws, not merely a scalar degree-of-freedom count. A translation paired with an offset rotation, for example, must retain its translational screw components because those components change the reciprocal constraint space.

Step 2 is to compute the complementary wrench space from $\text{null}(\boldsymbol{T}_d^{\mathsf{T}}\boldsymbol{\Delta})$. This calculation states what the mechanism must resist, but it does not yet prescribe hardware. Step 3 therefore restricts the reciprocal space to a physically realizable family of constraints. For ideal wire flexures, each candidate must be a line-force wrench; for sheet flexures, parallelograms, or compound limbs, the candidate must belong to the relevant element or limb constraint space. This distinction prevents an algebraically correct but mechanically unrealizable wrench basis from being carried into the layout.

Step 4 selects independent candidate constraints from the realizable family and assigns their directions and locations in the stage frame. The locations matter because they determine the moment components of the wrenches. Step 5 forms the candidate matrix $\boldsymbol{W}$ and verifies both $\text{rank}(\boldsymbol{W}) = 6 - n$ and $\text{null}(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Delta}) = \text{range}(\boldsymbol{T}_d)$. If either test fails, the candidate directions or locations are revised and the selection is repeated. Step 6 replaces the ideal constraints with flexure primitives and sizes their geometry so that the required compliance hierarchy, stress limit, and range of motion are achieved. The ideal screw result is therefore a topological design target, not the final geometry.

Figure 5.1 summarizes this six-step process. The two-degree-of-freedom example in Section 5.1.3 follows the same numbered sequence explicitly.

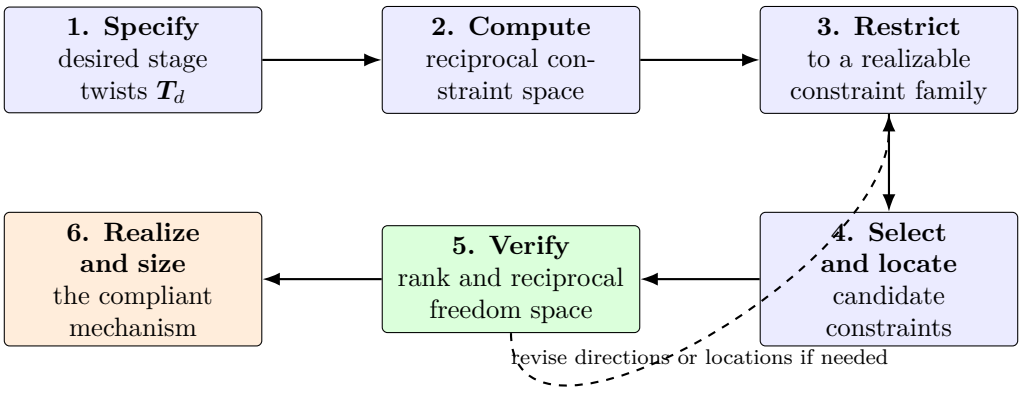



Figure 5.1: General screw-theory synthesis procedure for compliant mechanisms. The verification loop preserves the specified twist space while the geometry of the candidate constraints is refined.

### 5.1.2 Graphical View of Screw-System Synthesis

Graphical freedom-and-constraint representations help designers see families of screw-system solutions without first writing every matrix. Lines, planes, pencils, cylinders, and related entities are graphical encodings of the same reciprocal relationship expressed by $\boldsymbol{T}_d^{\mathsf{T}}\boldsymbol{\Delta}\boldsymbol{W} = \boldsymbol{0}$. In this book, those drawings are used as intuition for the algebra, while the screw-theory calculation remains

the design test. A Mathematica notebook can compute reciprocal spaces, reduce columns, test line-screw realizability, and compare candidate layouts in a way that a drawing alone cannot.

The design theme can therefore be summarized in one sentence. Graphical methods show the shape of the screw-system relationship; screw theory makes that relationship calculable.

This statement is the compact algebraic synthesis process. The corresponding graphical view begins with a desired freedom pattern, identifies the complementary constraint pattern, and then places flexures in compatible regions of that pattern [5, 6]. Screw theory performs the same operation through matrices. The matrix form is especially useful when the desired freedom contains helical screws, offset rotations, or combinations of translation and rotation that are hard to classify visually.

### 5.1.3 Synthesis Example: Two-Degree-of-Freedom Constraint Pattern Design

Constraint pattern design starts from desired freedoms and asks for compatible constraints. The following example uses the six steps in Figure 5.1 to produce a realizable four-wire constraint pattern directly from reciprocal equations [8].

**Step 1: Specify the desired stage twists.** Suppose the desired freedoms are a pure rotation about $z$ and a pure translation in the direction $(0,1,1)^{\mathsf{T}}$:

$$\hat{T}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \hat{T}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}. \tag{5.4}$$

**Step 2: Compute the reciprocal constraint space.** A general reciprocal wrench $\hat{W} = [F_x, F_y, F_z, M_x, M_y, M_z]^{\mathsf{T}}$ must satisfy

$$\hat{T}_1 \circ \hat{W} = 0 \quad \Rightarrow \quad M_z = 0, \qquad \hat{T}_2 \circ \hat{W} = 0 \quad \Rightarrow \quad F_y + F_z = 0. \tag{5.5}$$

**Step 3: Restrict to a realizable constraint family.** If the design is restricted to ideal wire constraints, each selected wrench must also be a line-force wrench,

$$F_x M_x + F_y M_y + F_z M_z = 0, \qquad F_x^2 + F_y^2 + F_z^2 \neq 0. \tag{5.6}$$

Substituting $M_z = 0$ and $F_z = -F_y$ gives the complementary ideal-constraint family

$$\hat{W} = \begin{bmatrix} F_x \\ F_y \\ -F_y \\ M_x \\ M_y \\ 0 \end{bmatrix}, \qquad F_x M_x + F_y M_y = 0, \qquad F_x^2 + 2F_y^2 \neq 0. \tag{5.7}$$

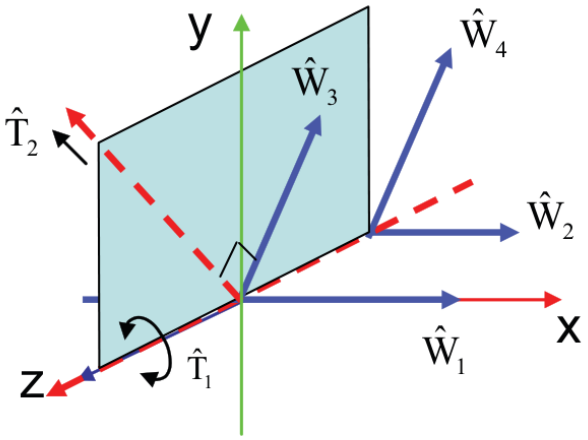



Figure 5.2: Four ideal line constraints for a prescribed two-degree-of-freedom motion space. The desired freedoms are rotation about $z$ and translation in the $(0, 1, 1)$ direction.

**Step 4: Select and locate candidate constraints.** Choosing simple coordinate-aligned cases from this family gives four independent ideal constraints:

$$\hat{W}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{W}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \hat{W}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{W}_4 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{5.8}$$

The first wrench is an $x$-directed line through the origin, and the second is an $x$-directed line through $(0, 0, 1)^\mathsf{T}$. The third is a line through the origin in the $(0, 1, -1)^\mathsf{T}$ direction, while the fourth has the same force direction and passes through $(0, -1, 0)^\mathsf{T}$. These placements reproduce the moment components in the selected wrench columns. Figure 5.2 shows the resulting constraint pattern.

**Step 5: Verify rank and reciprocal freedom space.** Collecting the four constraints as $\boldsymbol{W} = [\hat{W}_1\ \hat{W}_2\ \hat{W}_3\ \hat{W}_4]$ gives

$$\operatorname{rank}(\boldsymbol{W}) = 4 = 6 - 2, \qquad \boldsymbol{T}_d^\mathsf{T} \boldsymbol{\Delta} \boldsymbol{W} = \boldsymbol{0}, \qquad \operatorname{null}(\boldsymbol{W}^\mathsf{T} \boldsymbol{\Delta}) = \operatorname{span}\{\hat{T}_1, \hat{T}_2\}. \tag{5.9}$$

Thus the selected line constraints remove precisely the four undesired instantaneous motions. If their rank had been smaller than four, or if the reciprocal space had included an extra twist, this step would require the directions or locations to be revised before physical realization.

**Step 6: Realize and size the compliant mechanism.** The four ideal line constraints can be implemented with wire flexures placed along the four selected lines, or with equivalent sheet-flexure or compound-limb constraints that span the same wrench space. The physical mechanism must then be sized using the compliance and stiffness procedures of Chapter 6. The design lesson is that the drawing is discovered from the reciprocal family in Eq. (5.7), verified in Eq. (5.9), and only then converted into an elastic mechanism.

## 5.2 Type Synthesis of Compliant Prismatic Joints

The same design procedure gives the canonical compliant prismatic joint. Let the desired freedom be pure translation along $x$,

$$\hat{T}_P = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{5.10}$$

Reciprocity with $\hat{T}_P$ requires $F_x = 0$, so every ideal line constraint in the reciprocal space has the form

$$\hat{W} = \begin{bmatrix} 0 \\ f_y \\ f_z \\ m_x \\ m_y \\ m_z \end{bmatrix}. \tag{5.11}$$

The line-force condition further requires

$$f_y m_y + f_z m_z = 0, \qquad f_y^2 + f_z^2 \neq 0. \tag{5.12}$$

### 5.2.1 Sample Five Independent Ideal Constraints

An exact ideal prismatic joint needs five independent constraints. Setting $m_x = 0$ and choosing coordinate-aligned cases from Eq. (5.12) gives four wrenches,

$$\hat{W}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{W}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \hat{W}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \hat{W}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \tag{5.13}$$

These represent two constraint lines parallel to $y$ and two constraint lines parallel to $z$. A fifth independent ideal constraint is obtained by taking $m_x \neq 0$, for example

$$\hat{W}_5 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \tag{5.14}$$

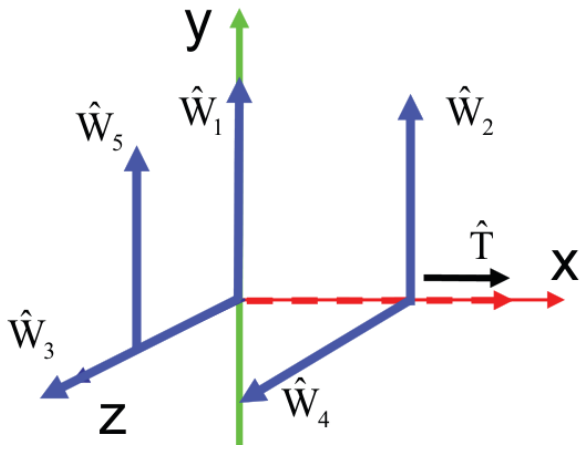



Figure 5.3: Five ideal line constraints spanning the reciprocal constraint space of a compliant prismatic joint [8]. The red dashed axis indicates the desired pure translation.

This is a $y$-directed line force through a point offset in $z$. The five columns are independent, and all are reciprocal to $\hat{T}_P$. Therefore their reciprocal twist space is exactly the desired $x$-translation. Figure 5.3 shows the five line constraints that span this reciprocal space.

### 5.2.2 Physical Realizations of the Same Constraint Space

The algebra now admits several physical realizations. One direct realization uses five wire flexures, each aligned with one of the five line constraints. A second realization replaces three coplanar wire constraints with one sheet flexure and keeps two wire flexures for the remaining constraints. Figure 5.4 compares these two realizations. These two designs have the same intended screw-space behavior even though the hardware is different.

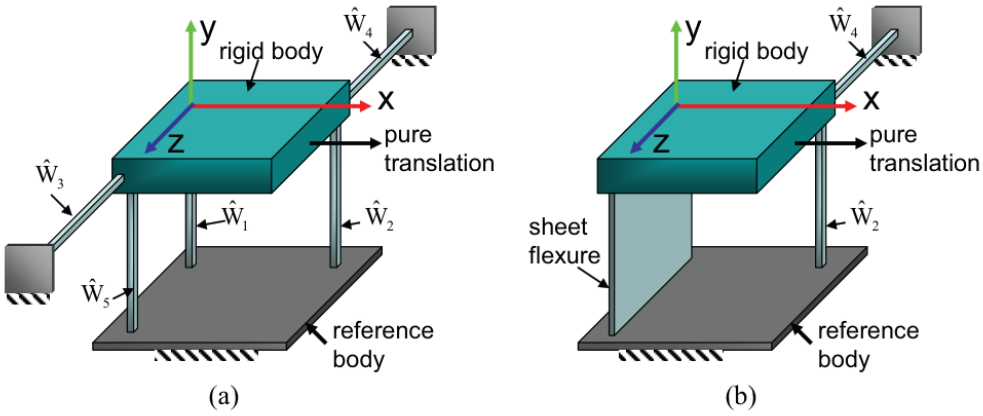



Figure 5.4: Two physical realizations of the same prismatic-joint constraint space [8]: five wire flexures, or one sheet flexure plus two wire flexures.

### 5.2.3 Apply Redundancy and Overconstraint to Physical Design

Exact constraint design often emphasizes the count $f + c = 6$. Compliant mechanism design often uses redundancy deliberately. Additional flexures may be placed in the same reciprocal constraint space to increase stiffness, improve symmetry, reduce thermal drift, distribute stress, or make fabrication practical. Algebraically, a redundant constraint is acceptable if it does not remove a desired freedom. That is, every added wrench $\hat{W}_r$ must satisfy

$$\boldsymbol{T}_d^{\mathsf{T}} \boldsymbol{\Delta} \hat{W}_r = \mathbf{0}. \tag{5.15}$$

If the added wrench lies outside the reciprocal constraint space, it is not benign redundancy; it constrains a desired motion and changes the mobility.

**Remark 5.1** (Redundancy is not a design sin)**.** *In rigid exact-constraint design, overconstraint can create assembly stress and unpredictable load sharing. In compliant mechanism design, redundant flexures can be beneficial because elastic averaging may improve stiffness and symmetry. The algebraic rule is not to avoid redundancy altogether. The rule is to keep redundant constraints inside the reciprocal constraint space of the desired freedom.*

Redundancy can also be introduced deliberately. Add a sixth constraint $\hat{W}_6$ parallel to $\hat{W}_2$ and arranged so that it lies in the span of $\hat{W}_1, \ldots, \hat{W}_5$. Since it does not change the rank or the reciprocal space, it does not alter the ideal mobility. It does, however, allow a second sheet flexure to replace a subset of wire constraints. The resulting two-sheet layout is the familiar parallel-flexure prismatic joint shown in Fig. 5.5. The screw-space point is subtle but important: the sixth physical constraint is acceptable because it is redundant inside the already-correct reciprocal constraint space.

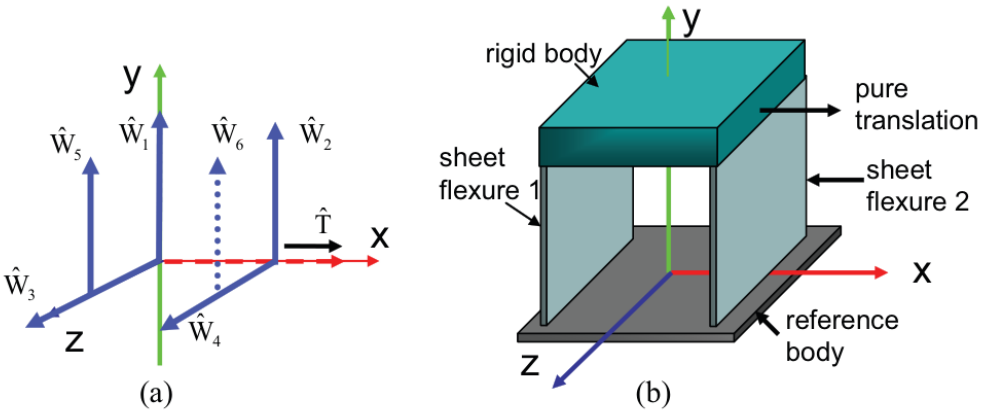



Figure 5.5: Prismatic-joint realization with two parallel sheet flexures [8]. The additional line constraint is redundant in the ideal screw-space model but permits a more practical sheet-flexure architecture.

This example is a useful synthesis template. First specify the desired twist. Second compute the reciprocal ideal-constraint family. Third choose independent line constraints that span the family. Fourth replace groups of line constraints by physical flexure primitives with equivalent constraint spaces. The last step is where design judgment enters: multiple physical mechanisms can realize the same screw-system target.

## 5.3 Synthesis with Parallel Wire Flexures

This section treats the direct synthesis of stages supported only by wire flexures connected in parallel between the functional body and ground. The $i$-th wire has unit direction $\boldsymbol{u}_i$ and passes through point $\boldsymbol{r}_i$, so it contributes the line-force wrench

$$\hat{W}_i = \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{r}_i \times \boldsymbol{u}_i \end{bmatrix}. \tag{5.16}$$

The assembled parallel constraint matrix is

$$\boldsymbol{W} = \begin{bmatrix} \hat{W}_1 & \hat{W}_2 & \cdots & \hat{W}_m \end{bmatrix}. \tag{5.17}$$

For a prescribed $n$-dimensional motion space, synthesis requires $6-n$ independent line-force wrenches in the reciprocal constraint space. Thus one-, two-, and three-degree-of-freedom mechanisms require five, four, and three independent wire constraints, respectively. The rank test is essential: wires with the same direction may be independent when their offsets produce different moment components, whereas visually distinct wires can still yield dependent wrenches. The synthesized motion basis is verified from $\boldsymbol{W}^{\mathsf{T}}\boldsymbol{\Delta T} = \boldsymbol{0}$. Orthogonal-motion synthesis shows that a reciprocal space is not automatically realizable with wires; it must contain a line-screw system of the required rank [9]. Figure 5.6 shows the ideal wire model used in this calculation.

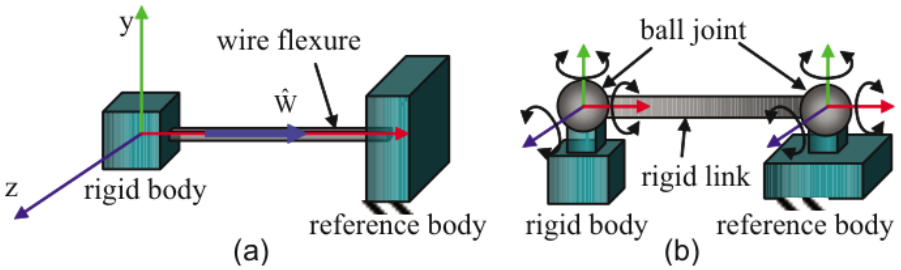



Figure 5.6: Wire-flexure model from [9]. The ideal wire contributes a line-force wrench along its axis; its location contributes the moment part of the wrench through $\boldsymbol{r} \times \boldsymbol{u}$.

**Definition 5.1** (Wire-realizable reciprocal space)**.** *A reciprocal constraint space is wire-realizable in the ideal parallel-wire model if it contains enough independent pure-force line screws to span the required nonredundant constraint space. For an $n$-degree-of-freedom exact design, this means $6-n$ independent line-force wrenches in $\mathcal{C}_d$.*

**Example 5.1** (Pure $x$-translation revisited)**.** *For the desired freedom $\mathcal{F}_d = \operatorname{span}\{\hat{P}_x\}$, Example 3.1 showed that the reciprocal constraints are all wrenches with $F_x = 0$. A wire parallel to $x$ is forbidden because it resists the desired translation. Wires parallel to $y$, $z$, or other directions with zero $x$-force component may be candidates, and their offsets can generate moment components. The synthesis task is to choose five independent line forces in this reciprocal space. The algebra does not merely say "add five wires." It says the five line-force wrenches must be independent and reciprocal to $\hat{P}_x$.*

This is the canonical compliant-slider example. The desired motion is written first as one twist, then the five reciprocal equations are solved, and only afterward are the five pure-force wrenches interpreted as possible wire flexures. That order matters pedagogically. If the designer begins by

sketching wires, the drawing can hide a missing moment constraint or an accidental force along the desired translation. If the designer begins with $\hat{P}_x$ and solves the reciprocal equations, the drawing becomes a realization of a known algebraic target.

A concrete five-wire realization is obtained by choosing two $z$-directed wires and three $y$-directed wires with offsets chosen to generate independent moment components:

$$\begin{aligned}
\hat{W}_1 &= \begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}^{\mathsf{T}}, \quad \hat{W}_2 = \begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}^{\mathsf{T}}, \\
\hat{W}_3 &= \begin{bmatrix}0 & 1 & 0 & -a & 0 & 0\end{bmatrix}^{\mathsf{T}}, \quad \hat{W}_4 = \begin{bmatrix}0 & 0 & 1 & 0 & -b & 0\end{bmatrix}^{\mathsf{T}}, \\
\hat{W}_5 &= \begin{bmatrix}0 & 1 & 0 & 0 & 0 & c\end{bmatrix}^{\mathsf{T}}.
\end{aligned} \tag{5.18}$$

Their assembled constraint matrix is

$$\boldsymbol{W}_x = \begin{bmatrix}0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & -a & 0 & 0 \\ 0 & 0 & 0 & -b & 0 \\ 0 & 0 & 0 & 0 & c\end{bmatrix}. \tag{5.19}$$

For nonzero $a$, $b$, and $c$, the five columns are independent and all have $F_x = 0$, so they are reciprocal to $\hat{P}_x$. The reciprocal-space script in Appendix A checks the case $a = b = c = 1$, obtains $\operatorname{rank}(\boldsymbol{W}_x) = 5$, and returns

$$\operatorname{null}(\boldsymbol{W}_x^{\mathsf{T}}\boldsymbol{\Delta}) = \operatorname{span}\left\{\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}^{\mathsf{T}}\right\}. \tag{5.20}$$

This calculation is the synthesis proof. The drawing is acceptable only because the rank and reciprocal-space checks recover exactly the desired pure $x$-translation. Figure 5.7 gives the corresponding parallel-wire layouts and later one-, two-, and three-degree-of-freedom examples.

**Example 5.2** (Two orthogonal translations)**.** *For the desired freedom space $\mathcal{F}_d = \operatorname{span}\{\hat{P}_x, \hat{P}_y\}$, reciprocal wrenches must satisfy $F_x = 0$ and $F_y = 0$. The reciprocal wrench space has dimension four, but its pure-force content is limited because any pure force in it must be parallel to $z$. Offsetting vertical wires generates moments about $x$ and $y$, but not all four reciprocal wrench directions can necessarily be represented by independent line forces in the required way. This is the kind of degeneracy that motivates line-screw-system criteria for orthogonal-motion synthesis [9].*

The calculation is short enough to do by hand. A general reciprocal wrench for $\operatorname{span}\{\hat{P}_x, \hat{P}_y\}$ has the form

$$\hat{W} = \begin{bmatrix}0 & 0 & F_z & M_x & M_y & M_z\end{bmatrix}^{\mathsf{T}}. \tag{5.21}$$

For this wrench to be realized by an ideal wire, it must be a line force, so $\boldsymbol{F}^{\mathsf{T}}\boldsymbol{M} = F_z M_z = 0$ with $\boldsymbol{F} \neq \boldsymbol{0}$. If the wire actually supplies a force, $F_z \neq 0$, then $M_z = 0$. The available line-force subset is therefore spanned by only three independent parameters $F_z$, $M_x$, and $M_y$. A two-DOF exact spatial design needs four independent constraints, so two pure orthogonal translations cannot be synthesized by ideal wires alone. The lesson is that the reciprocal space has the right dimension, but the wire-realizable part of that space has insufficient rank.

**Example 5.3** (Two orthogonal helical freedoms)**.** *Helical freedoms should be handled algebraically rather than by intuition alone. Let the desired freedom space be generated by two intersecting helical*

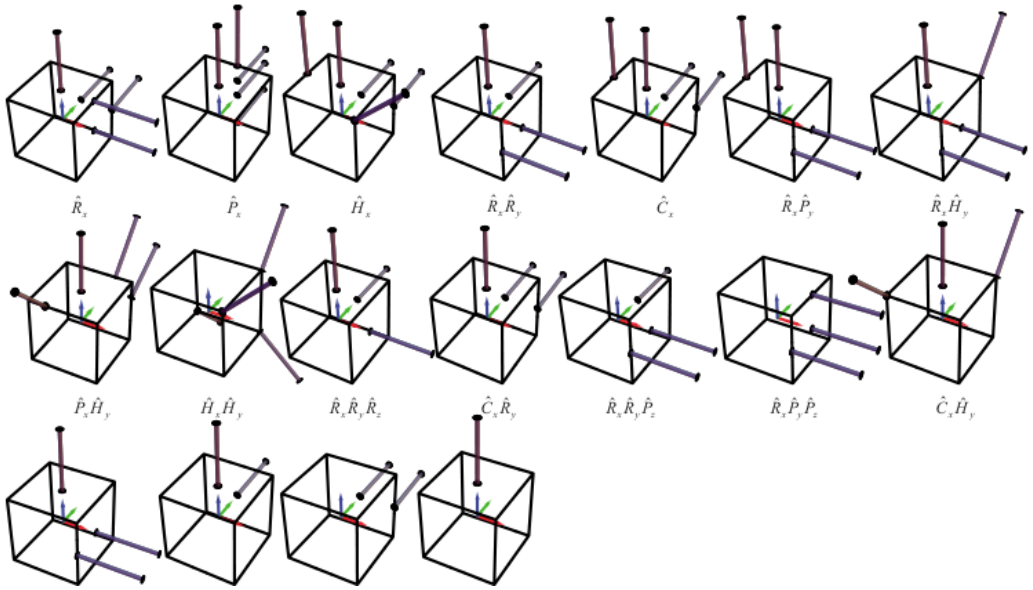


Figure 5.7: Representative parallel-wire flexure mechanisms from [9]. Each wire is a line-force constraint between the stage and ground. The layouts illustrate how specified one-, two-, and three-degree-of-freedom screw systems can be realized when their reciprocal constraint spaces contain sufficiently many independent line wrenches.

*twists about the $x$ and $y$ axes [9],*

$$\boldsymbol{T}_{H_xH_y} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ p_x & 0 \\ 0 & p_y \\ 0 & 0 \end{bmatrix}. \tag{5.22}$$

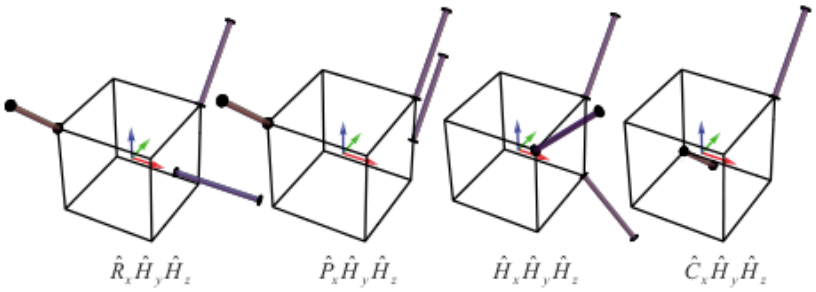
*A reciprocal wrench must satisfy $M_x = -p_xF_x$ and $M_y = -p_yF_y$, so the force-line condition becomes*

$$-p_xF_x^2 - p_yF_y^2 + F_zM_z = 0, \qquad F_x^2 + F_y^2 + F_z^2 \neq 0. \tag{5.23}$$

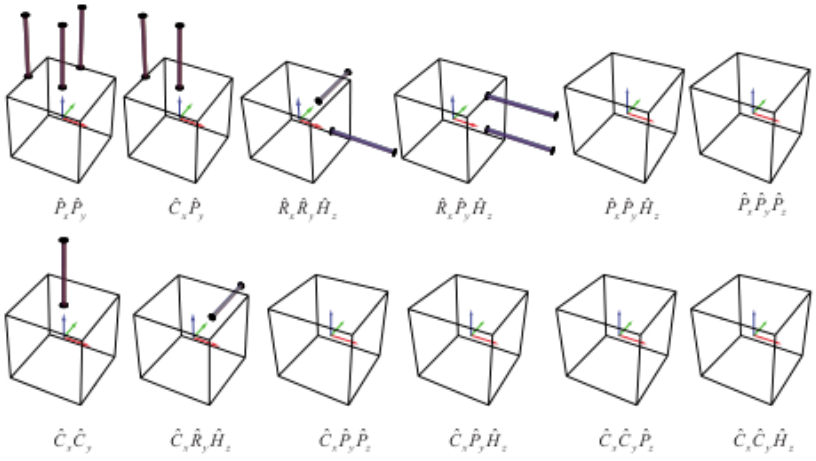
*One four-wire realization in the reciprocal space is*

$$\boldsymbol{W}_{H_xH_y} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ -p_x & -p_x & 0 & 0 \\ 0 & 0 & -p_y & -p_y \\ p_x & -p_x & p_y & -p_y \end{bmatrix}. \tag{5.24}$$

*Each column is a line wrench and is reciprocal to both helical twists. For generic nondegenerate pitches the four columns are independent, giving the required $6-2=4$ constraints. This example is a useful contrast with the two-translation case: both are two-dimensional freedom spaces, but only the helical case has a four-dimensional wire-realizable reciprocal line system.*




**(a)** The four motion spaces that can be realized only when their pitches have different signs




**(b)** The 12 unrealizable motion spaces in which the maximum number of independent wire flexures are shown. The cases with the cubic stage shown only represent that there does not exist any wire flexures in the design.

Figure 5.8: Conditional and unrealizable wire-flexure synthesis cases from [9]. The helical examples show that pitch signs and line-screw rank can decide whether an otherwise valid reciprocal space can be realized with independent wire flexures.

These examples are useful because they show that synthesis is more than dimension counting. The dimension $6-n$ tells us how many independent constraints are needed. The primitive class tells us which shapes of constraints are physically available. A reciprocal space that is valid for a general wrench may not be valid for a design restricted to wires. Figure 5.8 collects conditional and unrealizable cases where pitch signs and line-screw rank decide the outcome.

For three-degree-of-freedom targets, the same construction seeks three independent reciprocal line forces. In the orthogonal class, the following motion spaces are wire-realizable:

$$\begin{aligned} &\operatorname{span}\{\hat{R}_x, \hat{R}_y, \hat{R}_z\}, \quad \operatorname{span}\{\hat{C}_x, \hat{R}_y\}, \\ &\operatorname{span}\{\hat{R}_x, \hat{R}_y, \hat{P}_z\}, \quad \operatorname{span}\{\hat{R}_x, \hat{P}_y, \hat{P}_z\}. \end{aligned} \tag{5.25}$$

Corresponding mixed helical cases are also realizable [9]. In contrast, $\operatorname{span}\{\hat{P}_x, \hat{P}_y, \hat{P}_z\}$ has a reciprocal space of pure moments and therefore contains no line-force wrench; three independent wire constraints cannot be selected. This distinction completes the one- through three-degree-of-freedom design rule: first compute the reciprocal space, then test whether it contains $6-n$ independent line wrenches before drawing a wire layout.

## 5.4 Freedom and Constraint Elements

Section 5.3 established the wire-only route, in which individual line-force constraints are chosen directly from a reciprocal space. The synthesis vocabulary now expands to freedom and constraint elements [12]. A freedom element is a flexure assembly that realizes a desired elementary freedom, such as a rotation or a translation. A constraint element is the dual assembly that removes a desired elementary motion. This language is valuable because real compliant mechanisms are rarely built from one primitive at a time. They are built from reusable blocks.

The catalog categorizes primitives such as blades, wires, notches, and bellows-like springs, then synthesizes rotational and translational freedom elements through parallel connections. By duality, it synthesizes translational and rotational constraint elements through serial chains. Its all-wire entries are physical substitutions within larger element designs; they do not repeat the direct line-screw realizability test of Section 5.3. The result is a catalog of building blocks. In a book-length design workflow, this catalog sits between screw algebra and mechanism architecture. Screw algebra tells us what subspace is needed. The catalog tells us which physical blocks can provide it.

### 5.4.1 Equivalent Freedom and Constraint Spaces

Before using the element catalog, it is worth pausing on the two equivalent-space examples in [12]. They are small, but they are the bridge between geometric screw-system intuition and screw algebra. A serial chain of revolute freedoms with parallel axes can produce a translational freedom. Let the rotations have the same unit axis $\boldsymbol{s}$ and pass through points $\boldsymbol{c}_i$. The twist matrix is

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{s} & \boldsymbol{s} \\ \boldsymbol{c}_1 \times \boldsymbol{s} & \boldsymbol{c}_2 \times \boldsymbol{s} \end{bmatrix}. \tag{5.26}$$

Subtracting the first column from the second leaves

$$\boldsymbol{T} \sim \begin{bmatrix} \boldsymbol{s} & \boldsymbol{0} \\ \boldsymbol{c}_1 \times \boldsymbol{s} & (\boldsymbol{c}_2 - \boldsymbol{c}_1) \times \boldsymbol{s} \end{bmatrix}. \tag{5.27}$$

The second column has zero angular part, so it is a pure translation. This is the algebraic statement behind Fig. 5.9(a): two parallel rotations in series can synthesize one translational freedom.

With three non-coplanar parallel revolute axes, the same column-reduction argument gives two independent pure translations, provided $(\boldsymbol{c}_2 - \boldsymbol{c}_1) \times \boldsymbol{s}$ and $(\boldsymbol{c}_3 - \boldsymbol{c}_1) \times \boldsymbol{s}$ are linearly independent.

The dual calculation explains the constraint-space example. A parallel chain of two translational constraints, such as two wire-like constraints with the same force direction $\boldsymbol{F}$, has wrench matrix

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{F} \\ \boldsymbol{c}_1 \times \boldsymbol{F} & \boldsymbol{c}_2 \times \boldsymbol{F} \end{bmatrix} \sim \begin{bmatrix} \boldsymbol{F} & \boldsymbol{0} \\ \boldsymbol{c}_1 \times \boldsymbol{F} & (\boldsymbol{c}_2 - \boldsymbol{c}_1) \times \boldsymbol{F} \end{bmatrix}. \tag{5.28}$$

The reduced matrix contains one translational constraint and one pure moment constraint. Three non-coplanar parallel translational constraints similarly reduce to one force and two independent moments. Figure 5.9(b) is therefore not merely a picture of wires; it is a row-reduction rule for recognizing how repeated elementary constraints generate complementary patterns.

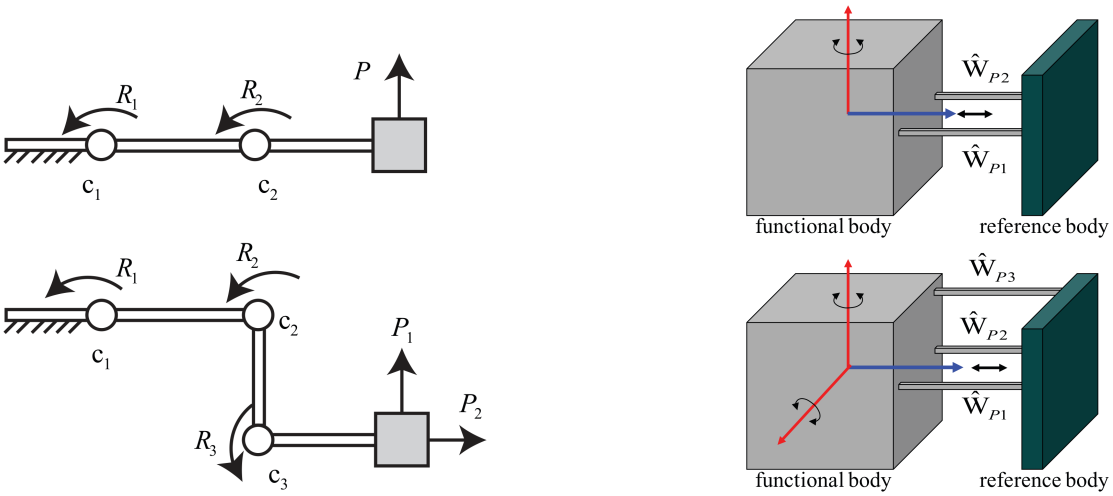



Figure 5.9: Equivalent screw spaces: (a) parallel rotations in a serial chain can synthesize translational freedoms after the rotational columns are reduced algebraically; (b) parallel translational constraints generate one force constraint and one or two moment constraints according to the relative locations of their constraint lines.

**Example 5.4** (R-joint and P-joint synthesis)**.** *A rotational compliant joint, or R-joint, is a freedom element whose dominant freedom space is one rotational twist. A translational compliant joint, or P-joint, is a freedom element whose dominant freedom space is one translational twist. In the screw-algebra view, both are one-dimensional freedom spaces. In physical design, they differ because the reciprocal constraint spaces require different combinations of line forces, blade constraints, or compound flexure limbs. The catalog in [12] is useful precisely because it maps those algebraic requirements to buildable flexure primitives.*

Figure 5.10 previews the two-blade $R$-joint geometry before deriving the blade reciprocity conditions in the next subsection.

### 5.4.2 Rotational Freedom Elements: $R$-Joints

Begin with one-degree-of-freedom rotational elements. Let the desired rotational freedom be an $R$-joint with axis direction $\boldsymbol{\Omega}$ through the origin [12],

$$\hat{T}_R = \begin{bmatrix} \boldsymbol{\Omega} \\ \boldsymbol{0} \end{bmatrix}. \tag{5.29}$$

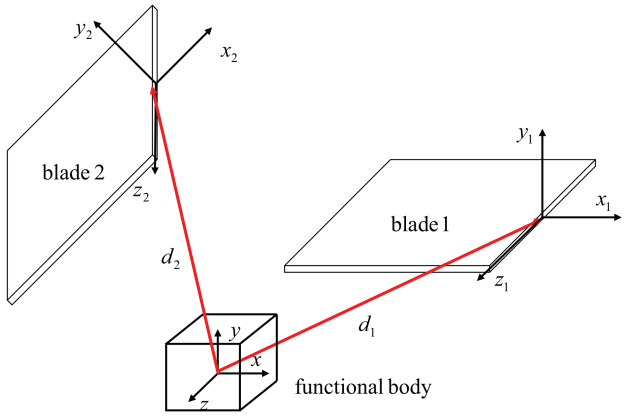



Figure 5.10: Two-blade $R$-joint synthesis geometry from [12]. The blade normals, blade positions, and desired rotation axis must satisfy the reciprocal conditions in Eqs. (5.31)–(5.32).

For a blade flexure transformed into the functional-body frame, the ideal wrench matrix can be written as

$$\boldsymbol{W}_i = \begin{bmatrix} \boldsymbol{x}_i & \boldsymbol{z}_i & \boldsymbol{0} \\ \boldsymbol{d}_i \times \boldsymbol{x}_i & \boldsymbol{d}_i \times \boldsymbol{z}_i & \boldsymbol{y}_i \end{bmatrix}, \tag{5.30}$$

where $\boldsymbol{y}_i$ is the blade normal, $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$ lie in the blade plane, and $\boldsymbol{d}_i$ locates the blade relative to the functional body. Reciprocity with $\hat{T}_R$ gives

$$\boldsymbol{\Omega} \cdot (\boldsymbol{d}_i \times \boldsymbol{x}_i) = 0, \qquad \boldsymbol{\Omega} \cdot (\boldsymbol{d}_i \times \boldsymbol{z}_i) = 0, \qquad \boldsymbol{\Omega} \cdot \boldsymbol{y}_i = 0. \tag{5.31}$$

For a blade, the first two equations reduce to a geometric placement rule:

$$\boldsymbol{d}_i \cdot \boldsymbol{y}_i = 0, \qquad \boldsymbol{\Omega} \cdot \boldsymbol{y}_i = 0. \tag{5.32}$$

Thus the blade attachment vector must lie in the blade plane, and the blade normal must be perpendicular to the desired rotation axis. For example, when $\boldsymbol{\Omega} = \boldsymbol{e}_x$, choosing two independent blade normals $\boldsymbol{y}_1 = \boldsymbol{e}_y$ and $\boldsymbol{y}_2 = \boldsymbol{e}_z$ gives two noncoplanar blade constraints that remove the five unwanted motions while leaving rotation about $x$. This is the algebra behind the familiar crossed-blade or two-blade flexure hinge.

Figure 5.11 collects all eleven $R$-joint examples in the catalog. Cases 1–3 use only blades and wires: two blades $2B$, one blade plus two wires $B$-$2W$, and five wires $5W$. Cases 4–5 use spherical notches and wires: two spherical notches $2S$, and one spherical notch plus two wires $S$-$2W$. Cases 6–11 add bellows springs, which act as rotational constraints in the catalog logic: $B_s$-$B$-$S$, $B_s$-$B$-$W$, $B_s$-$S$-$W$, $B_s$-$4W$, $2B_s$-$S$, and $2B_s$-$3W$. Table 5.1 records the corresponding primitive patterns and algebraic design ideas. The point of the list is not that these are the only possible $R$-joints; the point is that each one is produced by the same reciprocal-space test.

The $S$-$2W$ case is a useful calculation to show students because it makes offset wire constraints explicit. A spherical notch has constraint basis $\mathrm{span}\{\hat{P}_x, \hat{P}_y, \hat{P}_z\}$, so it removes translations and leaves three rotations. A wire parallel to $y$ at an $x$-offset $d$ contributes a wrench whose moment component removes rotation about $z$. A wire parallel to $z$ at the same offset removes rotation about $y$. The assembled constraint matrix therefore has five independent columns, and its reciprocal space is $\mathrm{span}\{\hat{R}_x\}$. This is the same rank-and-reciprocity calculation as in the wire examples, but the primitive vocabulary is richer.

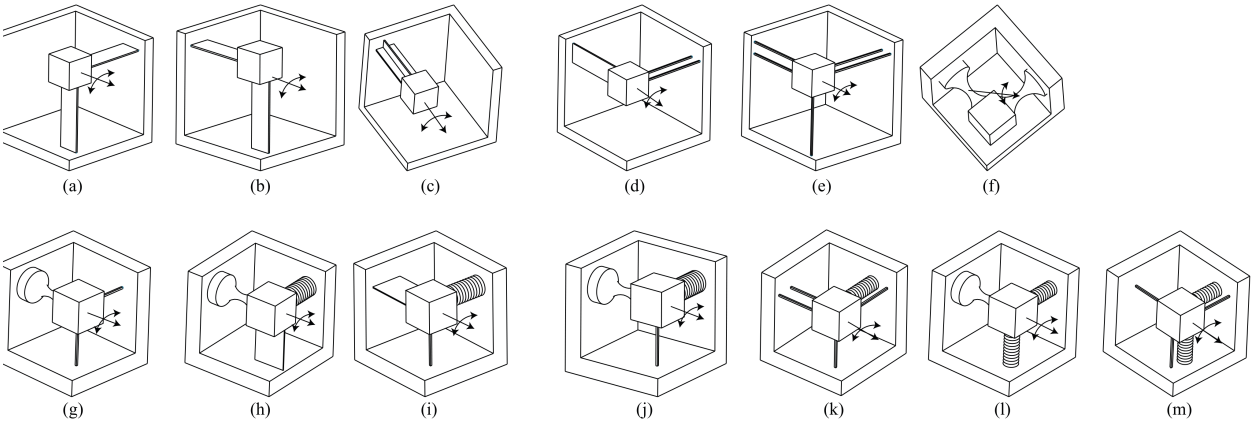



Figure 5.11: Eleven $R$-joint freedom-element designs from [12]. The catalog is organized by primitive combinations: blades $B$, wires $W$, spherical notches $S$, and bellows springs $B_s$.

Table 5.1: The eleven $R$-joint examples synthesized in [12].

| Case | Primitive pattern | Algebraic design idea |
|---|---|---|
| 1 | $2B$ | Two noncoplanar blade constraint spaces satisfy Eq. (5.32) and leave one rotation. |
| 2 | $B$-$2W$ | One blade is retained; the missing blade constraint is replaced by two parallel wire constraints. |
| 3 | $5W$ | A blade-equivalent constraint set is realized by three coplanar wires, giving an all-wire hinge. |
| 4 | $2S$ | Two spherical notches in parallel leave rotation about the line joining their centers. |
| 5 | $S$-$2W$ | The spherical notch removes translations; two offset wires remove the remaining unwanted rotations. |
| 6 | $B_s$-$B$-$S$ | A bellows spring removes one of the two rotations left by a blade-spherical-notch parallel structure. |
| 7 | $B_s$-$B$-$W$ | A bellows spring and a wire remove the extra rotation and translation of a blade. |
| 8 | $B_s$-$S$-$W$ | A bellows spring and an offset wire remove two rotations from the $3R$ freedom of a spherical notch. |
| 9 | $B_s$-$4W$ | Wires remove translations and one additional rotation; the bellows spring removes another rotation. |
| 10 | $2B_s$-$S$ | A spherical notch removes translations, and two bellows springs remove two rotations. |
| 11 | $2B_s$-$3W$ | Three wires remove translations, and two bellows springs remove two rotations. |

### 5.4.3 Translational Freedom Elements: $P$-Joints

The translational freedom element is the dual one-degree-of-freedom target: a $P$-joint with translation direction $\boldsymbol{V}$,

$$\hat{T}_P = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{V} \end{bmatrix}. \tag{5.33}$$

For a blade primitive, reciprocity with $\hat{T}_P$ requires

$$\boldsymbol{V} \cdot \boldsymbol{x}_i = 0, \qquad \boldsymbol{V} \cdot \boldsymbol{z}_i = 0. \tag{5.34}$$

Since $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$ span the blade plane, Eq. (5.34) says that the desired translation direction must be parallel to the blade normal $\boldsymbol{y}_i$. The familiar parallel-sheet translational joint is therefore not a drawing rule learned by imitation; it is the reciprocal-space condition for a $P$-joint.

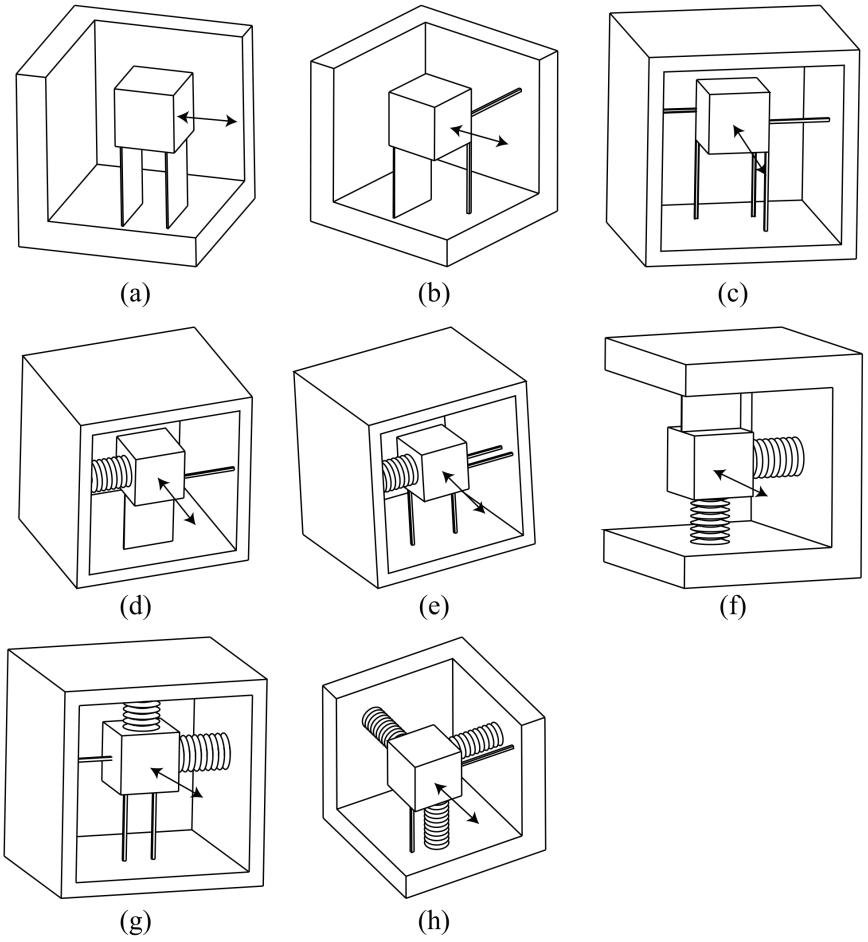



Figure 5.12: Eight $P$-joint freedom-element designs from [12]. The arrowed lines indicate the allowed translation of the functional body.

Figure 5.12 gives the eight translational-joint examples in the catalog. The first three cases use blades and wires only: $2B$, $B$-$2W$, and $5W$. The remaining five cases use bellows springs to remove rotational freedoms: $B_s$-$B$-$W$, $B_s$-$4W$, $2B_s$-$B$, $2B_s$-$3W$, and $3B_s$-$2W$. Table 5.2 summarizes these eight patterns. The $5W$ and $B_s$-$4W$ cases also illustrate a computational synthesis choice. Starting

from the reciprocal space of a pure translation, one can choose a basis consisting entirely of force wrenches for an all-wire realization, or choose four force wrenches plus one couple wrench and realize the couple with a bellows spring. The same mathematical reciprocal space therefore supports different physical embodiments.

Table 5.2: The eight $P$-joint examples synthesized in [12].

| Case | Primitive pattern | Algebraic design idea |
|---|---|---|
| 1 | $2B$ | Two blade normals align with the desired translation direction, giving the parallel-sheet $P$-joint. |
| 2 | $B$-$2W$ | One blade is replaced by two wires parallel to the blade plane. |
| 3 | $5W$ | A basis for the reciprocal space of $\hat{T}_P$ is selected as five realizable line-force wrenches. |
| 4 | $B_s$-$B$-$W$ | A wire removes one blade rotation and a bellows spring removes the second. |
| 5 | $B_s$-$4W$ | Four wires supply force wrenches and one bellows spring supplies a couple wrench. |
| 6 | $2B_s$-$B$ | Two bellows springs remove the two rotations left by the blade. |
| 7 | $2B_s$-$3W$ | Three coplanar wires replace the blade in the $2B_s$-$B$ case. |
| 8 | $3B_s$-$2W$ | Three bellows springs and two wires together remove the five unwanted motions. |

### 5.4.4 Translational Constraint Elements: $P$-Constraints

Constraint elements reverse the viewpoint. A $P$-constraint removes one translation and allows the complementary five-dimensional freedom space. With the constrained force direction $\boldsymbol{F}$, the target wrench is

$$\hat{W}_P = \begin{bmatrix} \boldsymbol{F} \\ \boldsymbol{0} \end{bmatrix}, \tag{5.35}$$

and the serial chain must provide a freedom space reciprocal to $\hat{W}_P$. Wire and bellows primitives are excluded for this particular synthesis because they already behave like elementary constraints of the wrong type. The catalog therefore uses spherical notches $S$, blades $B$, and notch hinges $R$ in serial chains [12].

The $S$-$S$ case is the cleanest calculation. Put one spherical notch at the origin and the other at $d\boldsymbol{e}_x$. Concatenating their rotational twist bases gives a serial freedom matrix. Subtracting corresponding rotational columns leaves three rotations plus translations in $y$ and $z$; the $x$-translation is absent. The reciprocal constraint is therefore a pure force along $x$. Figure 5.13 shows the $S$-$B$ geometry for the next case.

In the $S$-$B$ case, reciprocity requires the spherical notch to lie on the desired force line and the blade normal to be perpendicular to that force direction. The intersection of these simple geometric facts is the design rule. This example is useful because it shows how the serial-chain vocabulary turns a visual flexure sketch into a small set of geometric incidence and perpendicularity requirements.

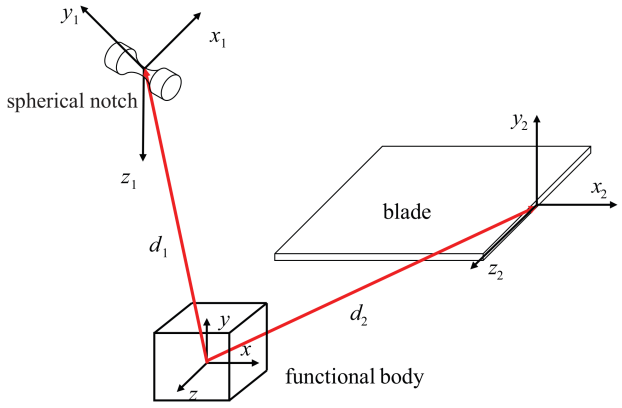



Figure 5.13: *S-B* geometry for a translational constraint element from [12]. The spherical notch position and blade normal must both satisfy the reciprocal conditions for the prescribed force direction.

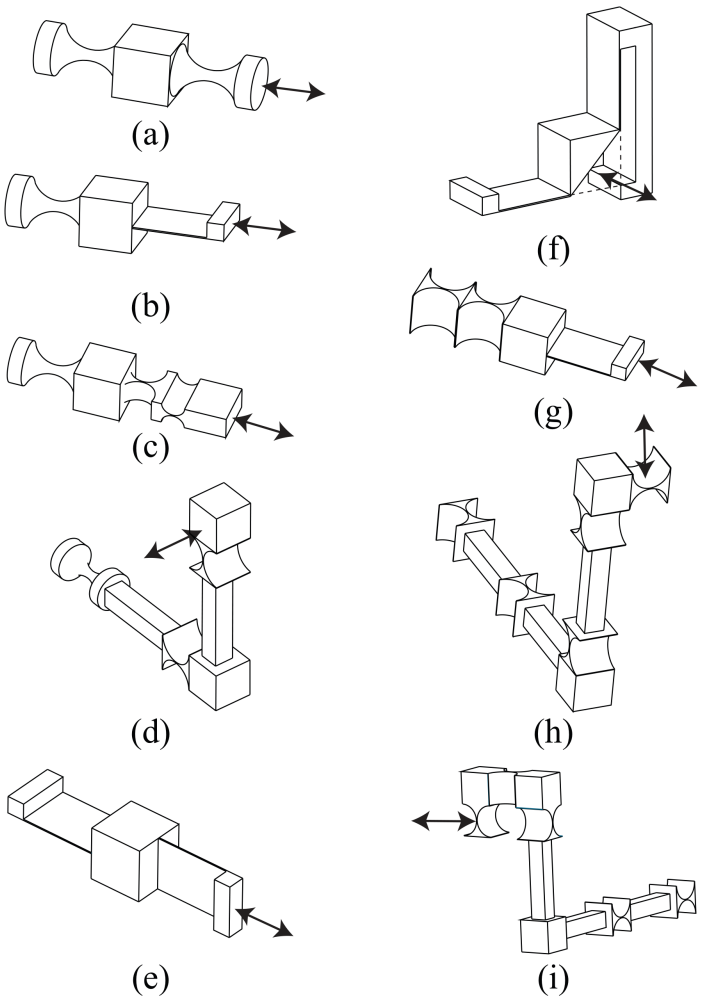



Figure 5.14: Six $P$-constraint designs from [12]. These are serial chains whose complementary freedom space has three rotations and two translations, leaving one removed translation.

Figure 5.14 gives the six $P$-constraint examples. Two spherical notches $S$-$S$ constrain translation along the line joining their centers. An $S$-$B$ chain constrains translation when the spherical notch lies on the desired force line and the blade normal is perpendicular to that line. The $S$-$2R$ case has two variants: two perpendicular $R$-joint axes paired with the spherical notch, or two parallel $R$-joint axes coplanar with one spherical-notch rotation. The $B$-$B$ case also has two variants depending on whether the constraint line follows the blade length or width direction. Table 5.3 lists the six serial-chain patterns. The $B$-$2R$ and $5R$ cases use parallel-rotation facts: two parallel rotations produce one translation, and three parallel rotations produce two translations.

### 5.4.5 Rotational Constraint Elements: $R$-Constraints

An $R$-constraint removes one rotation and is represented by the pure couple wrench

$$\hat{W}_R = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{M} \end{bmatrix}. \tag{5.36}$$

The complementary freedom space has two rotations and three translations. In the 2013 catalog, only blades and notch hinges are needed. For the $B$-$B$ case, reciprocity between a blade twist

Table 5.3: The six $P$-constraint examples synthesized in [12].

| Case | Serial-chain pattern | Algebraic design idea |
|---|---|---|
| 1 | $S$-$S$ | Two spherical centers define the constrained translation line. |
| 2 | $S$-$B$ | The spherical notch lies on the force line; the blade normal is perpendicular to the force direction. |
| 3 | $S$-$2R$ | Two notch hinges supply the two translations missing from the spherical notch freedom space. |
| 4 | $B$-$B$ | Two nonparallel blades satisfy the same translational-constraint reciprocity conditions. |
| 5 | $B$-$2R$ | Two parallel notch hinges add the translation needed to complement the blade. |
| 6 | $5R$ | Parallel groups of notch hinges generate the two translations in the complementary freedom space. |

matrix and $\hat{W}_R$ gives

$$\boldsymbol{M} \cdot \boldsymbol{x}_i = 0, \qquad \boldsymbol{M} \cdot \boldsymbol{z}_i = 0, \tag{5.37}$$

so both blade normals must be parallel to $\boldsymbol{M}$. The two blades must also be separated by a nonzero distance; otherwise the translational part of the complementary freedom space degenerates.

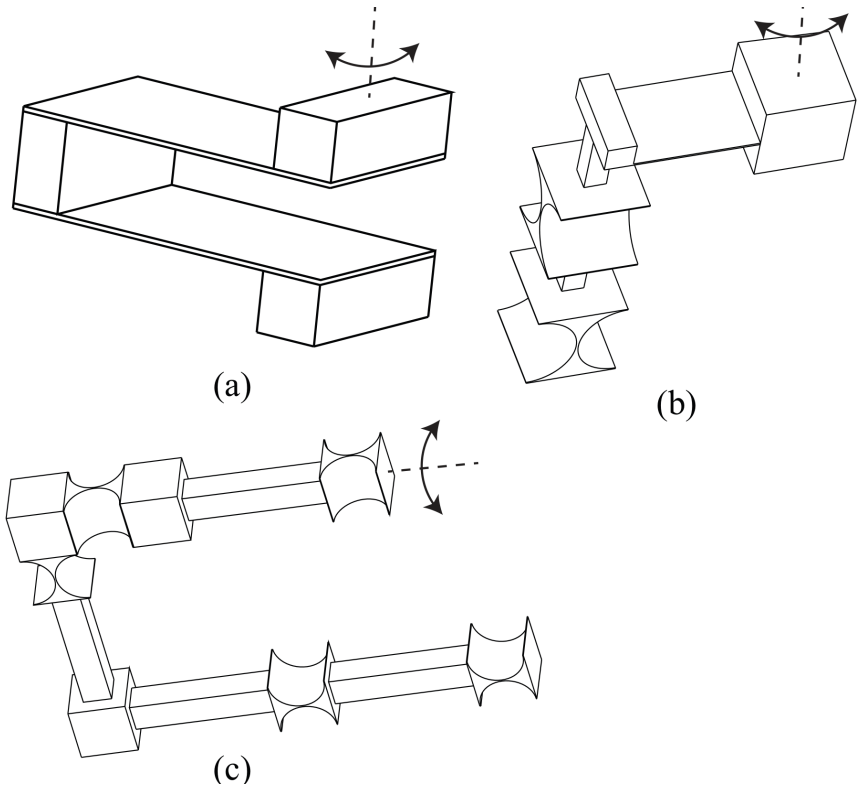



Figure 5.15: Three $R$-constraint designs from [12]. The arrowed arcs indicate the constrained rotation.

Figure 5.15 gives the three rotational-constraint examples: $B$-$B$, $B$-$2R$, and $5R$. The first uses two separated parallel blades whose normals align with the constrained moment direction. The

second uses a blade plus two notch hinges aligned with the blade bending and torsion directions so that the missing translations are supplied by the hinges. The third uses five notch hinges arranged as two groups of parallel rotations in perpendicular planes. Table 5.4 records the same cases in compact algebraic form.

Table 5.4: Rotational constraint element cases in [12].

| Case | Serial-chain pattern | Algebraic design idea |
|---|---|---|
| 1 | $B$-$B$ | Two separated blades share the desired moment-constraint direction. |
| 2 | $B$-$2R$ | A blade supplies the force-like constraints while two notches supply complementary rotations. |
| 3 | $5R$ | Five notch hinges are arranged so their serial freedom complement leaves one constrained rotation. |

### 5.4.6 Hybrid Structures from the Element Catalog

The last set of catalog examples demonstrates how the element catalog composes into larger mechanisms. The important lesson is that a designer can replace a primitive in a catalog element by any submechanism with the same freedom or constraint space. This is where screw theory becomes architectural: once the subspace is known, the designer can swap physical realizations while preserving type.

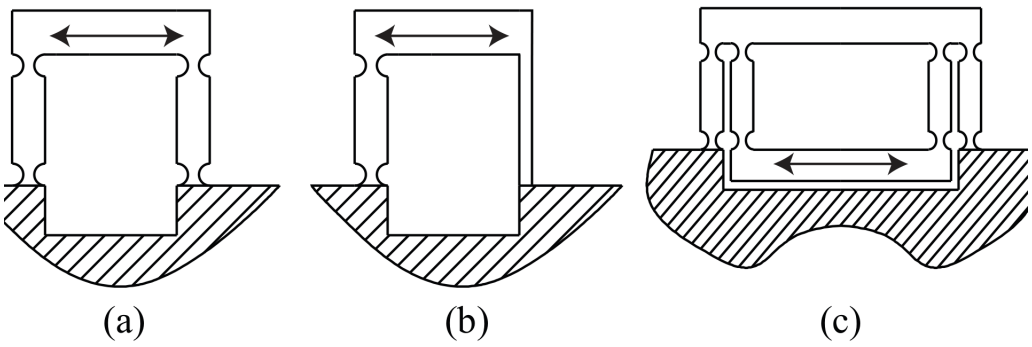
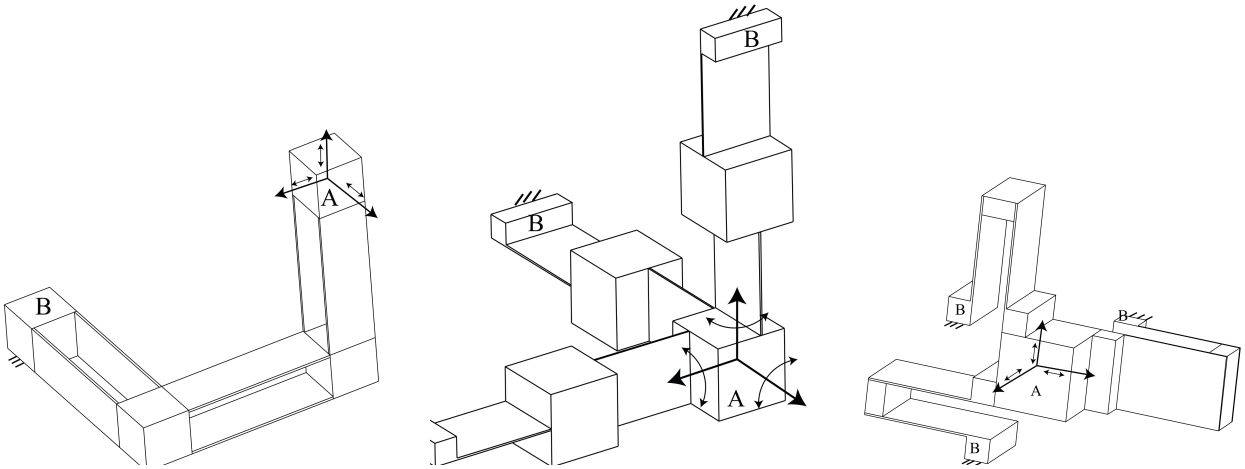



Figure 5.16: Three hybrid $P$-joint designs from [12]: a four-notch parallelogram, a mixed blade/notch version, and a double parallelogram for larger stroke and reduced parasitic error.

Figure 5.16 shows three hybrid $P$-joint examples. The classic parallelogram is a parallel assembly of two serial $R$-$R$ limbs. Replacing one $R$-$R$ limb with a blade gives a mixed hybrid design with the same intended translation. Connecting two parallelograms in series produces a double parallelogram, a common way to reduce parasitic rotation and increase usable displacement stroke.

Figure 5.17 collects three composition examples from the element catalog: a serial $PPP$ chain, a parallel assembly of translational constraints, and its dual parallel assembly of rotational constraints.

The serial chain in Fig. 5.17(a) realizes three translations by composing one-dimensional translational freedom elements. Each local $P$-joint contributes one dominant translation, and the serial topology adds these motion spaces in the same way that the algebraic mobility calculation concatenates twist columns. This example is useful because it shows how a catalog element becomes




(a) Serial $PPP$ chain.

(b) Parallel $P$-constraint assembly.

(c) Parallel $R$-constraint assembly.

Figure 5.17: Composition of flexure elements from [12]. (a) Three $P$-joint elements form a spatial serial $PPP$ chain. (b) Three translational constraint elements assembled in parallel remove the translations of the functional body and leave three rotations. (c) Three rotational constraint elements assembled in parallel remove rotations and leave three translations.

a reusable module rather than a one-off geometry.

The parallel $P$-constraint assembly in Fig. 5.17(b) removes the three translations of the functional body. Since each limb contributes a translational constraint, the parallel topology adds wrench spaces rather than twist spaces. The surviving freedom space is therefore rotational: the body can rotate about its center while translational motion is constrained.

The parallel $R$-constraint assembly in Fig. 5.17(c) removes the rotational freedoms of the functional body and leaves the three translations indicated by the arrows. These three composition examples close the loop between primitive-level screw algebra and mechanism-level type synthesis: serial composition adds motion spaces, while parallel composition adds constraint spaces.

# Chapter 6

# Stiffness and Compliance Analysis

Mobility analysis treats a compliant mechanism as if some directions were perfectly free and other directions perfectly constrained. Stiffness analysis removes that idealization. Every flexure direction has finite compliance. The design question becomes quantitative: how much does the stage move under a wrench, which compliance terms are desirable, which terms are parasitic, and how do geometry and material parameters change the matrix? The symbolic compliance formulation is the main source for this chapter [11]. Once the matrices are available, their use in estimating effective mobility is described in Section 4.6.1.

## 6.1 Compliance and Stiffness Matrices

Let the stage deformation be the twist

$$\hat{T} = \begin{bmatrix} \theta_x & \theta_y & \theta_z & \delta_x & \delta_y & \delta_z \end{bmatrix}^{\mathsf{T}} \tag{6.1}$$

and let the applied load be the wrench

$$\hat{W} = \begin{bmatrix} F_x & F_y & F_z & M_x & M_y & M_z \end{bmatrix}^{\mathsf{T}}. \tag{6.2}$$

Under small deformation and linear elasticity, the two are related by

$$\hat{T} = \boldsymbol{C}\hat{W}, \qquad \hat{W} = \boldsymbol{K}\hat{T}, \qquad \boldsymbol{C} = \boldsymbol{K}^{-1} \tag{6.3}$$

when the matrix is nonsingular. The $6 \times 6$ matrix $\boldsymbol{C}$ is the compliance matrix and $\boldsymbol{K}$ is the stiffness matrix. Figure 6.1 fixes the loading and displacement convention used for the matrix entries.

The entry $c_{ij}$ is the $i$-th motion component caused by the $j$-th load component. For example, using the ordering above, $c_{41} = \delta_x/F_x$ is the translational compliance along $x$ under a force along $x$, while $c_{21} = \theta_y/F_x$ is a rotation about $y$ caused by the same force. The matrix form is not merely compact notation. It is the bridge between qualitative screw design and quantitative performance. A one-degree-of-freedom translational stage should have one large translational compliance and five much smaller compliances in the constrained directions. A rotational flexure pivot should have a large rotational compliance about its intended axis and small parasitic translations at its functional point. A stiffness matrix lets the designer compare these quantities directly.

## 6.2 Coordinate Transformation of Compliance and Stiffness

Element matrices are easiest to derive in local coordinates. Mechanism matrices must be assembled in the stage frame. With the adjoint convention of Chapter 3, a local twist or wrench is transformed

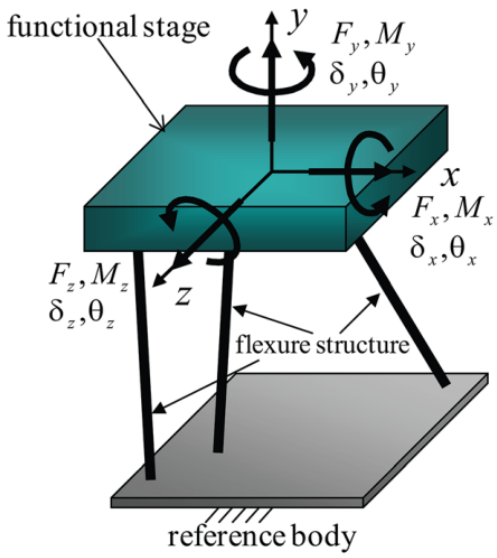



Figure 6.1: Six-component loading and displacement convention.

by

$$\hat{T}' = \mathrm{Ad}\ \hat{T}, \qquad \hat{W}' = \mathrm{Ad}\ \hat{W}. \tag{6.4}$$

The local compliance relation is $\hat{T} = \boldsymbol{C}\hat{W}$. Substituting the coordinate transformation gives

$$\hat{T}' = \mathrm{Ad}\,\hat{T} = \mathrm{Ad}\,\boldsymbol{C}\hat{W} = \mathrm{Ad}\,\boldsymbol{C}\,\mathrm{Ad}^{-1}\,\hat{W}'. \tag{6.5}$$

Therefore the transformed compliance matrix is

$$\boldsymbol{C}' = \mathrm{Ad}\,\boldsymbol{C}\,\mathrm{Ad}^{-1}\,. \tag{6.6}$$

Similarly,

$$\boldsymbol{K}' = \mathrm{Ad}\,\boldsymbol{K}\,\mathrm{Ad}^{-1}\,. \tag{6.7}$$

These are the transformation formulas used in symbolic compliance analysis and mirrored in the Mathematica notebooks through an `AdTrans` function [11].

**Remark 6.1** (Coordinate conventions)**.** *Some texts use a dual adjoint for wrench coordinates, leading to congruence transformations of stiffness and compliance. The convention above is used consistently throughout this book. Mixing conventions is a common source of sign and transpose errors. A project should choose one convention, state it, and use it consistently.*

## 6.3 Serial and Parallel Composition of Matrices

For serial flexure chains, the same load path passes through elements while deformations add. After transforming each element compliance matrix into the common stage frame, the serial-chain compliance is

$$\boldsymbol{C}_{\mathrm{ser}} = \sum_{j=1}^{m} \boldsymbol{C}'_j, \tag{6.8}$$

with

$$\boldsymbol{K}_{\mathrm{ser}} = \boldsymbol{C}_{\mathrm{ser}}^{-1} \tag{6.9}$$

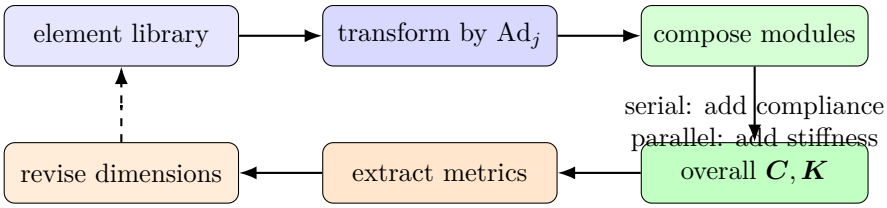



Figure 6.2: Compliance analysis workflow. Local element matrices are transformed to the stage frame, recursively composed according to topology, and then reduced to design metrics.

when the inverse exists. For parallel chains, the stage deformation is shared while wrenches add. After transforming each branch stiffness into the stage frame, the parallel stiffness is

$$\boldsymbol{K}_{\mathrm{par}} = \sum_{j=1}^{m} \boldsymbol{K}'_j, \tag{6.10}$$

with

$$\boldsymbol{C}_{\mathrm{par}} = \boldsymbol{K}_{\mathrm{par}}^{-1} \tag{6.11}$$

when the inverse exists. Figure 6.2 summarizes this transform–compose–reduce workflow.

These composition rules are the stiffness analogs of the mobility rules in Chapter 4. Serial mobility was assembled by adding motion spaces; serial compliance is assembled by adding compliance matrices. Parallel mobility was assembled by adding constraint spaces; parallel stiffness is assembled by adding stiffness matrices. This analogy is worth remembering because it keeps the topology logic consistent across qualitative and quantitative analysis.

## 6.4 General Derivation for a Flexure Mechanism

A general compliant mechanism can be analyzed with the following derivation.

1. Build a library of element compliance or stiffness matrices in local coordinates. These matrices are functions of material parameters, such as Young's modulus and shear modulus, and geometric parameters, such as length, width, thickness, notch radius, or wire diameter.
2. Identify the mechanism topology as a hierarchy of serial and parallel modules.
3. For each element or submodule, compute the coordinate transformation from its local frame to the module or stage frame.
4. Transform each local matrix using $\boldsymbol{C}' = \mathrm{Ad}\,\boldsymbol{C}\,\mathrm{Ad}^{-1}$ or $\boldsymbol{K}' = \mathrm{Ad}\,\boldsymbol{K}\,\mathrm{Ad}^{-1}$.
5. Add compliance matrices for serial modules and stiffness matrices for parallel modules.
6. Proceed upward until the stage-level $\boldsymbol{C}$ and $\boldsymbol{K}$ are obtained.

Figure 6.3 shows the blade-flexure element as the local matrix model before these transformations are applied.

The derivation is recursive because a module can be treated as an element once its matrix has been found. A serial chain of flexures becomes a module compliance. Several such chains in parallel become a module stiffness. Several modules may then be connected into a larger hybrid architecture. This is the same top-down and bottom-up idea used for mobility, now applied to quantitative matrices.

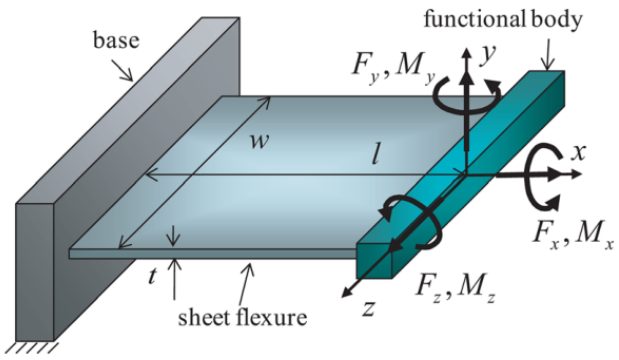



Figure 6.3: Blade-flexure element. The local element matrix is derived in the blade frame, then transformed into the stage frame before serial or parallel assembly.

## 6.5 Example: Single Flexure to Two Parallel Flexures

Let an idealized flexure element have local compliance $\boldsymbol{C}_e$ and local stiffness $\boldsymbol{K}_e = \boldsymbol{C}_e^{-1}$. Suppose two identical copies are placed in parallel between the stage and ground, with adjoint transformations $\mathrm{Ad}_1$ and $\mathrm{Ad}_2$. Their transformed stiffness matrices are

$$\boldsymbol{K}_1' = \mathrm{Ad}_1\, \boldsymbol{K}_e\, \mathrm{Ad}_1^{-1}, \qquad \boldsymbol{K}_2' = \mathrm{Ad}_2\, \boldsymbol{K}_e\, \mathrm{Ad}_2^{-1}\,. \tag{6.12}$$

The stage stiffness is

$$\boldsymbol{K}_{2\mathrm{par}} = \boldsymbol{K}_1' + \boldsymbol{K}_2', \tag{6.13}$$

and the compliance is

$$\boldsymbol{C}_{2\mathrm{par}} = \boldsymbol{K}_{2\mathrm{par}}^{-1}. \tag{6.14}$$

This is the pattern repeatedly used in the compliance notebooks for two parallel identical flexures. The same pattern also shows why symmetry matters. If the two flexures are placed symmetrically, certain off-diagonal compliance terms may cancel. If the placement is asymmetric, those terms remain and appear as parasitic rotations or translations.

The compact WolframScript version of the notebook uses the local symbolic compliance model

$$\boldsymbol{C}_e = \begin{bmatrix} 0 & 0 & 0 & c_{\theta x} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{\theta y} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{\theta z} \\ c_{\delta x} & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{\delta y} & 0 & 0 & 0 & 0 \\ 0 & 0 & c_{\delta z} & 0 & 0 & 0 \end{bmatrix}. \tag{6.15}$$

Because the wrench order is $[F_x, F_y, F_z, M_x, M_y, M_z]^{\mathsf{T}}$, the lower-left block maps forces to translations and the upper-right block maps moments to rotations. For two identical flexures placed at $y = \pm s/2$, the script computes $\boldsymbol{K}_{2\mathrm{par}}$, inverts it, and obtains the direct translational compliance

$$c_{41}^{(2par)} = \frac{\delta_x}{F_x} = \frac{c_{\delta x}}{2}. \tag{6.16}$$

This result is the matrix version of two equal springs in parallel. At the same time, the printed stiffness matrix contains geometry-induced terms such as

$$k_{41}^{(2par)} = \frac{2}{c_{\theta x}} + \frac{s^2}{2c_{\delta z}}, \qquad k_{63}^{(2par)} = \frac{2}{c_{\theta z}} + \frac{s^2}{2c_{\delta x}}, \tag{6.17}$$

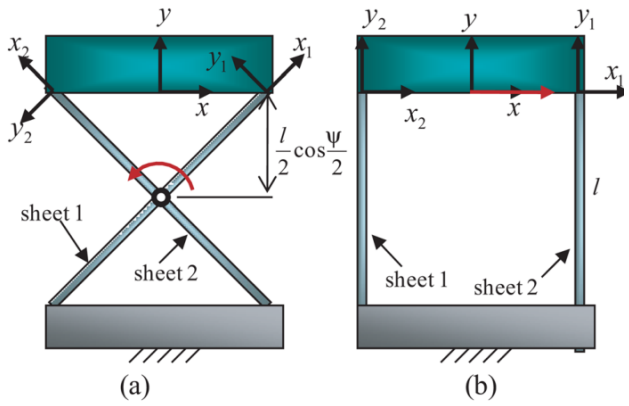



Figure 6.4: Cross-strip and parallelogram-style compliance examples from [11]. These mechanisms use both transformation and serial/parallel composition, so the symbolic matrix exposes parasitic terms that are difficult to see from mobility alone.

which show how offsets convert translational stiffness into rotational stiffness about the stage frame.

For two orthogonal flexures connected in series with offset $l$, the same script adds transformed compliances and prints

$$\boldsymbol{C}_{\text{ser}} = \begin{bmatrix} 0 & 0 & 0 & c_{\theta x} + c_{\theta y} & 0 & 0 \\ 0 & 0 & c_{\theta x} l & 0 & c_{\theta x} + c_{\theta y} & 0 \\ 0 & -c_{\theta z} l & 0 & 0 & 0 & 2c_{\theta z} \\ c_{\delta x} + c_{\delta y} & 0 & 0 & 0 & 0 & 0 \\ 0 & c_{\delta x} + c_{\delta y} + c_{\theta z} l^2 & 0 & 0 & 0 & -c_{\theta z} l \\ 0 & 0 & 2c_{\delta z} + c_{\theta x} l^2 & 0 & c_{\theta x} l & 0 \end{bmatrix}. \tag{6.18}$$

The off-diagonal terms are not errors. They are the algebraic record of the rigid offset between the two local flexure frames. This is why serial compliance must be transformed before it is added. Figure 6.4 shows mechanisms where these transformation and composition effects appear in practical flexure architectures.

## 6.6 Example: Two Spherical Notch Hinges in Series

Consider a serial chain consisting of two spherical notch hinges connected by an intermediate rigid body. Let each hinge have local compliance matrix $\boldsymbol{C}_s$, and let the transformations from each hinge frame to the chain output frame be $\mathrm{Ad}_1$ and $\mathrm{Ad}_2$. The serial-chain compliance is

$$\boldsymbol{C}_{\text{SS}} = \mathrm{Ad}_1 \, \boldsymbol{C}_s \, \mathrm{Ad}_1^{-1} + \mathrm{Ad}_2 \, \boldsymbol{C}_s \, \mathrm{Ad}_2^{-1} . \tag{6.19}$$

If three such SS chains are assembled in parallel between a stage and ground, with chain-level transformations $\mathrm{Ad}_{c1}$, $\mathrm{Ad}_{c2}$, and $\mathrm{Ad}_{c3}$, then each chain stiffness is

$$\boldsymbol{K}_{\text{SS},i} = \left( \mathrm{Ad}_{ci} \, \boldsymbol{C}_{\text{SS}} \, \mathrm{Ad}_{ci}^{-1} \right)^{-1}, \tag{6.20}$$

and the platform stiffness is

$$\boldsymbol{K}_{\text{platform}} = \boldsymbol{K}_{\text{SS},1} + \boldsymbol{K}_{\text{SS},2} + \boldsymbol{K}_{\text{SS},3}. \tag{6.21}$$

This example uses both composition rules. The SS limb is serial, so compliances add. The platform is parallel, so stiffnesses add. It is a compact example of the general mechanism workflow. Figure 6.5 shows the three-limb platform interpretation of this serial-then-parallel assembly.

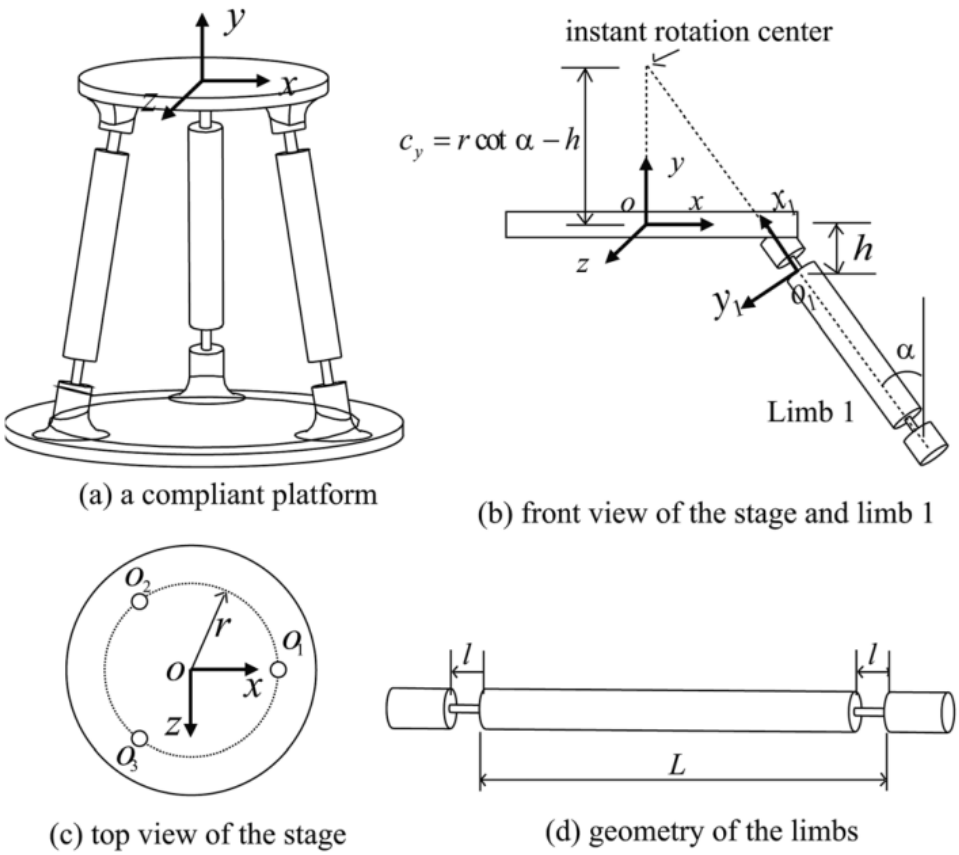



Figure 6.5: Three-limb platform example from [11]. Each limb is first reduced to a module matrix; the platform matrix is then obtained by transforming and adding the limb stiffnesses.

In the executable script, the serial matrix above is used as a limb compliance. Mathematica forms $\boldsymbol{K}_{\text{limb}} = \boldsymbol{C}_{\text{ser}}^{-1}$, rotates three copies by 0, $2\pi/3$, and $-2\pi/3$, and sums the transformed stiffness matrices. The full expression for $\boldsymbol{K}_{\text{platform}}$ is long because the off-axis limb locations introduce $h$, $r$, and $l$ couplings. Two representative entries printed by the script are

$$k_{36}^{(\text{platform})} = \frac{3(c_{\theta x} + c_{\theta y})}{2c_{\delta z}(c_{\theta x} + c_{\theta y}) + c_{\theta x}c_{\theta y}l^2} \tag{6.22}$$

and

$$k_{63}^{(\text{platform})} = \frac{3\left(c_{\delta x} + c_{\delta y} + c_{\theta z}(l^2 + 2lr + 2r^2)\right)}{c_{\theta z}(2c_{\delta x} + 2c_{\delta y} + c_{\theta z}l^2)}. \tag{6.23}$$

These entries show the expected coupling between rotation and translation in a spatial platform. The important computational lesson is that no new rule is needed: serial limb compliances are inverted to limb stiffnesses, transformed into the platform frame, and then added.

The compliance-assembly script in Appendix A generates the matrix results quoted in this chapter. The two compliance notebooks listed there provide the interactive workspaces.

## 6.7 Reading Design Information from the Matrix

Once $\boldsymbol{C}$ is known, design quantities can be read from its entries and ratios. Direct compliance in a desired motion direction appears on a diagonal term. Parasitic motion appears on off-diagonal terms. If a force $F_x$ causes both translation $\delta_x$ and rotation $\theta_y$, then the ratio

$$\frac{\theta_y/F_x}{\delta_x/F_x} = \frac{c_{21}}{c_{41}} \tag{6.24}$$

describes the rotational parasitic response per unit desired translation response. Depending on the geometry, a rotation center or instant center can be inferred from ratios of rotational and translational compliance terms. The notebooks contain examples labeled rotation-center error and compliance along $x$, which use exactly this kind of matrix interpretation.

Symbolic compliance analysis gives a practical example in which analytical compliance predictions are compared with finite element results and are accurate within a small error for linear deformation ranges [11]. The important lesson is not that symbolic matrices eliminate finite element analysis. The lesson is that symbolic matrices expose design dependencies. They show how a compliance term scales with length, thickness, angle, or material modulus, and they therefore support synthesis and sensitivity analysis.

## 6.8 Compliance Synthesis

Compliance synthesis asks for dimensions or layout parameters that produce a desired compliance behavior. In matrix form, the problem is to choose a parameter vector $\boldsymbol{p}$ so that selected entries or eigenvalues of $\boldsymbol{C}(\boldsymbol{p})$ meet design goals. For a linear spring along $x$, one might prescribe

$$c_{41}(\boldsymbol{p}) = c_x^* \tag{6.25}$$

while requiring parasitic compliances such as $c_{21}$, $c_{31}$, and $c_{51}$ to remain small. For a rotational pivot, one might prescribe a large $c_{33}$ while bounding translations under moment loading. Because symbolic formulas express matrix entries as functions of $\boldsymbol{p}$, they make these tradeoffs explicit.

The design process should remain sequential. First choose a topology whose ideal screw mobility is correct. Then derive or assemble $\boldsymbol{C}(\boldsymbol{p})$. Then tune dimensions to meet compliance and stress targets. Trying to synthesize compliance without first checking mobility can lead to designs whose numerical matrix looks acceptable in one loading case but whose topology admits an unintended motion.

## 6.9 Limits of Linear Compliance Models and the Design Loop

The formulas in this chapter assume small deformation, linear elasticity, and superposition. These assumptions are appropriate for many precision flexure mechanisms operating over small ranges. They are not appropriate for large-deflection compliant mechanisms, post-buckling devices, or mechanisms whose load paths change significantly with motion. In those cases, screw theory still helps at the conceptual level, but stiffness analysis must be updated with nonlinear beam theory, finite element analysis, or experimentally identified stiffness maps.

Even within the linear range, a compliance matrix is only as good as the primitive models used to build it. A blade flexure model that neglects shear deformation may be adequate for slender blades and poor for short thick blades. A notch hinge formula may depend on notch geometry and

stress concentration. A wire idealization may miss bending compliance that matters in a compact design. Good compliant mechanism design therefore uses a hierarchy of models: screw theory for topology, symbolic compliance for scaling and synthesis, finite element analysis for verification, and experiment for final validation.

The chapters now form a complete design loop. Screw theory defines freedom and constraint spaces. Mobility analysis computes the freedom space of a proposed architecture. Mobility synthesis computes the constraints needed for a desired freedom and maps them to flexure primitives. Stiffness analysis turns the ideal architecture into a quantitative model. Compliance decomposition then checks whether the quantitative model still behaves like the intended mobility design.

For graduate students, the most important habit is to keep the spaces and matrices connected. A twist basis without a stiffness matrix is a qualitative sketch. A compliance matrix without a twist-space interpretation is a table of numbers. A good compliant mechanism design needs both. The screw-theory-based approach is powerful because it lets the designer move between them without changing language.

# Appendix A

# Computational Companion Materials

The headless scripts are in `latex/mathematica` and the interactive notebooks are in `code`. Run `.wl` files with `wolframscript`; open `.nb` files in Mathematica.

- `screw_reciprocal_examples.wl`
  Reciprocal twist and wrench spaces for the wire-flexure examples.
- `mobility_analysis_examples.wl`
  Rank and reciprocal-space calculations for angled flexures and SS chains.
- `compliance_assembly_examples.wl`
  Transforms and assembles serial and parallel compliance and stiffness matrices.
- `compliance_mobility_criteria.wl`
  Scaled compliance eigenanalysis and the two effective-mobility criteria.
- `execute_source_notebooks.wl`
  Smoke-evaluates source notebooks where front-end-dependent content permits it.
- `flexure_screw_algebra.nb`
  Interactive screw-algebra, reciprocal-space, and mobility examples.
- `compliantmatrixexamples.nb`
  Interactive compliance models for flexures, SS chains, and parallel assemblies.
- `flexure_ComplianceMatrix_3.nb`
  Extended compliance-matrix examples, including parallelograms and an $xy$-stage.

# Bibliography


[1] R. S. Ball. *A Treatise on the Theory of Screws*. Cambridge University Press, 1900. Originally published in 1876.

[2] D. L. Blanding. *Exact Constraint: Machine Design Using Kinematic Principles*. ASME Press, New York, 1999.

[3] K. J. Waldron, G. L. Kinzel, and S. K. Agrawal. *Kinematics, Dynamics, and Design of Machinery*. 3rd edition, Wiley, 2016.

[4] S. Henein. *Flexures: Simply Subtle. Tutorial on the Design of Flexure-Mechanisms*. Second International Symposium on Compliant Mechanisms, IFToMM/ASME CoMe2011, Delft, 2011.

[5] J. B. Hopkins and M. L. Culpepper. Synthesis of multi-degree of freedom, parallel flexure system concepts via freedom and constraint topology (FACT). Part I: Principles. *Precision Engineering*, 34:259–270, 2010.

[6] J. B. Hopkins and M. L. Culpepper. Synthesis of multi-degree of freedom, parallel flexure system concepts via freedom and constraint topology (FACT). Part II: Practice. *Precision Engineering*, 34:271–278, 2010.

[7] J. B. Hopkins and M. L. Culpepper. Synthesis of precision serial flexure systems using freedom and constraint topologies (FACT). *Precision Engineering*, 35:638–649, 2011.

[8] H.-J. Su, D. V. Dorozhkin, and J. M. Vance. A screw theory approach for the type synthesis of compliant mechanisms with flexures. In *Proceedings of ASME IDETC/CIE 2009*, DETC2009-86684, San Diego, 2009.

[9] H.-J. Su and H. Tari. Realizing orthogonal motions with wire flexures connected in parallel. *Journal of Mechanical Design*, 2010.

[10] H.-J. Su. Mobility analysis of flexure mechanisms via screw algebra. *Journal of Mechanisms and Robotics*, 3:041010, 2011.

[11] H.-J. Su, H. Shi, and J. Yu. A symbolic formulation for analytical compliance analysis and synthesis of flexure mechanisms. *Journal of Mechanical Design*, 134, 2012. DOI: 10.1115/1.4006441.

[12] H.-J. Su and C. Yue. Type synthesis of freedom and constraint elements for design of flexure mechanisms. *Mechanical Sciences*, 4:263–277, 2013.

[13] Y. Zhang, H.-J. Su, and Q. Liao. Mobility criteria of compliant mechanisms based on decomposition of compliance matrices. *Mechanism and Machine Theory*, 79:80–93, 2014.

[14] C. Yue, Y. Zhang, H.-J. Su, and X. Kong. Type synthesis of three-degree-of-freedom translational compliant parallel mechanisms. *Journal of Mechanisms and Robotics*, 7:031012, 2015.

[15] H.-J. Su. Screw theory based design approach for compliant mechanisms, course application notes. https://haijunsu-osu.github.io/kinematics_ug/compliant_mechanisms_app/, accessed July 10, 2026.