Chapter 4

Type, Number, and Dimensional Synthesis

The science of mechanism kinematics is roughly divided in two divergent topics: analysis and synthesis. Analysis typically involves a defined mechanism and predicting how either the coupler or the follower will react to specific motions of the driver. Consider Figure 4.1 for example. It may be that we need to determine the position of reference point C on the coupler in the relatively non-moving frame Σ given an input angle ψ and all the link lengths, i.e. the distances between revolute centers, and position of point C in the coupler frame E. Alternately, it may be that we wish to determine the pose (position and orientation) of reference frame E that moves with the coupler in terms of fixed

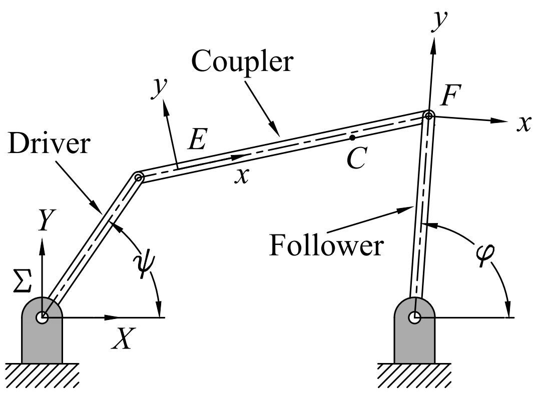
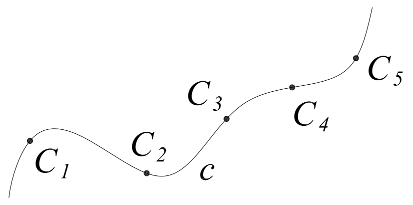


Figure 4.1: 4R four-bar mechanism.

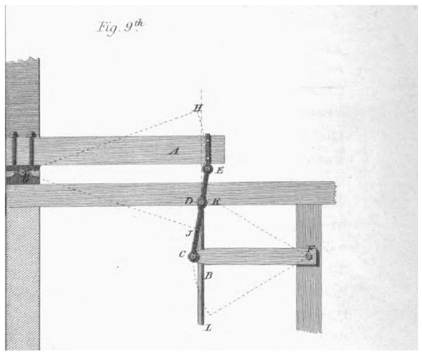
frame Σ for a given input ψ. Another problem we may need to solve is to determine the change in follower angle φ for a change in input angle ψ; or the kinematic equivalent, determine the motion of follower frame F expressed in Σ given the input ψ. We have examined analysis in Chapter 3.

A fundamentally different problem is that of kinematic synthesis. But, as we shall see, with some geometric insight, we can use the same approach as for analysis. By kinematic synthesis we mean the design or creation of a mechanism to attain specific motion characteristics. In this sense, synthesis is the inverse of analysis. Synthesis is the very essence of design because it represents the creation of new hardware to meet particular requirements of motion: displacement; velocity; acceleration; individually or in combination. Some typical examples of kinematic synthesis include:

1. Guiding a point along a specific curve, see Figure 4.2a. Examples of such devices include James Watt's straight line linkage illustrated in Figure 4.2b (taken from British Patent 1432, April 28, 1784), automated post-hole diggers, etc..



(a) Point C is guided along the curve c to sequential positions C_1 to C_5.



(b) Watt's straight line linkage.

Figure 4.2: Guiding a point on a rigid body along a curve.

2. Correlating driver and follower angles in a functional relationship: the resulting mechanism is a *function generator*. In essence the mechanism generates the function $\varphi = f(\psi)$, or vice versa, see Figure 4.3. The function generator we will examine is a four-bar steering linkage. The function is known as the *Akermann steering condition.* Ackermann steering geometry is a geometric arrangement of linkages in the steering of a wheeled vehicle designed to solve the problem of wheels on the inside and outside of a turn needing to trace out circles of different radii. The intention of

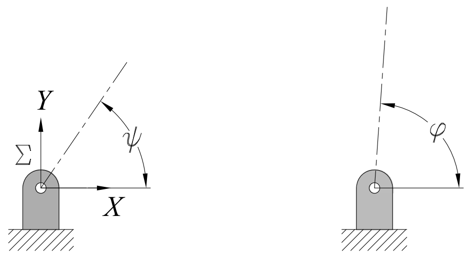


Figure 4.3: Function generator.

Ackermann geometry is to avoid the need for tires to slip sideways when following the path around a curve. It was invented by the German carriage builder Georg Lankensperger in Munich in 1817, then patented by his agent in England, Rudolph Ackermann (1764-1834) in 1818 for horse-drawn carriages [1]. The steering function creates the condition that the

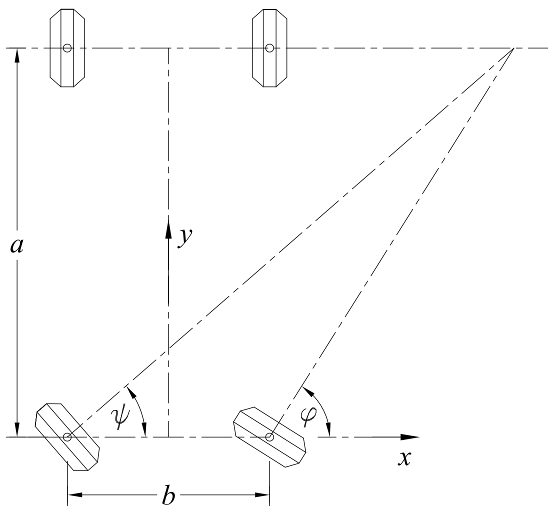


Figure 4.4: The steering condition.

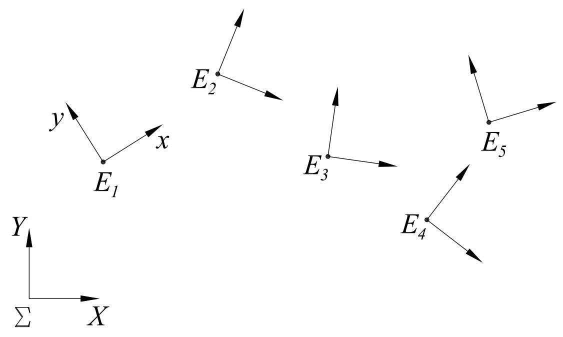
normals to the centre of each steerable wheel intersect the line passing through the centre of the common axle of the non-steerable wheels. The Akermann steering condition is defined by:

$$S(\Delta\psi, \Delta\varphi) \equiv \sin(\Delta\varphi - \Delta\psi) - \rho \sin(\Delta\psi)\sin(\Delta\varphi) \quad = \quad 0. \qquad (4.1)$$

In the steering condition $\Delta\psi$ and $\Delta\varphi$ are the change in input and output angles, respectively from the input and output *dial zeroes*, α and β, i.e.

$\psi = 0$ and $\varphi = 0$ are not necessarily in the direction of the x-axis, they may be offset. The quantity ρ is the length ratio b/a, with a the distance between axles and b the distance between wheel-carrier revolutes, as illustrated in Figure 4.4.

3. Guiding a rigid body through a finite set of positions and orientations, as in Figure 4.5a. A good example is a landing gear mechanism which must retract and extend the wheels, having down and up locked poses with specific intermediate poses for collision avoidance, see Figure 4.5b.



(a) Motion of coordinate system.

(b) Main landing gear.

Figure 4.5: Rigid body guidance (motion generation).

Synthesis for rigid body guidance is also known as the *Burmester Problem*, after Ludwig Burmester (1840-1927), Professor of Descriptive Geometry and Kinematics. His book *Lehrbuch der Kinematik* [2], is a classic systematic and comprehensive treatment of theoretical kinematics as understood in the late 19th century.

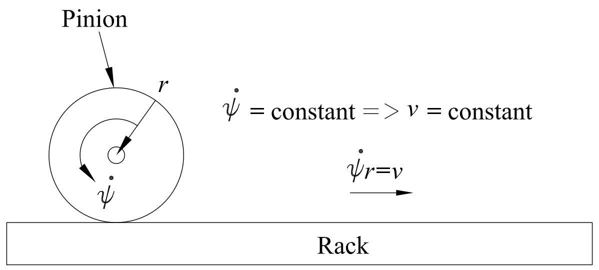


Figure 4.6: Rack-and-pinion drive.

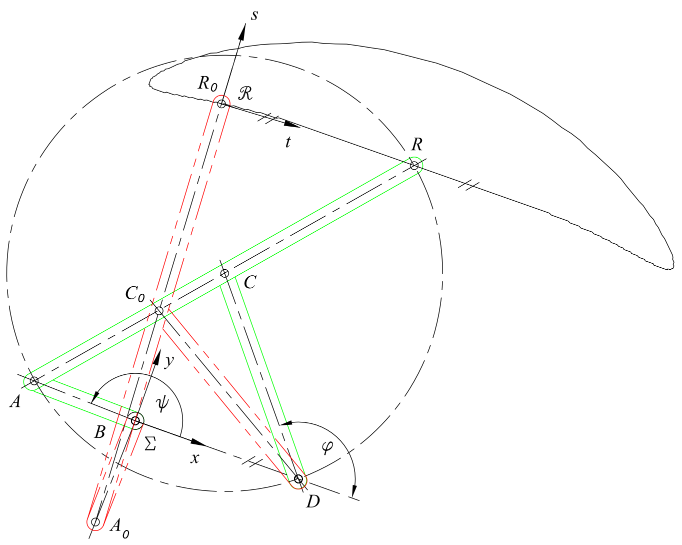


Figure 4.7: The line generating linkage where $\dot{\psi}$ is proportional to $\dot{R}$ along the straight line portion of the coupler curve for point R.

4. Trajectory Generation: Position, velocity, and/or acceleration must be correlated along a curve. Examples include film transport mechanisms in motion picture projectors. Another example is the four-bar linkage designed to emulate a rack-and-pinion drive, see Figures 4.6 and 4.7.

   If the input pinion angular velocity is constant then the linear velocity of the rack will also be constant along a straight line. The displacement of the rack is *timed* to that of the pinion via the meshing gear teeth. The linkage illustrated in Figure 4.7 produces approximately straight line motion along a section of its coupler curve. Moreover the position of the coupler point R is approximately timed to input angle ψ along the "straight" section, hence $d\psi/dt$ is linearly proportional to dR/dt on that section.

   Note that the coupler curve is endowed with central line symmetry (two halves are reflected in a line). The coupler point R generating this symmetric coupler curve is on a circle centered at C and passing though A. The central line symmetry of the coupler curve implies the coupler and follower have the same length: the distance between centers A and C is equal to the distance between C and D. This is also equal to the distance between C and coupler point R, while A, C, and R are collinear.

4.1 Background Concepts in Linear Algebra

The following material discusses the fundamental concepts in linear algebra that are needed to develop comprehensive mathematical and geometric models that can be used to solve the problems associated with mechanism design. As we will soon see, mechanism design typically involves establishing the best compromise among link lengths to achieve specified motion characteristics. This usually involves establishing the link lengths that minimise the deviation from an ideal value of a performance index. In other words, minimising the size of an error. In linkage design, this typically involves minimising the magnitude of a vector.

Vector norms serve the same purpose on vector spaces that the absolute value does on the real line. They furnish a measure of distance. More precisely, the vector space R^n together with a norm on R^n define a *metric space.* Since we will be comparing metric and non-metric geometries, we'd better have a definition of *metric space.*

Cartesian Product. The Cartesian product of any two sets S and T, denoted $S \times T$, is the set of all ordered pairs (s,t) such that $s \in S$ and $t \in T$.

Definition of a Metric. Let S be any set. A function d from $S \times S$ into $\mathbb{R}$, the set of real numbers

$$\mathbb{R} = d(S \times S) \equiv d_{s_i s_j}$$

is a metric on S if:

1. $d_{xy} \geq 0, \forall \, (x,y) \; \in \; S$;
2. $d_{xy} = d_{yx}, \forall \, (x,y) \; \in \; S$;
3. $d_{xy} = 0$ iff $x = y$;
4. $d_{xz} + d_{zy} \geq d_{xy}, \; \forall \, (x,y,z) \; \in \; S$.

Metric Space. A metric space is a non-empty set S, together with a metric d defined on S. For example, the Euclidean distance function for the orthogonal $x - y - z$ system:

$$d_{12} = d_{21} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

4.1.1 General Vector Spaces

Definition: Let V be an arbitrary non-empty set of objects on which two operations are defined: addition and multiplication by scalars (real numbers). If the following axioms are satisfied by all objects $\mathbf{u}, \mathbf{v}, \mathbf{w}$ in V and all scalars k and l then V is a *vector space* and the objects in V are vectors.

1. if $\mathbf{u}, \mathbf{v} \in V$, then $\mathbf{u} + \mathbf{v} \in V$.
2. $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$,
3. $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$.
4. $\exists\, \mathbf{0} \in V$ such that $\mathbf{0} + \mathbf{v} = \mathbf{v} + \mathbf{0} = \mathbf{v}$.
5. $\mathbf{u} + (-\mathbf{u}) = (-\mathbf{u}) + \mathbf{u} = \mathbf{0}$.
6. $k\mathbf{u} \in V$, $\forall\, k \in R$ and $\forall\, \mathbf{u} \in V$.
7. $k(\mathbf{u} + \mathbf{v}) = k\mathbf{u} + k\mathbf{v}$.
8. $(k + l)\mathbf{u} = k\mathbf{u} + l\mathbf{u}$.
9. $k(l\mathbf{u}) = (kl)\mathbf{u}$,
10. $1\mathbf{u} = \mathbf{u}$.

4.1.2 Vector Norms

A useful class of vector norms are the L_p-norms defined by:

$$\|\mathbf{x}\|_p = (|x_1|^p + ... + |x_n|^p)^{1/p}. \tag{4.2}$$

Of these the $p = 1$, $p = 2$, and $p = \infty$ norms are frequently used [3]:

$$\begin{aligned}
\|\mathbf{x}\|_1 &= |x_1| + ... + |x_n|; \\
\|\mathbf{x}\|_2 &= (|x_1|^2 + ... + |x_n|^2)^{1/2} \\
&= (\mathbf{x} \cdot \mathbf{x})^{1/2} \\
&= (\mathbf{x}^T \mathbf{x})^{1/2}; \\
\|\mathbf{x}\|_\infty &= \max_{1<i<n} |x_i|.
\end{aligned}$$

The Manhattan[1] and Chebyshev norms are the limiting cases, $p = 1$ and $p = \infty$, respectively, of the family of L_p-norms [4]. The $p = 2$ norm, or 2-norm is otherwise known as the Euclidean norm, which is the generalized Euclidean distance and is geometrically very intuitive. The L_p-norms obey the following relationship:

$$\|\mathbf{x}\|_\infty \leq \cdots \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1.$$

[1]The term Mahattan norm arises because this vector norm corresponds to sums of distances along the basis vector directions, as one would travel along a rectangular street plan.

Typically, the most appropriate norm must be selected to evaluate the magnitude of the objective function for the error minimisation, given a motion objective that is to be approximated by the resulting linkage. However, it turns out that Lawson's algorithm [5, 6] can be used to sequentially minimise the Chebyshev norm via the minimisation of the Euclidean norm [7]. This means that the continuous approximate approach to the design error minimisation is independent of the L_p-norm because it applies to both the Chebyshev and Euclidean norms, and hence all intermediate ones. Therefore, without loss in generality the Euclidean norm will be used in the development of the material developed in the following sections.

Example: Consider the vector in a four dimensional vector space:

$$\begin{aligned}
\mathbf{x} &= \begin{bmatrix} 2 \\ 1 \\ -2 \\ 4 \end{bmatrix}, \\
\|\mathbf{x}\|_1 &= 2+1+2+4 \;=\; 9, \\
\|\mathbf{x}\|_2 &= (4+1+4+16)^{1/2} \;=\; 5, \\
\|\mathbf{x}\|_\infty &= 4.
\end{aligned}$$

It is clear that $\|\mathbf{x}\|_\infty \le \cdots \le \|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1$, as required.

4.1.3 Question of Existence

There is an old joke that goes: the way to tell the difference between a mathematician and an engineer is to give them a problem. The engineer will immediately proceed to seek a solution; the mathematician will first try to prove that a solution exists. Engineers laugh at mathematicians and vice versa when the joke is told. The history of mathematics contains many stories of wasted effort attempting to solve problems that were ultimately proved to be unsolvable. The question of *existence* should not be ignored.

The existence of a solution must be examined for linear systems before we proceed to attempt to solve. We must ask two questions:

1. Is the system consistent; does at least one solution exist?
2. If one solution exists, is it unique?

We can always represent a set of m linear equations in n unknowns as:

$$\mathbf{A}_{m\times n}\mathbf{x}_{n\times 1} \;=\; \mathbf{b}_{m\times 1}, \tag{4.3}$$

where $\mathbf{A}$ is a matrix with m rows and n columns, $\mathbf{x}$ is an $n \times 1$ column vector, and $\mathbf{b}$ is an $m \times 1$ column vector. In general, we can make the following three claims.

1. The system is *under-determined*: there are fewer equations than unknowns $(m < n)$. If one solution exists then an infinite number of solutions exist.

2. The system is *determined*: the number of equations equals the number of unknowns $(m = n)$. Then if a solution exists it is unique.

3. The system is *over-determined*: there are more equations than unknowns $(m > n)$. Then, in general, no solutions exist.

4.1.4 Vector, Array, and Matrix Operations

While Chinese mathematicians had developed algorithms for solving linear systems by 250 BC, linear algebra as we know it was first formalized by German mathematician and Sanskrit scholar Hermann Grassmann (1808-77). The new algebra of vectors was presented in his book *Die Ausdehnungslehre* [8] first published in 1844.

Linear Dependence

Consider three equations linear in three unknowns of the form:

$$\mathbf{Ax} = \mathbf{b}.$$

These three linear equations can always be thought of as three planes. One way to test if the three equations are linearly independent (no line common to all three planes) is to look at the coefficient matrix $\mathbf{A}$. The system is linearly dependent if at least two rows (or two columns) are scalar multiples of each other, consider the following:

$$\begin{bmatrix} 5 & -1 & 2 \\ -2 & 6 & 9 \\ -10 & 2 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 18 \end{bmatrix}.$$

The first and third rows of $\mathbf{A}$ are scalar multiples of each other which implies a linear dependence, in turn meaning that there are no real finite values of x, y, and z that satisfy the equations. This is due to the fact that the determinant of Matrix $\mathbf{A}$ is identically equal to zero meaning that it is not invertible.

Solutions to Linear Systems

In 3D Euclidean space, every system of linear equations has either: no solution (the system is inconsistent); exactly one solution (the system is consistent); or infinitely many solutions (the system is consistent). For now we shall concentrate on determined systems: three equations linear in three unknowns. Geometrically, one linear equation in three unknowns (x, y, z) represents a *plane* in the *space* of the unknowns.

Taking the three equations to represent three planes in Euclidean 3D space, solutions to the system are bounded by Euclid's *parallel axiom*: Given a line L and a point P, not on L, there is one and only one line through P that is parallel to L (and two parallel lines do not intersect). Extending this axiom to planes, three planes either:

1. have no point in common;
2. have exactly one point in common;
3. have infinitely many points in common (the planes have a line in common, or are all incident)

If we extend 3-D Euclidean space to include all points at infinity, things change. Now every parallel line intersects in a point on a line at infinity, and every parallel plane intersects in a line on the plane at infinity. In this sense, there are five possibilities for a system of three equations linear in three unknowns:

1. unique finite solution;
2. infinite finite solution;
3. double infinity of finite solutions;
4. infinite solutions at infinity;
5. unique solution at infinity, occurs in two ways.

1: Unique Finite Solution. Consider the linear system:

$$\begin{aligned} 5x - y + 2z &= 7, \\ -2x + 6y + 9z &= 0, \\ -7x + 5y - 3z &= -7. \end{aligned}$$

This system can be represented ($\mathbf{Ax} = \mathbf{b}$) as:

$$\begin{bmatrix} 5 & -1 & 2 \\ -2 & 6 & 9 \\ -7 & 5 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ -7 \end{bmatrix}.$$

The solution can easily be obtained using the MATLAB "\" operator:

$$\mathbf{x} = \mathbf{A} \setminus \mathbf{b},$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = (1/13) \begin{bmatrix} 21 \\ 10 \\ -2 \end{bmatrix}.$$

There is only one possible solution, the three planes intersect at a single common finite point, see Figure 4.8. While the \ operator implies the division of a matrix by a vector, which is an undefined operation in a vector space, it actually relies on an operation that uses a matrix orthogonalisation technique employing Householder reflections.

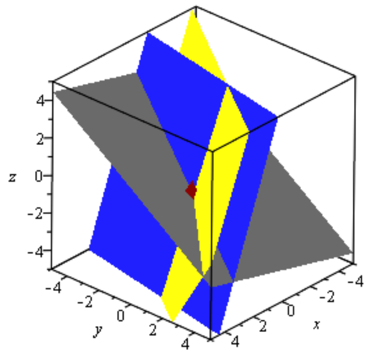


Figure 4.8: One real, finite unique solution.

2: Infinite Solutions Along a Finite Line. Consider the linear system:

$$\begin{aligned} 5x - y + 2z &= 7, \\ -2x + 6y + 9z &= 0, \\ 6x + 10y + 22z &= 14. \end{aligned}$$

This system can be represented ($\mathbf{Ax} = \mathbf{b}$) as:

$$\begin{bmatrix} 5 & -1 & 2 \\ -2 & 6 & 9 \\ 6 & 10 & 22 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 14 \end{bmatrix}.$$

Solution:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = (1/4) \begin{bmatrix} 6-3t \\ 2-7t \\ 4t \end{bmatrix}.$$

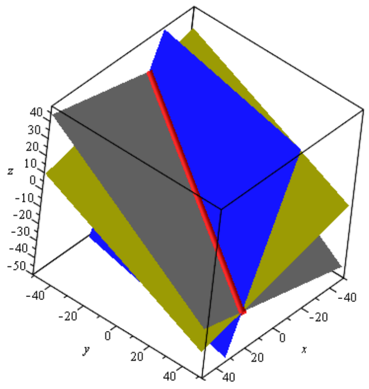


Figure 4.9: Infinitely many real solutions: a one parameter family of points.

The three planes all possess the same linear dependency: they all intersect in the same line, see Figure 4.9. There are an infinite number of solutions along the line of intersection between the three intersecting plane. However, numerical methods to obtain the solution, such as the MATLAB \ operator, cannot be used, symbolic algebra is required because the equations are *singular* in the sense that they are linearly dependent. While the solution family is the one parameter set of points laying on a finite straight line, there is additionally the point at infinity of the finite line when the line is expressed using homogeneous coordinates.

3: Double Infinity of Solutions on a Finite Plane. Consider the system:

$$\begin{aligned} 5x - y + 2z &= 7, \\ 10x - 2y + 4z &= 14, \\ -15x + 3y - 6z &= -21. \end{aligned}$$

This system can be represented ($\mathbf{Ax} = \mathbf{b}$) as:

$$\begin{bmatrix} 5 & -1 & 2 \\ 10 & -2 & 4 \\ -15 & 3 & -6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 14 \\ -21 \end{bmatrix}.$$

Solution:

$$5x - y + 2z = 7$$

All three planes are superimposed as in Figure 4.10. There will be a

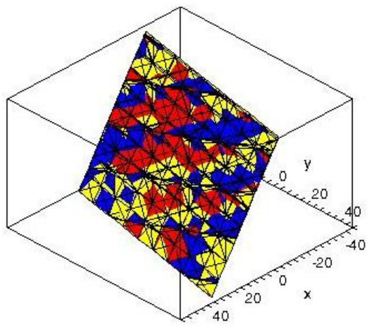


Figure 4.10: ∞^2 real solutions.

double infinty of solutions: a two parameter family of lines covering the entire plane. Additionally, if the system were expressed using homogeneous coordinates, there is a one parameter family of solutions on the line at infinity corresponding to the intersection of the plane at infinity with the three superimposed planes. Despite the fact that a double infinity of real, finite solutions exist, using the MATLAB \ operator yields the MATLAB error message: *matrix is singular to working precision.* Do you hear the polite chuckles of the mathematicians?

4: Infinite Solutions at Infinity. Consider the linear system:

$$\begin{aligned} 5x - y + 2z &= 7, \\ -10x + 2y + -4z &= -5, \\ 15x - 3y + 6z &= -5. \end{aligned}$$

This system can be represented ($\mathbf{Ax} = \mathbf{b}$) as:

$$\begin{bmatrix} 5 & -1 & 2 \\ -10 & 6 & -4 \\ 15 & -3 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ -5 \\ -5 \end{bmatrix}.$$

Solution:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \infty.$$

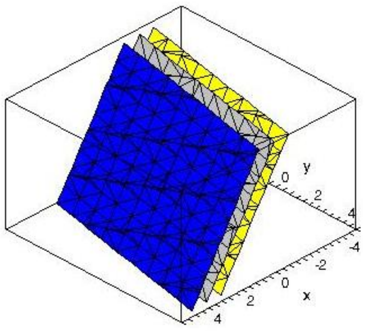


Figure 4.11: One solution where the line at infinity of two of the planes intersects the third plane.

The three equations represent three parallel non-coincident planes, as in Figure 4.11. When represented using homogeneous coordinates, the three planes all intersect the plane at infinity in the same real, but non finite line: a unique line on the plane at infinity.

5: No Finite, But One Infinite Solution. Consider the linear system:

$$\begin{aligned} 5x - y + 2z &= 7, \\ -2x + 6y + 9z &= 0, \\ -10x + 2y - 4z &= 18. \end{aligned}$$

This system can be represented ($\mathbf{Ax} = \mathbf{b}$) as:

$$\begin{bmatrix} 5 & -1 & 2 \\ -2 & 6 & 9 \\ -10 & 2 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 0 \\ 18 \end{bmatrix}.$$

Solution:

$$\begin{aligned} 5x - y + 2z &= 7, \\ 5x - y + 2z &= -9. \end{aligned}$$

In this system, there are two parallel (linearly dependent, but not coincident) planes, intersecting a third plane as in Figure 4.12. There will never be a finite point/line/plane where the system of equations will intersect. But, it can be seen that the two parallel planes intersect the third in two distinct parallel lines. These two parallel lines intersect in a unique point at infinity.

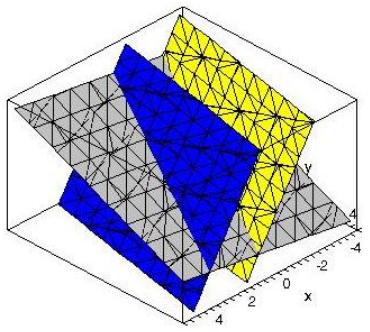


Figure 4.12: Lines of intersection of each pair of planes are parallel, meaning one unique solution at infinity. plane.

6: No Finite, But One Infinite Solution. Consider the linear system:

$$\begin{aligned} 5x - y + 2z &= -12, \\ -2x + 6y + 9z &= 0, \\ 8y + 14z &= 8. \end{aligned}$$

This system can be represented ($\mathbf{Ax} = \mathbf{b}$) as:

$$\begin{bmatrix} 5 & -1 & 2 \\ -2 & 6 & 9 \\ 0 & 8 & 14 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -12 \\ 0 \\ 8 \end{bmatrix},$$

Solution: three parallel lines. Each pair of the three planes intersect in a distinct line, as shown in Figure 4.13. The line of intersection of each pair of planes is parallel to the third. The parallel lines intersect in a single real point at infinity.

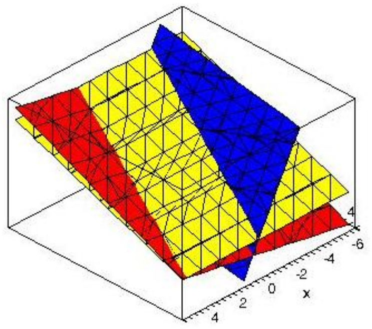


Figure 4.13: No Finite Solution: three planes intersecting in three non-intersecting lines.

4.1.5 Linear Least-Squares Problem

In a system of linear equations $\mathbf{A}_{m\times n}\mathbf{x} = \mathbf{b}$, when there are more equations than unknowns, $m > n$, and the set is said to be *over-determined*, there is, in general, no solution vector $\mathbf{x}$. The next best thing to an *exact* solution is the best *approximate* solution. This is the solution that comes closest to satisfying all the equations simultaneously. If *closeness* is defined in the least-squares sense, then the sum of the squares of the differences between the left- and right-hand sides of $\mathbf{A}_{m\times n}\mathbf{x} = \mathbf{b}$ must be minimised. Then the over-determined linear problem becomes a (usually) solvable linear problem: the linear least-squares problem. This concept will be employed extensively starting in Section 4.4.1.

Generally, we start with an optimality condition called the *normality condition*. Conceptually, we can transform the $m \times n$ system to an $n \times n$ system by pre-multiplying both sides by $\mathbf{A}^T$:

$$(\mathbf{A}^T_{n\times m}\mathbf{A}_{m\times n})\mathbf{x}_{n\times 1} = \mathbf{A}^T_{n\times m}\mathbf{b}_{m\times 1}.$$

These are typically referred to as the *normal equations* of the linear least-squares problem. Direct solution of the over-determined system of equations relies on the *Moore-Penrose generalised inverse* [4] (M-PGI) of the rectangular matrix $\mathbf{A}$, which yields:

$$\mathbf{x}_{n\times 1} \quad \approx \quad (\mathbf{A}^T_{n\times m}\mathbf{A}_{m\times n})^{-1}\mathbf{A}^T_{n\times m}\mathbf{b}_{m\times 1}.$$

The quantity $(\mathbf{A}^T_{n\times m}\mathbf{A}_{m\times n})^{-1}\mathbf{A}^T_{n\times m}$ is the M-PGI.

It turns out that singular value decomposition can solve the normal equations, using householder reflections, without explicitly evaluating them, see Sections 4.1.8 and 4.1.9. It is always worth repeating that direct solution of the normal equations is usually the worst way to find least-squares solutions because of numerical conditioning, see Section 4.1.7, and indirect methods of approximating the best compromise solution relative to some index should be employed, such as Householder reflections, or singular value decomposition. Using the M-PGI to approximate a solution is termed the *naïve* approach.

In summary, given an objective function $z = (1/2)\mathbf{e}^T\mathbf{e}$ to minimise over $\mathbf{x}$, in an over-determined set of equations, we have

The Normality Condition:

$$\frac{\partial z}{\partial \mathbf{x}} = \mathbf{0};$$

The Normal Equations:

$$(\mathbf{A}^T\mathbf{A})\mathbf{x} - \mathbf{A}^T\mathbf{b} = \mathbf{0};$$

The Naïve Solution:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}.$$

In what follows we discuss why using the M-PGI is considered naïve and what can be used in its place.

4.1.6 Nullspace, Range, Rank:

The nullspace, range, rank, and nullity are measures of capability of linear mappings in the form of vector-matrix equations that will help us avoid using the naïve approach to solving the system of equations outlined above to solve systems of linear equations. Consider the following linear system as a mapping

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

Matrix $\mathbf{A}$ may be viewed as a map from the vector space $\mathbf{x}$ to the vector space $\mathbf{b}$.

Nullspace: The nullspace of $\mathbf{A}$ is a subspace of $\mathbf{x}$ that $\mathbf{A}$ maps to $\mathbf{0}$. It is the set of all linearly independent $\mathbf{x}$ that satisfy

$$\mathbf{A}\mathbf{x} = \mathbf{0}.$$

The dimension of the nullspace is called the *nullity of* $\mathbf{A}$.

Range: The range of $\mathbf{A}$ is the subspace of $\mathbf{b}$ that can be reached by $\mathbf{A}$ for the $\mathbf{x}$ *not* in the nullspace of $\mathbf{A}$

Rank: The dimension of the range is called the *rank* of $\mathbf{A}$. For an $n \times n$ square matrix, $rank + nulity = n$. For a non-singular $n \times n$ square matrix $rank(\mathbf{A}) = n$.

The maximum *effective rank* of a rectangular $m \times n$ matrix is the rank of the largest square sub-matrix. For $m > n$, full rank means $rank(\mathbf{A}) = n$ and rank deficiency means $rank < n$

4.1.7 Numerical Conditioning

The condition number of a matrix is a measure of how *invertible* or how *close* to singular a matrix is. The condition number of a matrix is the ratio of the largest to the smallest singular values.

$$k(\mathbf{A}) = \frac{\sigma_{\max}}{\sigma_{\min}}.$$

A matrix is singular if $k = \infty$, and is ill-conditioned if k is very large. But, how big is large? Relative to what? The condition number is lower bounded by a finite number, but the upper bound is infinite

$$1 \leq k(\mathbf{A}) \leq \infty.$$

Instead, let's consider the reciprocal of $k(\mathbf{A})$. Let this reciprocal be $\lambda(\mathbf{A})$ such that

$$\lambda(\mathbf{A}) = 1/k(\mathbf{A}).$$

This ratio is bounded both from above and below

$$0 \leq \lambda(\mathbf{A}) \leq 1.$$

Now we can say $\mathbf{A}$ is well-conditioned if $\lambda(\mathbf{A}) \simeq 1$, and ill-conditioned if $\lambda(\mathbf{A}) \simeq 0$. But, now how small is *too close* to zero. Well, this we can state with certainty. A numerical value for *too small* is the computer's floating-point precision which is typically 10^{-12} for double precision computations. MATLAB uses double floating-point precision for all computations. Hence, if $\lambda(\mathbf{A}) < 10^{-12}$, $\mathbf{A}$ is ill-conditioned.

4.1.8 Householder Reflections

We will generally want to avoid using the naïve approach to solving the system of equations outlined in Section 4.1.5, and adopt an approach that is insensitive to the conditioning of the equations. Consider the matrix equation

$$\mathbf{A}_{m\times n}\mathbf{x}_{n\times 1} = \mathbf{b}_{m\times 1}.$$

The least-squares solution to a suitably large over-determined system involves the *pseudo*, or *Moore-Penrose generalized inverse* of $\mathbf{A}$. Given $\mathbf{A}$ and $\mathbf{b}$, we can approximate $\mathbf{x}$ in a least-squares sense as:

$$\mathbf{x} \quad \approx \quad (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{4.4}$$

Practically, however, *if* $\mathbf{A}$ *has full rank*, then Householder reflections [9] (HR), also known as Householder transformations, are employed to transform $\mathbf{A}$ and $\mathbf{b}$. They represent a change of basis that do not change the relative angles between the basis column vectors of $\mathbf{A}$. An HR is a linear transformation that describes a reflection about a plane or hyperplane containing the origin. Hence, they are proper isometries that do not distort the metric on either $\mathbf{A}$ or $\mathbf{b}$. The i^{th} HR is defined such that

$$\mathbf{H}_i^T\mathbf{H}_i = I, \text{ and } \det(\mathbf{H}_i) = -1.$$

Thus, $\mathbf{H}_i$ is orthogonal, but not necessarily symmetric. The dimensions of the system of original equations are illustrated in Figure 4.14. For the dimensions

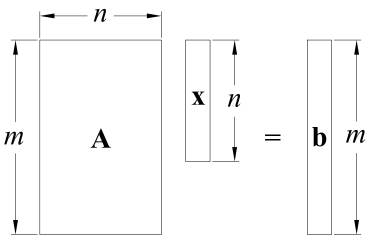


Figure 4.14: Dimensions of the system of linear equations.

of an $m \times n$ matrix $\mathbf{A}$ and an $m \times 1$ vector $\mathbf{b}$, n HR's are required to transform $\mathbf{A}$ and $\mathbf{b}$. Each HR is an $m \times m$ matrix that transforms the system into a set of n *canonical* equations, as illustrated in Figure 4.15, which allows the unknown elements of $\mathbf{x}$ to be solved for using back-substitution. It is important to re-

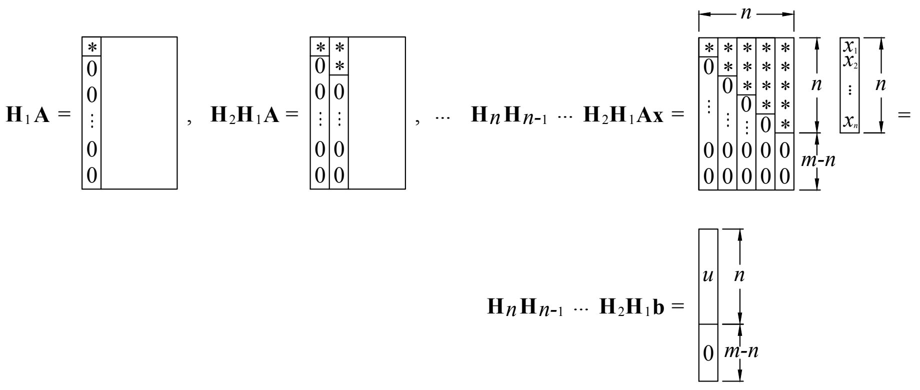


Figure 4.15: Visualisation of how the Householder reflections transform the $m \times n$ equations into a set of n canonical equations.

member that when coding an algorithmic based piece of software for designing optimised linkages that it is typically not a good choice to blindly use the pseudo inverse to approximate $\mathbf{x}$, but Householder reflections instead when $\mathbf{A}$ has full rank. What if $\mathbf{A}$ does not possess full rank, or worse, possesses full rank but with a very large condition number? This is where SVD can be used. It is *much* more than the flavor-of-the-month least-squares numerical approximation algorithm. It is a very powerful technique for dealing with sets of equations that are either singular (contain linear dependencies), numerically *close* to singular, or even fully determined and well constrained.

4.1.9 Singular Value Decomposition (SVD)

Singular value decomposition (SVD) is a powerful technique for dealing with sets of equations that are either singular, or numerically close to singular. In many cases where Gaussian elimination, Householder reflections, or the M-PGI fail to yield useful results, SVD will diagnose exactly what the cause of problem is, and determine a useful numerical solution. This is a result of the fact that SVD explicitly constructs orthonormal bases for both the nullspace and range of every matrix $\mathbf{A}$.

SVD is based on the fact that every $m \times n$ matrix $\mathbf{A}$, square or rectangular of any dimension, can be decomposed into the product of the following matrix *factors*:

$$\mathbf{A}_{m\times n} = \begin{cases} \mathbf{U}_{m\times n}\mathbf{\Sigma}_{n\times n}\mathbf{V}^T_{n\times n} & \text{called the } \textit{economy size} \text{ version[10],} \\ \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}\mathbf{V}^T_{n\times n} & \text{called the } \textit{full size} \text{ version,} \end{cases}$$

where for the economy size version, which we will use:

- $\mathbf{U}_{m\times n}$ is column-orthonormal, the column vectors are all mutually orthogonal unit vectors. In other words, using MATLAB syntax

$$\mathbf{U}(:,j)\cdot\mathbf{U}(:,k) = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}$$

- $\mathbf{\Sigma}_{n\times n}$ is a diagonal matrix whose diagonal elements are the *singular values*, σ_i of $\mathbf{A}$. The singular values of $\mathbf{A}$ are the square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$. They are positive numbers arranged in descending order lower bounded by zero. The eigenvalues are obtained by solving the characteristic equation of $\mathbf{A}$ which is obtained from

$$(\mathbf{A}^T\mathbf{A})\mathbf{x} = \lambda\mathbf{x} \Rightarrow (\mathbf{A}^T\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}.$$

The corresponding characteristic equation is

$$\mathbf{A}^T\mathbf{A} - \lambda\mathbf{I} = \mathbf{0}.$$

Solving this system of equations for all values of λ reveals the singular values of $\mathbf{A}$:

$$\sigma_i = \sqrt{\lambda_i}.$$

- $\mathbf{V}$ is a square orthogonal matrix, i.e. both it's columns and rows are orthonormal, so

$$\mathbf{V}\mathbf{V}^T = \mathbf{I}.$$

These important properties mean that the inverses of **U**, **Σ**, and **V** are trivial to compute. Because **U** and **V** are orthogonal their inverses are equal to their transposes. Because **Σ** is diagonal, it's inverse is another diagonal matrix whose elements are the reciprocals of the elements σ_i. The only thing that can go wrong is for one of the σ_i to be zero, or for it to be so small that it's numerical value is dominated by roundoff error and therefore unknowable.

Recall that for an $m \times n$ matrix **A**, where $m > n$, to possess full rank means that $rank = n$. In this case, all $\sigma_i, i = 1...n$, are positive, non-zero real numbers. The matrix is (analytically) non-singular. If **A** is rank deficient then some of the σ_i are identically zero. The number of $\sigma_i = 0$ corresponds to the dimension of the nullspace ($n - rank(\mathbf{A})$). The σ_i are arranged in **Σ** on the diagonal in descending order: $\sigma_1 \geq \sigma_2,\ \sigma_2 \geq \sigma - 3,\ ...\ , \sigma_{n-1} \geq \sigma_n$

- The columns of **U** whose same-numbered elements σ_i are non-zero are an orthogonal set of orthonormal basis vectors that span the range of A: every possible product $\mathbf{Ax} = \mathbf{b}$ is a linear combination of these column vectors of **U**.

- The columns of **V** whose same numbered elements $\sigma_i = 0$ form an orthogonal set of orthonormal basis vectors that span the nullspace of **A**: every possible product $\mathbf{Ax} = \mathbf{0}$ is a linear combination of these column vectors of **V**, see Figure 4.16.

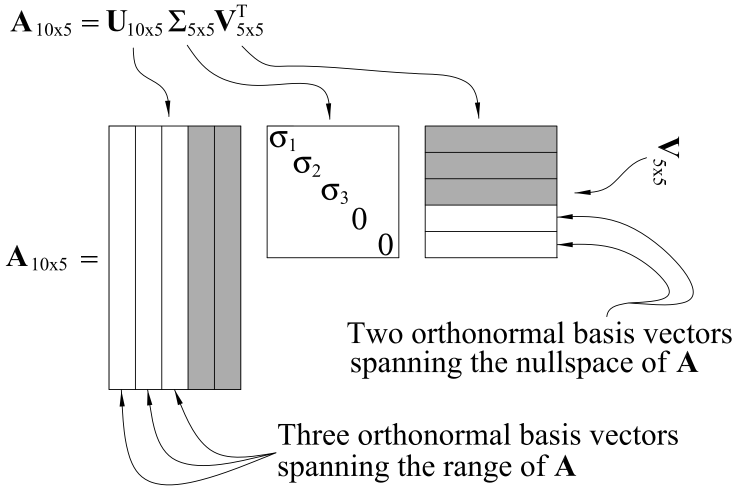

$\mathbf{A}_{10x5} = \mathbf{U}_{10x5}\mathbf{\Sigma}_{5x5}\mathbf{V}^T_{5x5}$
$\mathbf{A}_{10x5} =$
σ_1
σ_2
σ_3
0
0
$\mathbf{V}_{5x5}$
Two orthonormal basis vectors spanning the nullspace of **A**
Three orthonormal basis vectors spanning the range of **A**

Figure 4.16: Visualisation of the *economy size* singular value decomposition.

MATLAB Note: To get the form of SVD in Figure 4.16, the syntax is

$$[\mathrm{U}, \mathrm{S}, \mathrm{V}] = \mathrm{svd}(\mathrm{A}, 0).$$

For any matrix $\mathbf{A}_{m\times n}$, the MATLAB command svd(A,0) returns the three matrices $\mathbf{U}_{m\times n}$, $\mathbf{S}_{n\times n}$ (the singular values), and $\mathbf{V}_{n\times n}$ such that $\mathbf{A} = \mathbf{USV}^T$. Note that the matrix $\mathbf{S}$ is equivalent to the matrix $\mathbf{\Sigma}$ described earlier.

What computational advantage does this provide? Every matrix, no matter how singular, can be inverted! By virtue of the properties of $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$, the inverse of any matrix $\mathbf{A}$, square or not, is simply

$$\mathbf{A}^{-1} = (\mathbf{U}_{m\times n}\mathbf{\Sigma}_{n\times n}\mathbf{V}^T_{n\times n})^{-1} = \mathbf{V}_{n\times n}\mathbf{\Sigma}^{-1}_{n\times n}\mathbf{U}^T_{n\times m}.$$

Because $\mathbf{V}$ and $\mathbf{U}$ are orthogonal their inverses are equal to their transposes. Because $\mathbf{\Sigma}$ is diagonal its inverse, $\mathbf{\Sigma}^{-1}$, has elements $1/\sigma_i$ on the diagonal and zeroes everywhere else. So we can write explicitly:

$$\mathbf{A}^{-1}_{n\times m} = \mathbf{V}_{n\times n}[\mathrm{diag}(1/\sigma_i)]\mathbf{U}^T_{n\times m}.$$

Computational problems can only be caused if any $\sigma_i = 0$, or are so close to zero that the value of σ_i is dominated by roundoff error so that $1/\sigma_i$ becomes a very large, but equally inaccurate number, which pushes A^{-1} towards infinity in the wrong direction. That doesn't bode well for the solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

4.1.10 What to do When A is Singular

When matrix $\mathbf{A}$ is non-singular and square, then it maps one vector space onto one of the same dimension. For the $m \times n$ case $\mathbf{x}$ is mapped onto $\mathbf{b}$ so that $\mathbf{Ax} = \mathbf{b}$ is satisfied but the dimensions of $\mathbf{x}$ and $\mathbf{b}$ are different, see Figure 4.17. However, if $\mathbf{A}$ is singular we can still find a useful solution. A more detailed discussion of the material in this section may be found in the book *Numerical Recipes in C* [10]. There are three cases to consider if $k(\mathbf{A}) = \infty$.

1. If $\mathbf{b} = \mathbf{0}$, the solution for $\mathbf{x}$ is any column of $\mathbf{V}$ whose corresponding $\sigma_i = 0$. This is so because for a non-trivial solution to exist for the system $\mathbf{Ax} = \mathbf{0}$ then at least one $\sigma_i = 0$ because the determinant of $\mathbf{A}$ must be 0.

2. If $\mathbf{b} \neq 0$ the important question is if there exist any $\mathbf{x}$ that can be reached by $\mathbf{A}$, in other words, is $\mathbf{b}$ in the range of $\mathbf{A}$? If $\mathbf{b}$ is in the range of $\mathbf{A}$, but $\mathbf{A}$ is singular then the set of equations does have a solution $\mathbf{x}$, in fact it has infinitely many. Any vector in the nullspace, i.e., any column of $\mathbf{V}$

with a corresponding zero σ_i, can be added to $\mathbf{x}$ in any linear combination. This is graphically illustrated in Figure 4.18.

But this is where things get really interesting. The $\mathbf{x}$ with the smallest Euclidean norm can be obtained by setting $\infty = 0$!! This, however, is not the desperation mathematic is appears to be at first glance. If $\sigma_i = 0$, set $(1/\sigma_i) = 1/0 = 0$ in $\mathbf{\Sigma}^{-1}$. Then, the $\mathbf{x}$ with the smallest 2-norm is

$$\mathbf{x} \quad = \quad \mathbf{V}[\mathrm{diag}(1/\sigma_i)](\mathbf{U}^T\mathbf{b}),\ i = 1,\ ...,\ n. \tag{4.5}$$

Where for all $\sigma_i = 0$, $1/\sigma_i$ is set to zero. (Note, in Figure 4.18 the equation $\mathbf{Ax} = \mathbf{d}$ is used for this case.)

3. If $\mathbf{b} \neq 0$ but is *not* in the range of $\mathbf{A}$ (this is typical in over-determined systems of equations). In this case the set of equations are inconsistent, and no solution exists that simultaneously satisfies them. We can still use Equation (4.5) to *construct* a solution vector $\mathbf{x}$ that is the closest approximation in a least-squares sense. In other words, SVD finds the $\mathbf{x}$ that minimises the norm of the residual r of the solution, where

$$r \equiv ||\mathbf{Ax} - \mathbf{b}||_2.$$

Referring to Equation (4.11), note the happy similarity to the design error!! This is illustrated in Figure 4.18.

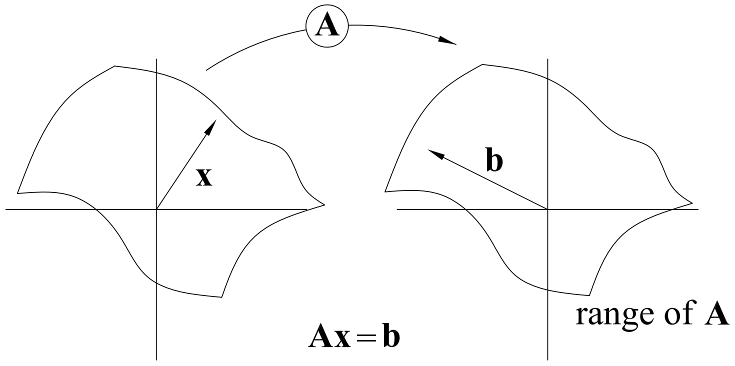


Figure 4.17: Matrix $\mathbf{A}$ is non-singular and maps one vector space onto one of the same dimension if $\mathbf{A}$ is square. For the $m \times n$ case $\mathbf{x}$ is mapped onto $\mathbf{b}$ so that $\mathbf{Ax} = \mathbf{b}$ is satisfied but the dimensions of $\mathbf{x}$ and $\mathbf{b}$ are different.

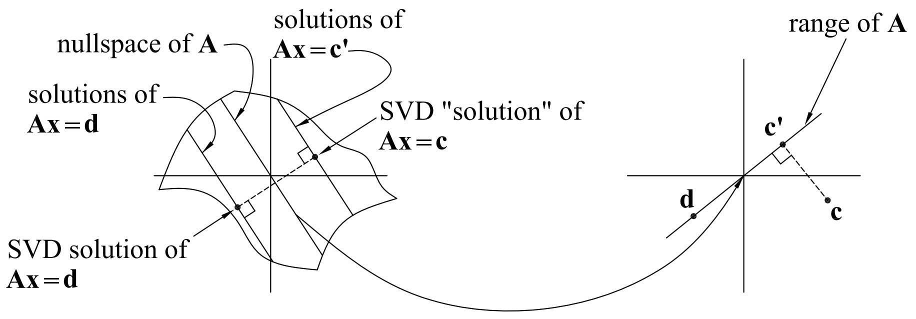

nullspace of **A**
solutions of **Ax** = **c'**
range of **A**
solutions of **Ax** = **d**
SVD "solution" of **Ax** = **c**
c'
d
c
SVD solution of **Ax** = **d**

Figure 4.18: Matrix **A** is singular. If **A** is square it maps a vector space of lower dimension (in the figure **A** maps a plane onto a line, the range of **A**). The nullspace of **A** maps to zero. The "solution" vector **x** of **Ax** = **d** consist of a particular solution plus any vector in the nullspace. In this example, these solutions form a line parallel to the nullspace. SVD finds the **x** closest to zero by setting $1/\sigma_i = 0$ for all $\sigma_i = 0$. Similar statements hold for rectangular **A**.

4.2 Numerical Reality

Thus far we have presumes that either a matrix *is* singular or *not*. From an analytic standpoint this is true, but not numerically. It is quite common for some σ_i to be very, very small, but not zero. Analytically the matrix is not singular, but numerically it is ill-conditioned. In this case Gaussian elimination, or the normal equations will yield a solution, but one when multiplied by **A** very poorly approximates **b**.

The solution vector **x** obtained by *zeroing* the small σ_i and then using equation (4.5) usually gives a better approximation than direct methods and the SVD solution where small σ_i are not zeroed.

Small σ_i are dominated by, and hence are an artifact of, roundoff error. But again we must ask "How small is too small?". A plausible answer is to edit $(1/\sigma - i)$ if $(\sigma_i/\sigma_{max}) < \mu\epsilon$ where μ is a constant and ϵ is the machine precision. MATLAB will tell you what ϵ is on your machine: in the command window, type *eps* followed by pressing *enter*. On my desktop $\epsilon = 2.2204 \times 10^{-16}$. It turns out that a useful value for μ is $rank(V)$. So we can state the following, easy rule for editing $1/\sigma_i$:

$$\mathrm{set}\ 1/\sigma - \mathrm{i} = 0\ \mathrm{if}\ (\sigma_\mathrm{i}/\sigma_\mathrm{max}) < \mathrm{rank(V)}\epsilon.$$

At first it may seem that zeroing the reciprocal of a singular value makes a bad situation worse by eliminating one linear combination of the set of equations we are attempting to solve. But we are really eliminating a linear combination of equations that is so corrupted by roundoff error that it is, at best, useless. In fact, worse than useless because it pushes the solution vector towards infinity in a direction parallel to a nullspace vector. This makes the roundoff error worse by inflating the residual $r = ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2$.

In general the matrix Σ will not be singular and no σ_i will have to be eliminated, unless there are column degeneracies in **A** (near linear combinations of the columns). The i^{th} column in **V** corresponding to the annihilated σ_i gives the linear combination of the elements of **x** that are ill-determined even by the over-determined set of equations. These elements of **x** are insensitive to the data, and should not be trusted.

4.3 Kinematic Synthesis

The design of a device to achieve a desired motion involves three distinct synthesis problems: type, number, and dimensional synthesis. These three problems will be individually discussed in the following three subsections.

4.3.1 Type Synthesis

Machines and mechanisms involve assemblies made from six basic types of mechanical components. These components were identified by the German kinematician Franz Reuleaux in his classic book from 1875 *Theroetische Kinematik* [11], translated into English in 1876 by Alexander Blackie William Kennedy (of Aronhold-Kennedy Theorem fame) as *Kinematics of Machines* [12]. The six categories are:

1. eye-bar type link (the lower pairs);
2. wheels, including gears;
3. cams in their many forms;
4. the screw, which transmits force and motion;
5. intermittent-motion devices, such as rachets;
6. tension-compression parts with one-way rigidity such as belts, chains, and hydraulic or pneumatic lines.

The selection of the type of mechanism needed to perform a specified motion depends to a great extent on design factors such as manufacturing processes, materials, safety, reliability, space, and economics, which are arguably outside the field of kinematics.

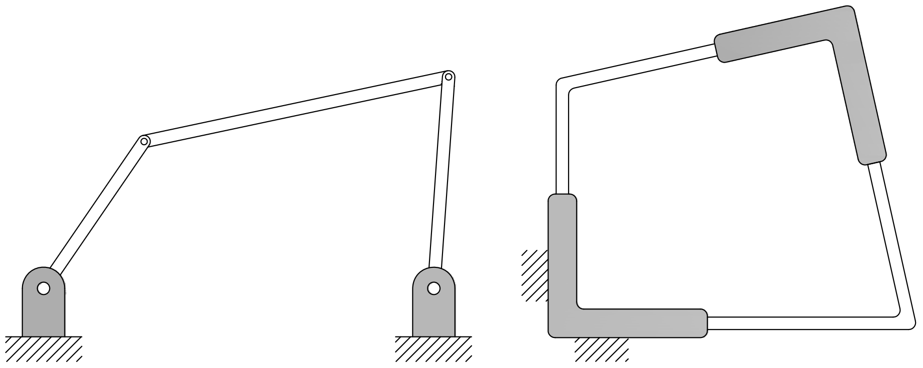


Figure 4.19: There are 16 distinct ways to make a planar four-bar using only R- and P-pairs taken 4 at a time, but only the extreme cases $RRRR$ and $PPPP$ are shown.

Even if the decision is made to use a planar four-bar mechanism, the *type* issue is not laid to rest. There remains the question: "what type of four-bar?". Should it be an $RRRR$? An $RRRP$? An $RPPR$? There can be 16 possible planar chains comprising R- and P-pairs. Any one of the four links joined by the four joints can be the fixed link, each resulting in a different motion, see Figure 4.19 for example.

4.3.2 Number Synthesis

Number synthesis is the second step in the process of mechanism design. It deals with determining the number of DOF and the number of links and joints required, see Figure 4.20. If each DOF of the linkage is to be controlled in a

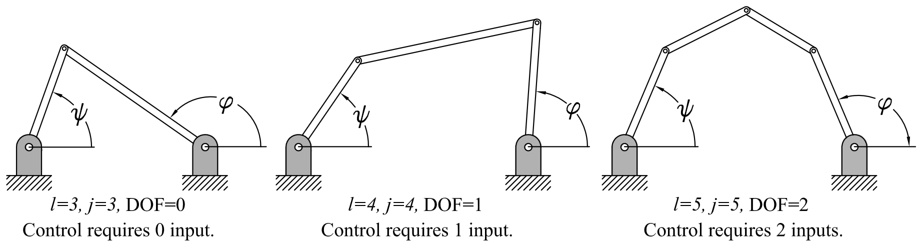


Figure 4.20: Linkages with 0, 1, and 2 DOF, requiring 0, 1, and 2 inputs to control each DOF.

closed kinematic chain, then an equal number of active inputs is required. If the linkage is a closed chain with 2 DOF then actuators are required at two joints. If more actuators are used the device is *redundantly actuated*, if less are used, the device is *under actuated* with uncontrolled DOF.

Open kinematic chains require that each joint be actuated. A $7R$ planar open chain, as in Figure 4.21, has 3 DOF, but requires 7 actuators to control the 3 DOF, although it is still said to be redundantly actuated. Note that in

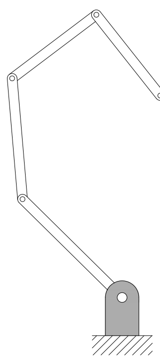


Figure 4.21: Planar 7R with 3 DOF, even though CGK says 7 DOF.

the $7R$ open planar chain example, the CGK formula predicts 7 DOF:

$$\begin{aligned} 3(8-1)-7(2)=21-14 &= 7 \text{ DOF}, \\ 6(8-1)-7(5)=42-35 &= 7 \text{ DOF}. \end{aligned}$$

The 7 DOF in E_2 and E_3 are indeed real, but linearly coupled. There is a maximum of 3 DOF in the plane, and a maximum of 6 DOF in Euclidean space.

4.3.3 Dimensional Synthesis

The third step in mechanism design is dimensional synthesis. We will consider three distinct problem classes in dimensional synthesis mentioned earlier, namely: function generation; rigid body guidance; and trajectory generation.

4.3.4 Function Generation

Dimensional synthesis boils down to solving a system of synthesis equations that express the desired motion in terms of the required unknown dimensions. These unknowns include, among other things, the link lengths, relative locations

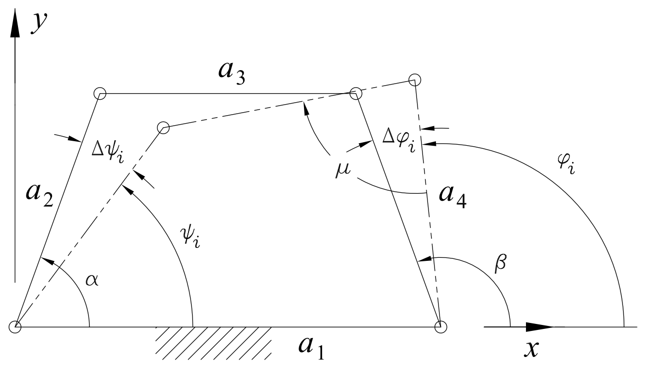


Figure 4.22: Elements of the Freudenstein parameters for a planar 4R linkage.

of revolute centers, and in particular for function generators, a *zero* or *home* configuration.

4.3.5 Dimensional Synthesis Continued

The synthesis equations for a four-bar function generator that are currently used were developed by Ferdinand Freudenstein in his Ph.D thesis in 1954, and published one year later [13]. Consider the mechanism in the Figure 4.22. The *Freudenstein parameters*, k_1, k_2, and k_3 are the link length ratios:

$$\left.\begin{array}{lcc} k_1 & \equiv & \dfrac{(a_1^2+a_2^2+a_4^2-a_3^2)}{2a_2a_4}, \\ k_2 & \equiv & \dfrac{a_1}{a_2}, \\ k_3 & \equiv & \dfrac{a_1}{a_4}. \end{array}\right\} \Leftrightarrow \left\{\begin{array}{lcc} a_1 & = & 1, \\ a_2 & = & \dfrac{1}{k_2}, \\ a_4 & = & \dfrac{1}{k_3}, \\ a_3 & = & (a_1^2+a_2^2+a_4^2-2a_2a_4k_1)^{1/2}. \end{array}\right. \tag{4.6}$$

The i^{th} configuration of the linkage illustrated in Figure 4.22 is governed by the following *synthesis equation*:

$$k_1 + k_2\cos(\varphi_i) - k_3\cos(\psi_i) \quad = \quad \cos(\psi_i - \varphi_i). \tag{4.7}$$

Three distinct sets of input and output angles (ψ, φ) define a fully determined set of synthesis equations, implying that if there is a solution for k_1, k_2, k_3 that satisfies the three equations, it will be unique. Thus, we can determine the link length ratios (k_1, k_2, k_3) that will exactly satisfy the desired function $\varphi = f(\psi)$. Of course for a function generator we only require ratios of the links because the scale of the mechanism is irrelevant. We will find it helpful to express a set of three, or more, synthesis equations in vector-matrix form, defined shortly in Section 4.4.

What happens if you need a function generator to be exact over a continuous range of motion of the mechanism? Sadly, you can't have it. A solution, in general, does not exist! Now do you hear the somewhat more enthusiastic chuckles of the mathematicians? To be exact, greater than three prescribed configurations results in an over-determined system (more equations than unknowns), which in general has no solution. We could try to make an improvement by formulating the problem so that there were more unknowns. Recall the steering condition, reproduced here:

$$S(\Delta\psi\Delta\varphi) \equiv \sin(\Delta\varphi - \Delta\psi) - \rho\sin(\Delta\psi)\sin(\Delta\varphi) \quad = \quad 0. \tag{4.8}$$

Define the *dial zeroes* to be α and β such that:

$$\begin{aligned} \psi_i &= \alpha + \Delta\psi_i, \\ \varphi_i &= \beta + \Delta\varphi_i, \end{aligned} \tag{4.9}$$

The synthesis equation now becomes:

$$k_1 + k_2\cos(\beta + \Delta\varphi_i) - k_3\cos(\alpha + \Delta\psi_i) = \cos\left((\alpha + \Delta\psi_i) - (\beta + \Delta\varphi_i)\right). \tag{4.10}$$

Now five sets of $(\Delta\psi_i, \Delta\varphi_i)$ will produce five equations in five unknowns. This means that the function will be generated exactly for five inputs only. There is no guarantee about the values in between the *precision* input-output values of the function. If this was how steering linkages were designed we would wear out our tires very fast, and have difficulty cornering.

Note, the design constant ρ must be selected, where $\rho = b/a$, with $b =$ track (distance between wheel carrier revolutes), and $a =$ wheelbase (distance between axle center-lines). A typical value for a car is $\rho = 0.5$. We will use this value in the following example shown in Figure 4.23. To resolve the problem, we resort to *approximate synthesis*, instead of *exact synthesis*. This involves over-constraining the synthesis equation. We generate a set of m equations in n unknowns with $m \gg n$. In our case $n = 5$. Two questions immediately arise: What value should be given to m? Moreover, how should the input data be distributed? The answer to both questions depends on context, and usually requires some experimentation.

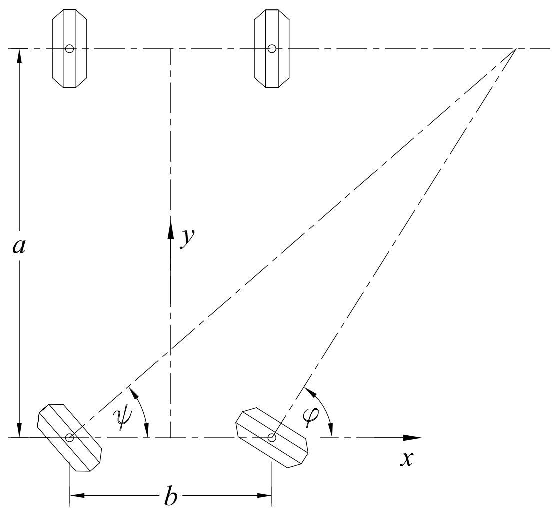


Figure 4.23: The steering condition, Figure 4.4 reproduced here for easy reference.

Assuming we have selected m and a distribution for the m input-output (I/O) values, we want a solution that will generate the function with the least error. The *design* and *structural* errors are two performance indicators used to optimise approximate synthesis.

4.4 Design Error

Given an over-constrained system of linear equations, an over-abundance of equations compared to unknowns, in general the system is inconsistent. If dial zeroes α and β have been determined, then we have the three unknown Freudenstein parameters $[k_1, k_2, k_3]^T = \mathbf{k}$ that must satisfy the $m \gg 3$ equations. Since, in general, no vector of Freudenstein parameters $\mathbf{k}$ can satisfy all the synthesis equations exactly and the product of the *synthesis matrix* with the Freudenstein parameter vector does not equal the vector whose elements are the cosine of the differences of the prescribed input and output angles, or $\mathbf{Sk} \neq \mathbf{b}$. Hence, the *design error* $\mathbf{d}$ is defined as:

$$\mathbf{d} = \mathbf{b} - \mathbf{Sk}, \tag{4.11}$$

where for the m equations $\mathbf{S}$ is an $m \times 3$ matrix called the synthesis matrix

$$\mathbf{S} = \begin{bmatrix} 1 & \cos(\varphi_1) & -\cos(\psi_1) \\ 1 & \cos(\varphi_2) & -\cos(\psi_2) \\ \vdots & \vdots & \vdots \\ 1 & \cos(\varphi_m) & -\cos(\psi_m) \end{bmatrix},$$

and $\mathbf{b}$ is the $m \times 1$ vector

$$\mathbf{b} = \begin{bmatrix} \cos(\psi_1 - \varphi_1) \\ \cos(\psi_2 - \varphi_2) \\ \vdots \\ \cos(\psi_m - \varphi_m) \end{bmatrix}. \tag{4.12}$$

$\mathbf{k}$ is the 3×1 vector of unknown Freudenstein parameters, and $\mathbf{d}$ is the $m \times 1$ design error vector. The Euclidean norm of $\mathbf{d}$ is a measure of how well the computed $\mathbf{k}$ satisfies the vector equation

$$\mathbf{Sk} = \mathbf{b}.$$

Clearly, if $m = 3$ and $\mathbf{S}$ is non-singular and square then

$$\mathbf{k} = \mathbf{S}^{-1}\mathbf{b}.$$

For $m > 3$ we can only approximate a solution for $\mathbf{k}$ using a linear least squares approach. This means that we require some form objective function to optimise. For the design error, the objective function is defined to be

$$z = (1/2)(\mathbf{d}^T\mathbf{W}\mathbf{d}), \tag{4.13}$$

which must be minimised over $\mathbf{k}$. The quantity $(\mathbf{d}^T\mathbf{W}\mathbf{d})^{1/2}$ is the *weighted Euclidean norm*, while $\mathbf{d}^T\mathbf{W}\mathbf{d}$ is the weighted Euclidean innerproduct. This requires a few words of explanation.

The matrix $\mathbf{W}$ is a diagonal matrix of positive weighting factors. They are used to make certain data points affect the minimisation more, or less, than others depending on their relative importance to the design. We usually start by setting $\mathbf{W} = \mathbf{I}$ (the identity matrix).

4.4.1 Minimising the Design Error

We are now in a position to resume examining the objective function for minimising $\mathbf{d}$ over $\mathbf{k}$:

$$z = (1/2)\mathbf{d}^T\mathbf{W}\mathbf{d}.$$

Typically, one half of the square of the weighted 2-norm is used. The weighted 2-norm of **d** over **k** can be minimised, in a least-squares sense by transforming **S** using Householder reflections which were discussed in Section 4.1.8, or singular value decomposition (SVD), discussed in Section 4.1.9.

Briefly, in MATLAB householder orthogonalization is invoked when a rectangular matrix is *divided* by a vector, using the syntax

$$\mathbf{k} = \mathbf{S} \setminus \mathbf{b}.$$

Be warned that this symbolic matrix division is *literally* symbolic: the operation does not exist in a vector space! The *backslash* operator invokes an algorithm that employs Householder orthogonalization of the matrix. Using SVD is somewhat more involved, but information on how well each element of **k** has been estimated is a direct, and extremely useful output of the decomposition.

We can now develop the equations needed to minimise the design error. To summarise, we have developed the expressions for the design error and objective function.

Design Error:

$$\mathbf{d} = \mathbf{b} - \mathbf{S}\mathbf{k}$$

$$\begin{aligned}
\mathbf{S} &= \begin{bmatrix} 1 & \cos\varphi_1 & -\cos\psi_1 \\ \vdots & \vdots & \vdots \\ 1 & \cos\varphi_m & -\cos\psi_m \end{bmatrix}, \\
\mathbf{k} &= \begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix}, \\
\mathbf{b} &= \begin{bmatrix} \cos(\psi_1 - \varphi_1) \\ \vdots \\ \cos(\psi_m - \varphi_m) \end{bmatrix}.
\end{aligned}$$

Objective Function:

$$z \equiv (1/2)(\mathbf{d}^{\,T}\mathbf{W}\mathbf{d}) \Rightarrow \text{minimise with respect to } \mathbf{k}.$$

Normality Condition: Given a vector-matrix equation $\mathbf{A}\mathbf{x} = \mathbf{b}$, the *normal equation* is that which minimises the sum of the square differences between the left and right sides [4]:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}.$$

We start by imposing the condition that

$$\frac{dz}{d\mathbf{k}} \;=\; \mathbf{0} \;\Rightarrow\; \begin{bmatrix} \partial z/\partial k_1 \\ \partial z/\partial k_2 \\ \partial z/\partial k_3 \end{bmatrix} \;=\; \mathbf{0},$$

which is called the *normality condition.* Next, we wish to express the normality condition in terms of the synthesis equations. Using the chain rule we see the normality condition possesses the form:

$$\begin{aligned} \frac{dz}{d\mathbf{k}} \quad &\Rightarrow \quad \frac{dz}{dk_i} = \frac{dz}{dd_j}\frac{dd_j}{dk_i} \\ &\Rightarrow \quad \frac{dz}{d\mathbf{k}} = \left(\frac{d\mathbf{d}}{d\mathbf{k}}\right)^T \frac{dz}{d\mathbf{d}}. \end{aligned}$$

From this, it is to be seen that

$$\left(\frac{d\mathbf{d}}{d\mathbf{k}}\right)^T = -\mathbf{S}^T, \quad \text{and} \quad \frac{dz}{d\mathbf{d}} = \mathbf{W}\mathbf{d}.$$

Therefore, the normality condition can be expressed as

$$\frac{dz}{d\mathbf{k}} \;=\; -\mathbf{S}^T\mathbf{W}\mathbf{d} \;=\; -\mathbf{S}^T\mathbf{W}(\mathbf{b}-\mathbf{S}\mathbf{k}) \;=\; 0.$$

Finally, from the normality condition we see that the normal equations possess the form that the definition requires:

$$-\mathbf{S}^T\mathbf{W}\mathbf{b}+\mathbf{S}^T\mathbf{W}\mathbf{S}\mathbf{k} \;=\; 0 \;\Rightarrow\; (\mathbf{S}^T\mathbf{W}\mathbf{S})_{(3\times3)}\mathbf{k}_{(3\times1)} \;=\; (\mathbf{S}^T\mathbf{W})_{(3\times m)}\mathbf{b}_{(m\times1)}.$$

Because $\mathbf{W}$ is positive definite then $\mathbf{S}_i^T\mathbf{W}\mathbf{S}_i > 0$ and the $\mathbf{k}$ obtained, at least conceptually, are the optimal k that minimise the objective function z, and we can state that

$$\mathbf{k}_{opt} = (\mathbf{S}^T\mathbf{W}\mathbf{S})^{-1}\mathbf{S}^T\mathbf{W}\mathbf{b}.$$

The term $(\mathbf{S}^T\mathbf{W}\mathbf{S})^{-1}\mathbf{S}^T\mathbf{W}$ in the above equation is called the weighted *Moore-Penrose Generalized Inverse* (M-PGI) of $\mathbf{S}$.

Calculating the M-PGI *conceptually* leads to $\mathbf{k}_{opt}$, however, this is generally not the case numerically when $m \gg n$ because $(\mathbf{S}^T\mathbf{W}\mathbf{S})^{-1}\mathbf{S}^T\mathbf{W}$ is typically ill-conditioned, meaning that the ratio of the associated maximum and minimum singular values of $\mathbf{S}^T\mathbf{W}\mathbf{S}$ is very large:

$$\frac{\sigma_{max}}{\sigma_{min}} \approx \infty \quad \text{because} \quad \sigma_{max} \gg \sigma_{min}.$$

The norm of the optimal solution vector, $\mathbf{k}_{opt}$, obtained using the M-PGI is in general so corrupted by round-off error that it is, at best, useless.

We *should* resort to a numerical approximation that is insensitive to the conditioning of the M-PGI of **S**. Geometrically, we must find **k** whose image under **S** is as close as possible to **b**, see Figure 4.24. What the figure illustrates is that the left and right hand sides of the synthesis equation can be represented as vector magnitudes both originating at the point 0. Different magnitudes for **k** effectively rotate **Sk** about 0 in the plane. The distance between **Sk** and **b** is **d**. Certainly, $\mathbf{k}_{opt}$ yields $\mathbf{d}_{min}$. This implies that Householder Reflections, described in Section 4.1.8, or some other numerical method should be used to identify $\mathbf{k}_{opt}$.

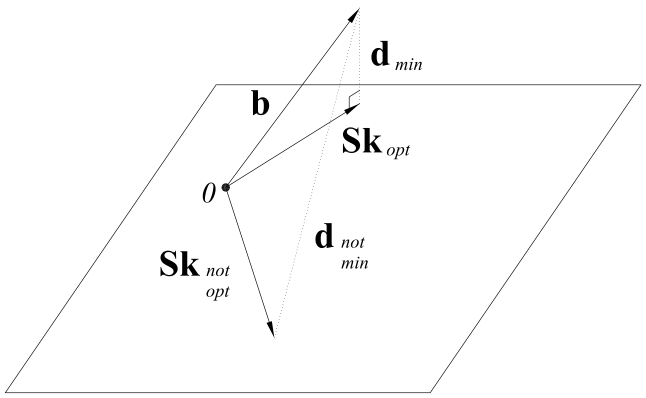


Figure 4.24: The optimal Freudenstein parameters, $\mathbf{k}_{opt}$, yields smallest difference, $\mathbf{d}_{min}$, between **Sk** and **b**.

4.5 Structural Error

The structural error is defined to be the difference between the prescribed and generated output angle values for a given input angle value. From a design optimisation perspective, it could be argued that the structural error is a better performance indicator than the design error in the sense that it directly measures the performance of the mechanism, while the design error is an algebraic residual created by any difference between **Sk** and **b** for identified values of **k** which minimise an objective function. However, in [14] it has been observed that as the cardinality of the prescribed discrete input-output (I/O) data-set increases, the corresponding linkages that minimise the Euclidean norms of the design and structural errors tend to converge to the same linkage. The important implication of this observation is that the minimisation of the Euclidean norm of the structural error can be accomplished indirectly via the minimisation of the corresponding norm of the design error, provided that a suitably large number of I/O pairs is prescribed.

The importance arises from the fact that the minimisation of the Euclidean norm of the design error leads to a linear least-squares problem whose solution can be obtained directly as opposed to iteratively [4], while the minimisation of the same norm of the structural error leads to a nonlinear least-squares problem, and hence, calls for an iterative solution [15]. Working with the observation reported in [14], an effort has been initiated to extend the discrete approximate synthesis problem into a continuous one by integrating the synthesis equations in the range of minimum input angle to maximum input angle and minimising the design error [16]. Currently, minimising the structural error in the context of continuous approximate synthesis is an open problem. The solution of this problem is an important challenge because it could stand to prove, or disprove the observation in [14].

Let the structural error vector $\mathbf{s}$ be defined as

$$\mathbf{s} \equiv [\varphi_{pres,i} - \varphi_{gen,i}],\ i\ \in\ 1, 2, .., m.$$

To minimise the Euclidean, or 2-Norm, of $\mathbf{s}$ requires an iterative numerical procedure. The scalar objective function to be minimised over $\mathbf{k}$ is

$$\zeta = (1/2)(\mathbf{s}^T \mathbf{W} \mathbf{s}).$$

Here, again $\mathbf{W}$ is an $m \times m$ diagonal matrix of weighting factors.

Minimising the norm of the structural error $\mathbf{s}$ is a non-linear problem. One way to proceed is to use the Gauss-Newton procedure, using the Freudenstein parameters that minimise the Euclidean norm of the design error as an initial guess. The guess is modified until the *normality condition*, which is the gradient of ζ with respect to $\mathbf{k}$, is satisfied to a specified tolerance. For *exact* synthesis the normality condition must be identically satisfied as

$$\frac{\partial \zeta}{\partial \mathbf{k}} = \mathbf{0}.$$

Setting $\mathbf{W} = \mathbf{I}$ for representational simplicity, the objective function reduces to

$$\zeta \quad = \quad \frac{1}{2}\mathbf{s}^T\mathbf{s}, \text{ which is essentially } \frac{1}{2}||\mathbf{s}||_2^2.$$

We wish to minimise this error over the Freudenstein parameter vector $\mathbf{k}$.

From the normality condition we can write

$$\frac{\partial\zeta}{\partial\mathbf{k}} = \left(\frac{\partial\mathbf{s}}{\partial\mathbf{k}}\right)^T_{3\times m}\frac{\partial\zeta}{\partial\mathbf{s}}_{m\times 1}. \tag{4.14}$$

It is to be seen that

$$\frac{\partial\zeta}{\partial\mathbf{s}} = \mathbf{s},$$

and that

$$\frac{\partial\mathbf{s}}{\partial\mathbf{k}} = \frac{\partial}{\partial\mathbf{k}}(\boldsymbol{\varphi}_{pres} - \boldsymbol{\varphi}_{gen}).$$

Since the prescribed output angle values are constant, by definition, we have

$$\frac{\partial\boldsymbol{\varphi}_{pres}}{\partial\mathbf{k}} = \mathbf{0},$$

which, in turn, implies that

$$\frac{\partial\mathbf{s}}{\partial\mathbf{k}} = -\frac{\partial\boldsymbol{\varphi}_{gen}}{\partial\mathbf{k}}.$$

Recall that the synthesis equations are $\mathbf{Sk} - \mathbf{b} = \mathbf{0}$ for an ideal function generator. It will be convenient to express these equations as a function of the generated output angles, the Freudenstein parameters, and the input angles:

$$\mathbf{Sk} - \mathbf{b} = \mathbf{f}(\boldsymbol{\varphi}_{gen}, \mathbf{k}; \boldsymbol{\psi}) = \mathbf{0}. \tag{4.15}$$

For m input angles we will obtain m values for the function:

$$\boldsymbol{\psi} = \begin{bmatrix}\psi_1\\ \vdots\\ \psi_m\end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix}f_1\\ \vdots\\ f_m\end{bmatrix}.$$

If the derivative of $\mathbf{f}$ with respect to $\mathbf{k}$ is identically equal to $\mathbf{0}$, then the *total derivative* of $\mathbf{f}$ with respect to $\mathbf{k}$ can be written as the sum of the partials:

$$\frac{d\mathbf{f}}{d\mathbf{k}} = \left(\frac{\partial\mathbf{f}}{\partial\mathbf{k}}\right)_{m\times 3} + \left(\frac{\partial\mathbf{f}}{\partial\boldsymbol{\varphi}_{gen}}\right)_{m\times m}\left(\frac{\partial\boldsymbol{\varphi}_{gen}}{\partial\mathbf{k}}\right)_{m\times 3} = \mathbf{0}_{m\times 3}. \tag{4.16}$$

As was just shown, the last partial in Equation 4.16 is

$$\frac{\partial\boldsymbol{\varphi}_{gen}}{\partial\mathbf{k}} = -\frac{\partial\mathbf{s}}{\partial\mathbf{k}},$$

because the prescribed values of the output angles are constant with respect to differentiation. The middle partial derivative of $\mathbf{f}$ with respect to $\boldsymbol{\varphi}_{gen}$ is an $m \times m$ diagonal matrix:

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\varphi}_{gen}} \;=\; \operatorname{diag}\left(\frac{\partial f_1}{\partial \varphi_{gen,1}}, \cdots, \frac{\partial f_m}{\partial \varphi_{gen,m}}\right) \;\equiv\; \mathbf{D}.$$

Since we assume that Equation (4.16) is identically zero, we have

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\varphi}_{gen}} \frac{\partial \boldsymbol{\varphi}_{gen}}{\partial \mathbf{k}} \;=\; -\frac{\partial \mathbf{f}}{\partial \mathbf{k}}.$$

Rearranging gives:

$$\frac{\partial \boldsymbol{\varphi}_{gen}}{\partial \mathbf{k}} \;=\; -\mathbf{D}^{-1}\frac{\partial \mathbf{f}}{\partial \mathbf{k}}. \tag{4.17}$$

Since $\mathbf{f}$ is linear in $\mathbf{k}$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{k}} \;=\; \mathbf{S}.$$

All of the above relations can be used to rewrite Equation (4.17) as:

$$\frac{\partial \mathbf{s}}{\partial \mathbf{k}} \;=\; -(-\mathbf{D}^{-1}\mathbf{S}) \;=\; \mathbf{D}^{-1}\mathbf{S}. \tag{4.18}$$

Substituting Equation (4.18) into Equation (4.14) yields

$$\frac{\partial \zeta}{\partial \mathbf{k}} \;=\; \left(\mathbf{D}^{-1}\mathbf{S}\right)^T \mathbf{s} \;=\; \mathbf{S}^T\mathbf{D}^{-1}\mathbf{s} \;=\; \mathbf{0}. \tag{4.19}$$

The Euclidean norm of the structural error $\mathbf{s}$ attains a minimum value when $\mathbf{D}^{-1}\mathbf{s}$ lies in the nullspace of $\mathbf{S}^T$. From Equation (4.15) we can write:

$$\boldsymbol{\varphi}_{gen} = \boldsymbol{\varphi}_{gen}(\mathbf{k}),$$

but we want

$$\boldsymbol{\varphi}_{gen}(\mathbf{k}) = \boldsymbol{\varphi}_{pres}.$$

Assume we have an estimate for $\mathbf{k}_{opt}$, which we call $\mathbf{k}^{\upsilon}$ obtained from the υ^{th} iteration. We can introduce a correction vector $\Delta\mathbf{k}$ so that we get

$$\boldsymbol{\varphi}_{gen}(\mathbf{k}^{\upsilon} + \Delta\mathbf{k}) \;=\; \boldsymbol{\varphi}_{pres}. \tag{4.20}$$

We can linearise the problem by expanding the left hand side of Equation (4.20) in a Taylor series and ignoring the nonlinear higher order terms:

$$\boldsymbol{\varphi}_{gen}(\mathbf{k}^{\upsilon}) + \frac{\partial \boldsymbol{\varphi}_{gen}}{\partial \mathbf{k}}|_{\mathbf{k}^{\upsilon}}\Delta\mathbf{k} + \text{higher order terms} \;=\; \boldsymbol{\varphi}_{pres}.$$

Substituting the results from Equation (4.17)

$$\frac{\partial \boldsymbol{\varphi}_{gen}}{\partial \mathbf{k}} = -\mathbf{D}^{-1}\mathbf{S},$$

into the Taylor series and rearranging leads to:

$$\mathbf{D}^{-1}\mathbf{S}\Delta\mathbf{k} = \boldsymbol{\varphi}_{gen}(\mathbf{k}^{\upsilon}) - \boldsymbol{\varphi}_{pres} = -\mathbf{s}^{\upsilon}. \tag{4.21}$$

Next, we can compute $\Delta\mathbf{k}$ conceptually as the M-PGI:

$$\Delta\mathbf{k} = -(\mathbf{S}^T\mathbf{D}^{-1}\mathbf{S})^{-1}\mathbf{S}^T\mathbf{s}^{\upsilon}. \tag{4.22}$$

However, in MATLAB a least squares approximation of Equation (4.22) for the υ^{th} iteration is obtained using Householder transformations as

$$\Delta\mathbf{k} = -\mathbf{D}^{-1}\mathbf{S} \setminus \mathbf{s}^{\upsilon}. \tag{4.23}$$

The procedure stops when

$$||\Delta\mathbf{k}|| < \epsilon > 0,$$

and

$$\Delta\mathbf{k} \rightarrow \mathbf{0} \Rightarrow \mathbf{S}^T\mathbf{D}^{-1}\mathbf{s}^{\upsilon} \rightarrow \mathbf{0},$$

which is *exactly* the normality condition. The procedure will converge to a minimum ζ.

The elements of the diagonal matrix $\mathbf{D}$ are computed as

$$\begin{aligned}\frac{\partial f_1}{\partial \varphi_{gen,1}} &= \frac{\partial}{\partial \varphi_{gen,1}}\left[k_1 + k_2\cos(\varphi_{gen,1}) - k_3\cos(\psi_1) - \cos(\psi_1 - \varphi_{gen,1})\right],\\ &= -k_2\sin(\varphi_{gen,1}) - \sin(\psi_1 - \varphi_{gen,1}).\end{aligned}$$

The matrix $\mathbf{D}$ can then be assembled as

$$\mathbf{D} = \begin{bmatrix} \frac{\partial f_1}{\partial \varphi_{gen,1}} & 0 & \cdots & 0 \\ 0 & \frac{\partial f_2}{\partial \varphi_{gen,2}} & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & \cdots & 0 & \frac{\partial f_m}{\partial \varphi_{gen,m}} \end{bmatrix} = \begin{bmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ 0 & d_{22} & 0 & \cdots & 0 \\ 0 & 0 & d_{33} & \cdots & \vdots \\ 0 & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & d_m \end{bmatrix}.$$

Conveniently, the inverse of a diagonal matrix consists of the reciprocals of the diagonal elements, giving

$$\mathbf{D}^{-1} = \begin{bmatrix} 1/d_{11} & 0 & 0 & \cdots & 0 \\ 0 & 1/d_{22} & 0 & \cdots & 0 \\ 0 & 0 & 1/d_{33} & \cdots & \vdots \\ 0 & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 1/d_m \end{bmatrix}.$$

The formulation above yields the minimum structural error in a least squares sense such that

$$\boldsymbol{\varphi}_{gen}(\mathbf{k}) \approx \boldsymbol{\varphi}_{pres}. \tag{4.24}$$

Bibliography

[1] W. Norris. *Modern Steam Road Wagons.* Longmans, Green, and Co., London, New York, Bombay, 1906.

[2] L. Burmester. *Lehrbuch der Kinematik.* Arthur Felix Verlag, Leipzig, Germany, 1888.

[3] J. E. Gentle. *Numerical Linear Algebra for Applications in Statistics.* Springer, New York, N.Y., U.S.A., 1998.

[4] G. Dahlquist and Å. Björck. *Numerical Methods,* translated from Swedish by N. Anderson. Prentice-Hall, Inc., U.S.A., 1969.

[5] C. L. Lawson. "Contributions to the Theory of Linear Least Maximum Approximations". Ph.D. Thesis, UCLA, Los Angeles, Ca., U.S.A., 1961.

[6] J. R. Rice and K. H. Usow. "The Lawson Algorithm and Extensions". *Mathematics of Computation*, vol. 101: pages 341–347, December 1967.

[7] F. Angeles and J. Angeles. "Synthesis of Function-generating Linkages with Minimax Structural Error: the Linear Case". *Proc. 13th IFToMM World Congress*, June 2011.

[8] H. Grassmann. *A New Branch of Mathematics,* English translation 1995 by Kannenberg, L. C. Open Court, Chicago, Ill., U.S.A., 1844.

[9] A.S. Householder. "Unitary Triangularization of a Nonsymmetric Matrix". *Journal of the Association for Computing Machinery (JACM)*, vol. 5, no. 4: pages 339–342, 1958.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C, 2nd Edition.* Cambridge University Press, Cambridge, England, 1992.

[11] F. Reuleaux. *Theoretische kinematik: Grundzüge einer Theorie des Maschinenwesens*. Braunschweig, F. Vieweg und Sohn, 1875.

[12] A.B. Kempe. "On a General Method of Describing Plane Curves of the n^{th} Degree by Linework". *Proceedings of the London Mathematical Society*, vol. 7: pages 213–216, 1876.

[13] F. Freudenstein. "Approximate Synthesis of Four-Bar Linkages". *Trans. ASME 77,* pages 853-861, 1955.

[14] M.J.D. Hayes, K. Parsa, and J. Angeles. "The Effect of Data-Set Cardinality on the Design and Structural Errors of Four-Bar Function-Generators". *Proceedings of the Tenth World Congress on the Theory of Machines and Mechanisms,* Oulu, Finland, pages 437–442, 1999.

[15] S. O. Tinubu and K. C. Gupta. "Optimal Synthesis of Function Generators Without the Branch Defect". *ASME, J. of Mech., Trans., and Autom. in Design*, vol. 106: pages 348354, 1984.

[16] A. Guigue and M.J.D. Hayes. "Continuous Approximate Synthesis of Planar Function-generators Minimising the Design Error". *Mechanism and Machine Theory*, vol. 101: pages 158–167, DOI: 10.1016/j.mechmachtheory.2016.03.012, 2016.