CS 205
Mathematical Methods for Robotics and Vision

Carlo Tomasi
Stanford University

Fall 1999

Chapter 1

Introduction

Robotics and computer vision are interdisciplinary subjects at the intersection of engineering and computer science. By their nature, they deal with both computers and the physical world. Although the former are in the latter, the workings of computers are best described in the black-and-white vocabulary of discrete mathematics, which is foreign to most classical models of reality, quantum physics notwithstanding.

This class surveys some of the key tools of applied math to be used at the interface of continuous and discrete. It is not on robotics or computer vision. These subjects evolve rapidly, but their mathematical foundations remain. Even if you will not pursue either field, the mathematics that you learn in this class will not go wasted. To be sure, applied mathematics is a discipline in itself and, in many universities, a separate department. Consequently, this class can be a quick tour at best. It does not replace calculus or linear algebra, which are assumed as prerequisites, nor is it a comprehensive survey of applied mathematics. What is covered is a compromise between the time available and what is useful and fun to talk about. Even if in some cases you may have to wait until you take a robotics or vision class to fully appreciate the usefulness of a particular topic, I hope that you will enjoy studying these subjects in their own right.

1.1 Who Should Take This Class

The main goal of this class is to present a collection of mathematical tools for both understanding and solving problems in robotics and computer vision. Several classes at Stanford cover the topics presented in this class, and do so in much greater detail. If you want to understand the full details of any one of the topics in the syllabus below, you should take one or more of these other classes instead. If you want to understand how these tools are implemented numerically, you should take one of the classes in the scientific computing program, which again cover these issues in much better detail. Finally, if you want to understand robotics or vision, you should take classes in these subjects, since this course is not on robotics or vision.

On the other hand, if you do plan to study robotics, vision, or other similar subjects in the future, and you regard yourself as a *user* of the mathematical techniques outlined in the syllabus below, then you may benefit from this course. Of the proofs, we will only see those that add understanding. Of the implementation aspects of algorithms that are available in, say, Matlab or LApack, we will only see the parts that we need to understand when we use the code.

In brief, we will be able to cover more topics than other classes because we will be often (but not always) unconcerned with rigorous proof or implementation issues. The emphasis will be on intuition and on practicality of the various algorithms. For instance, why are singular values important, and how do they relate to eigenvalues? What are the dangers of Newton-style minimization? How does a Kalman filter work, and why do PDEs lead to sparse linear systems? In this spirit, for instance, we discuss Singular Value Decomposition and Schur decomposition both because they never fail and because they clarify the structure of an algebraic or a differential linear problem.

1.2 Syllabus

Here is the ideal syllabus, but how much we cover depends on how fast we go.

1. Introduction
2. Unknown numbers
   - 2.1 Algebraic linear systems
     - 2.1.1 Characterization of the solutions to a linear system
     - 2.1.2 Gaussian elimination
     - 2.1.3 The Singular Value Decomposition
     - 2.1.4 The pseudoinverse
   - 2.2 Function optimization
     - 2.2.1 Newton and Gauss-Newton methods
     - 2.2.2 Levenberg-Marquardt method
     - 2.2.3 Constraints and Lagrange multipliers
3. Unknown functions of one real variable
   - 3.1 Ordinary differential linear systems
     - 3.1.1 Eigenvalues and eigenvectors
     - 3.1.2 The Schur decomposition
     - 3.1.3 Ordinary differential linear systems
     - 3.1.4 The matrix zoo
     - 3.1.5 Real, symmetric, positive-definite matrices
   - 3.2 Statistical estimation
     - 3.2.1 Linear estimation
     - 3.2.2 Weighted least squares
     - 3.2.3 The Kalman filter
4. Unknown functions of several variables
   - 4.1 Tensor fields of several variables
     - 4.1.1 Grad, div, curl
     - 4.1.2 Line, surface, and volume integrals
     - 4.1.3 Green's theorem and potential fields of two variables
     - 4.1.4 Stokes' and divergence theorems and potential fields of three variables
     - 4.1.5 Diffusion and flow problems
   - 4.2 Partial differential equations and sparse linear systems
     - 4.2.1 Finite differences
     - 4.2.2 Direct versus iterative solution methods
     - 4.2.3 Jacobi and Gauss-Seidel iterations
     - 4.2.4 Successive overrelaxation

1.3 Discussion of the Syllabus

In robotics, vision, physics, and any other branch of science whose subjects belongs to or interacts with the real world, mathematical models are developed that describe the relationship between different quantities. Some of these quantities are measured, or *sensed*, while others are inferred by calculation. For instance, in computer vision, equations tie the coordinates of points in space to the coordinates of corresponding points in different images. Image points are data, world points are unknowns to be computed.

Similarly, in robotics, a robot arm is modeled by equations that describe where each link of the robot is as a function of the configuration of the link's own joints and that of the links that support it. The desired position of the end effector, as well as the current configuration of all the joints, are the data. The unknowns are the motions to be imparted to the joints so that the end effector reaches the desired target position.

Of course, what is data and what is unknown depends on the problem. For instance, the vision system mentioned above could be looking at the robot arm. Then, the robot's end effector position could be the unknowns to be solved for by the vision system. Once vision has solved *its* problem, it could feed the robot's end-effector position as data for the robot controller to use in its own motion planning problem.

Sensed data are invariably noisy, because sensors have inherent limitations of accuracy, precision, resolution, and repeatability. Consequently, the systems of equations to be solved are typically overconstrained: there are more equations than unknowns, and it is hoped that the errors that affect the coefficients of one equation are partially cancelled by opposite errors in other equations. This is the basis of *optimization* problems: Rather than solving a minimal system exactly, an optimization problem tries to solve many equations simultaneously, each of them only approximately, but collectively as well as possible, according to some global criterion. Least squares is perhaps the most popular such criterion, and we will devote a good deal of attention to it.

In summary, the problems encountered in robotics and vision are optimization problems. A fundamental distinction between different classes of problems reflects the complexity of the unknowns. In the simplest case, unknowns are scalars. When there is more than one scalar, the unknown is a vector of numbers, typically either real or complex. Accordingly, the first part of this course will be devoted to describing systems of algebraic equations, especially linear equations, and optimization techniques for problems whose solution is a vector of reals. The main tool for understanding linear algebraic systems is the Singular Value Decomposition (SVD), which is both conceptually fundamental and practically of extreme usefulness. When the systems are nonlinear, they can be solved by various techniques of function optimization, of which we will consider the basic aspects.

Since physical quantities often evolve over time, many problems arise in which the unknowns are themselves functions of time. This is our second class of problems. Again, problems can be cast as a set of equations to be solved exactly, and this leads to the theory of Ordinary Differential Equations (ODEs). Here, "ordinary" expresses the fact that the unknown functions depend on just one variable (e.g., time). The main conceptual tool for addressing ODEs is the theory of eigenvalues, and the primary computational tool is the Schur decomposition.

Alternatively, problems with time varying solutions can be stated as minimization problems. When viewed globally, these minimization problems lead to the calculus of variations. Although important, we will skip the calculus of variations in this class because of lack of time. When the minimization problems above are studied locally, they become state estimation problems, and the relevant theory is that of dynamic systems and Kalman filtering.

The third category of problems concerns unknown functions of more than one variable. The images taken by a moving camera, for instance, are functions of time and space, and so are the unknown quantities that one can compute from the images, such as the distance of points in the world from the camera. This leads to Partial Differential equations (PDEs), or to extensions of the calculus of variations. In this class, we will see how PDEs arise, and how they can be solved numerically.

1.4 Books

The class will be based on these lecture notes, and additional notes handed out when necessary. Other useful references include the following.

- R. Courant and D. Hilbert, *Methods of Mathematical Physics,* Volume I and II, John Wiley and Sons, 1989.
- D. A. Danielson, *Vectors and Tensors in Engineering and Physics,* Addison-Wesley, 1992.
- A. Gelb *et al.*, *Applied Optimal Estimation,* MIT Press, 1974.
- P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1993.
- G. H. Golub and C. F. Van Loan, *Matrix Computations,* 2nd Edition, Johns Hopkins University Press, 1989, or 3rd edition, 1997.
- W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C,* 2nd Edition, Cambridge University Press, 1992.
- **G. Strang**, *Introduction to Applied Mathematics,* **Wellesley- Cambridge Press, 1986. This book is recommended reading, and two copies of it are on reserve at the math and computer science library.**
- A. E. Taylor and W. R. Mann, *Advanced Calculus,* 3rd Edition, John Wiley and Sons, 1983.

Chapter 2

Algebraic Linear Systems

An *algebraic* linear system is a set of m equations in n unknown scalars, which appear linearly. Without loss of generality, an algebraic linear system can be written as follows:

$$A\mathbf{x} = \mathbf{b} \tag{2.1}$$

where A is an $m \times n$ matrix, $\mathbf{x}$ is an n-dimensional vector that collects all of the unknowns, and $\mathbf{b}$ is a known vector of dimension m. In this chapter, we only consider the cases in which the entries of A, $\mathbf{b}$, and $\mathbf{x}$ are real numbers.

Two reasons are usually offered for the importance of linear systems. The first is apparently deep, and refers to the principle of superposition of effects. For instance, in dynamics, superposition of forces states that if force $\mathbf{f}_1$ produces acceleration $\mathbf{a}_1$ (both possibly vectors) and force $\mathbf{f}_2$ produces acceleration $\mathbf{a}_2$, then the combined force $\mathbf{f}_1 + \alpha\mathbf{f}_2$ produces acceleration $\mathbf{a}_1 + \alpha\mathbf{a}_2$. This is Newton's second law of dynamics, although in a formulation less common than the equivalent $\mathbf{f} = m\mathbf{a}$. Because Newton's laws are at the basis of the entire edifice of Mechanics, linearity appears to be a fundamental principle of Nature. However, like all physical laws, Newton's second law is an abstraction, and ignores viscosity, friction, turbulence, and other nonlinear effects. Linearity, then, is perhaps more in the physicist's mind than in reality: if nonlinear effects can be ignored, physical phenomena are linear!

A more pragmatic explanation is that linear systems are the only ones we know how to solve in general. This argument, which is apparently more shallow than the previous one, is actually rather important. Here is why. Given two algebraic equations in two variables,

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0 \,, \end{aligned}$$

we can eliminate, say, y and obtain the equivalent system

$$\begin{aligned} F(x) &= 0 \\ y &= h(x) \,. \end{aligned}$$

Thus, the original system is as hard to solve as it is to find the roots of the polynomial F in a single variable. Unfortunately, if f and g have degrees d_f and d_g, the polynomial F has generically degree $d_f d_g$.

Thus, the degree of a system of equations is, roughly speaking, the product of the degrees. For instance, a system of m quadratic equations corresponds to a polynomial of degree 2^m. The only case in which the exponential is harmless is when its base is 1, that is, when the system is linear.

In this chapter, we first review a few basic facts about vectors in sections 2.1 through 2.4. More specifically, we develop enough language to talk about linear systems and their solutions in geometric terms. In contrast with the promise made in the introduction, these sections contain quite a few proofs. This is because a large part of the course material is based on these notions, so we want to make sure that the foundations are sound. In addition, some of the proofs lead to useful algorithms, and some others prove rather surprising facts. Then, in section 2.5, we characterize the solutions of linear algebraic systems.

2.1 Linear (In)dependence

Given n vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ and n real numbers $x_1, \ldots, x_n$, the vector

$$\mathbf{b} = \sum_{j=1}^{n} x_j \mathbf{a}_j \tag{2.2}$$

is said to be a *linear combination* of $\mathbf{a}_1, \ldots, \mathbf{a}_n$ with coefficients $x_1, \ldots, x_n$.

The vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are *linearly dependent* if they admit the null vector as a nonzero linear combination. In other words, they are linearly dependent if there is a set of coefficients $x_1, \ldots, x_n$, not all of which are zero, such that

$$\sum_{j=1}^{n} x_j \mathbf{a}_j = \mathbf{0} . \tag{2.3}$$

For later reference, it is useful to rewrite the last two equalities in a different form. Equation (2.2) is the same as

$$A\mathbf{x} = \mathbf{b} \tag{2.4}$$

and equation (2.3) is the same as

$$A\mathbf{x} = \mathbf{0} \tag{2.5}$$

where

$$A = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix} , \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} , \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} .$$

If you are not convinced of these equivalences, take the time to write out the components of each expression for a small example. This is important. Make sure that you are comfortable with this.

Thus, the columns of a matrix A are dependent if there is a nonzero solution to the homogeneous system (2.5). Vectors that are not dependent are *independent.*

Theorem 2.1.1 *The vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are linearly dependent iff*[1] *at least one of them is a linear combination of the others.*

Proof. In one direction, dependency means that there is a nonzero vector $\mathbf{x}$ such that

$$\sum_{j=1}^{n} x_j \mathbf{a}_j = \mathbf{0} .$$

Let x_k be nonzero for some k. We have

$$\sum_{j=1}^{n} x_j \mathbf{a}_j = x_k \mathbf{a}_k + \sum_{j=1,\, j\neq k}^{n} x_j \mathbf{a}_j = \mathbf{0}$$

so that

$$\mathbf{a}_k = - \sum_{j=1,\, j\neq k}^{n} \frac{x_j}{x_k} \mathbf{a}_j \tag{2.6}$$

[1] "iff" means "if and only if."

as desired. The converse is proven similarly: if

$$\mathbf{a}_k = \sum_{j=1,\, j\neq k}^{n} x_j \mathbf{a}_j$$

for some k, then

$$\sum_{j=1}^{n} x_j \mathbf{a}_j = \mathbf{0}$$

by letting $x_k = -1$ (so that $\mathbf{x}$ is nonzero). Δ [2]

We can make the first part of the proof above even more specific, and state the following

Lemma 2.1.2 *If n nonzero vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are linearly dependent then at least one of them is a linear combination of the ones that* precede *it.*

Proof. Just let k be the *last* of the nonzero x_j. Then $x_j = 0$ for $j > k$ in (2.6), which then becomes

$$\mathbf{a}_k = \sum_{j<k}^{n} \frac{x_j}{x_k} \mathbf{a}_j$$

as desired. Δ

2.2 Basis

A set $\mathbf{a}_1, \ldots, \mathbf{a}_n$ is said to be a *basis* for a set B of vectors if the $\mathbf{a}_j$ are linearly independent and every vector in B can be written as a linear combination of them. B is said to be a *vector space* if it contains *all* the linear combinations of its basis vectors. In particular, this implies that every linear space contains the zero vector. The basis vectors are said to *span* the vector space.

Theorem 2.2.1 *Given a vector $\mathbf{b}$ in the vector space B and a basis $\mathbf{a}_1, \ldots, \mathbf{a}_n$ for B, the coefficients $x_1, \ldots, x_n$ such that*

$$\mathbf{b} = \sum_{j=1}^{n} x_j \mathbf{a}_j$$

are uniquely determined.

Proof. Let also

$$\mathbf{b} = \sum_{j=1}^{n} x'_j \mathbf{a}_j \; .$$

Then,

$$\mathbf{0} = \mathbf{b} - \mathbf{b} = \sum_{j=1}^{n} x_j \mathbf{a}_j - \sum_{j=1}^{n} x'_j \mathbf{a}_j = \sum_{j=1}^{n} (x_j - x'_j) \mathbf{a}_j$$

but because the $\mathbf{a}_j$ are linearly independent, this is possible only when $x_j - x'_j = 0$ for every j. Δ

The previous theorem is a very important result. An equivalent formulation is the following:

[2] This symbol marks the end of a proof.

If the columns $\mathbf{a}_1, \ldots, \mathbf{a}_n$ *of* A *are linearly independent and the system* $A\mathbf{x} = \mathbf{b}$ *admits a solution, then the solution is unique.*

Pause for a minute to verify that this formulation is equivalent.

Theorem 2.2.2 *Two different bases for the same vector space* B *have the same number of vectors.*

Proof. Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ and $\mathbf{a}'_1, \ldots, \mathbf{a}'_{n'}$ be two different bases for B. Then each $\mathbf{a}'_j$ is in B (why?), and can therefore be written as a linear combination of $\mathbf{a}_1, \ldots, \mathbf{a}_n$. Consequently, the vectors of the set

$$G = \mathbf{a}'_1, \mathbf{a}_1, \ldots, \mathbf{a}_n$$

must be linearly dependent. We call a set of vectors that contains a basis for B a *generating set* for B. Thus, G is a generating set for B.

The rest of the proof now proceeds as follows: we keep removing $\mathbf{a}$ vectors from G and replacing them with $\mathbf{a}'$ vectors in such a way as to keep G a generating set for B. Then we show that we cannot run out of $\mathbf{a}$ vectors before we run out of $\mathbf{a}'$ vectors, which proves that $n \geq n'$. We then switch the roles of $\mathbf{a}$ and $\mathbf{a}'$ vectors to conclude that $n' \geq n$. This proves that $n = n'$.

From lemma 2.1.2, one of the vectors in G is a linear combination of those preceding it. This vector cannot be $\mathbf{a}'_1$, since it has no other vectors preceding it. So it must be one of the $\mathbf{a}_j$ vectors. Removing the latter keeps G a generating set, since the removed vector depends on the others. Now we can add $\mathbf{a}'_2$ to G, writing it right after $\mathbf{a}'_1$:

$$G = \mathbf{a}'_1, \mathbf{a}'_2, \ldots .$$

G is still a generating set for B.

Let us continue this procedure until we run out of either $\mathbf{a}$ vectors to remove or $\mathbf{a}'$ vectors to add. The $\mathbf{a}$ vectors cannot run out first. Suppose in fact *per absurdum* that G is now made only of $\mathbf{a}'$ vectors, and that there are still left-over $\mathbf{a}'$ vectors that have not been put into G. Since the $\mathbf{a}'$s form a basis, they are mutually linearly independent. Since B is a vector space, all the $\mathbf{a}'$s are in B. But then G cannot be a generating set, since the vectors in it cannot generate the left-over $\mathbf{a}'$s, which are independent of those in G. This is absurd, because at every step we have made sure that G remains a generating set. Consequently, we must run out of $\mathbf{a}'$s first (or simultaneously with the last $\mathbf{a}$). That is, $n \geq n'$.

Now we can repeat the whole procedure with the roles of $\mathbf{a}$ vectors and $\mathbf{a}'$ vectors exchanged. This shows that $n' \geq n$, and the two results together imply that $n = n'$. Δ

A consequence of this theorem is that any basis for $\mathbf{R}^m$ has m vectors. In fact, the basis of *elementary vectors*

$$\mathbf{e}_j = j\text{th column of the } m \times m \text{ identity matrix}$$

is clearly a basis for $\mathbf{R}^m$, since any vector

$$\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

can be written as

$$\mathbf{b} = \sum_{j=1}^{m} b_j \mathbf{e}_j$$

and the $\mathbf{e}_j$ are clearly independent. Since this elementary basis has m vectors, theorem 2.2.2 implies that any other basis for $\mathbf{R}^m$ has m vectors.

Another consequence of theorem 2.2.2 is that n vectors of dimension $m < n$ are bound to be dependent, since any basis for $\mathbf{R}^m$ can only have m vectors.

Since all bases for a space have the same number of vectors, it makes sense to define the *dimension* of a space as the number of vectors in any of its bases.

2.3 Inner Product and Orthogonality

In this section we establish the geometric meaning of the algebraic notions of norm, inner product, projection, and orthogonality. The fundamental geometric fact that is assumed to be known is the *law of cosines*: given a triangle with sides a, b, c (see figure 2.1), we have

$$a^2 = b^2 + c^2 - 2bc\cos\theta$$

where θ is the angle between the sides of length b and c. A special case of this law is Pythagoras' theorem, obtained when $\theta = \pm\pi/2$.

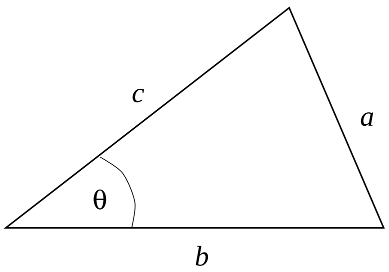

Figure 2.1: The law of cosines states that $a^2 = b^2 + c^2 - 2bc\cos\theta$.

In the previous section we saw that any vector in $\mathbf{R}^m$ can be written as the linear combination

$$\mathbf{b} = \sum_{j=1}^{m} b_j \mathbf{e}_j \tag{2.7}$$

of the elementary vectors that point along the coordinate axes. The length of these elementary vectors is clearly one, because each of them goes from the origin to the unit point of one of the axes. Also, any two of these vectors form a 90-degree angle, because the coordinate axes are orthogonal by construction. How long is $\mathbf{b}$? From equation (2.7) we obtain

$$\mathbf{b} = b_1\mathbf{e}_1 + \sum_{j=2}^{m} b_j \mathbf{e}_j$$

and the two vectors $b_1\mathbf{e}_1$ and $\sum_{j=2}^{m} b_j\mathbf{e}_j$ are orthogonal. By Pythagoras' theorem, the square of the length $\|\mathbf{b}\|$ of $\mathbf{b}$ is

$$\|\mathbf{b}\|^2 = b_1^2 + \|\sum_{j=2}^{m} b_j\mathbf{e}_j\|^2 .$$

Pythagoras' theorem can now be applied again to the last sum by singling out its first term $b_2\mathbf{e}_2$, and so forth. In conclusion,

$$\|\mathbf{b}\|^2 = \sum_{j=1}^{m} b_j^2 .$$

This result extends Pythagoras' theorem to m dimensions.

If we define the *inner product* of two m-dimensional vectors as follows:

$$\mathbf{b}^T\mathbf{c} = \sum_{j=1}^{m} b_j c_j ,$$

then

$$\|\mathbf{b}\|^2 = \mathbf{b}^T\mathbf{b} . \tag{2.8}$$

Thus, the squared length of a vector is the inner product of the vector with itself. Here and elsewhere, vectors are *column* vectors by default, and the symbol ^{T} makes them into row vectors.

Theorem 2.3.1

$$\mathbf{b}^T\mathbf{c} = \|\mathbf{b}\|\,\|\mathbf{c}\|\cos\theta$$

where θ is the angle between $\mathbf{b}$ and $\mathbf{c}$.

Proof. The law of cosines applied to the triangle with sides $\|\mathbf{b}\|$, $\|\mathbf{c}\|$, and $\|\mathbf{b}-\mathbf{c}\|$ yields

$$\|\mathbf{b}-\mathbf{c}\|^2 = \|\mathbf{b}\|^2 + \|\mathbf{c}\|^2 - 2\|\mathbf{b}\|\,\|\mathbf{c}\|\cos\theta$$

and from equation (2.8) we obtain

$$\mathbf{b}^T\mathbf{b} + \mathbf{c}^T\mathbf{c} - 2\mathbf{b}^T\mathbf{c} = \mathbf{b}^T\mathbf{b} + \mathbf{c}^T\mathbf{c} - 2\|\mathbf{b}\|\,\|\mathbf{c}\|\cos\theta\ .$$

Canceling equal terms and dividing by -2 yields the desired result. Δ

Corollary 2.3.2 *Two nonzero vectors $\mathbf{b}$ and $\mathbf{c}$ in $\mathbf{R}^m$ are mutually orthogonal iff $\mathbf{b}^T\mathbf{c} = 0$.*

Proof. When $\theta = \pm\pi/2$, the previous theorem yields $\mathbf{b}^T\mathbf{c} = 0$. Δ

Given two vectors $\mathbf{b}$ and $\mathbf{c}$ applied to the origin, the *projection* of $\mathbf{b}$ onto $\mathbf{c}$ is the vector from the origin to the point p on the line through $\mathbf{c}$ that is nearest to the endpoint of $\mathbf{b}$. See figure 2.2.

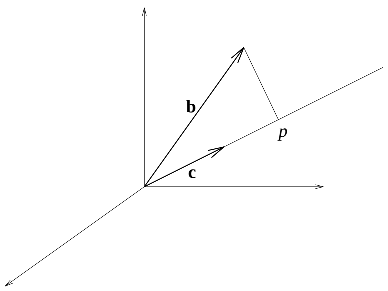

Figure 2.2: The vector from the origin to point p is the projection of $\mathbf{b}$ onto $\mathbf{c}$. The line from the endpoint of $\mathbf{b}$ to p is orthogonal to $\mathbf{c}$.

Theorem 2.3.3 *The projection of $\mathbf{b}$ onto $\mathbf{c}$ is the vector*

$$\mathbf{p} = P_{\mathbf{c}}\mathbf{b}$$

where $P_{\mathbf{c}}$ is the following square matrix:

$$P_{\mathbf{c}} = \frac{\mathbf{c}\mathbf{c}^T}{\mathbf{c}^T\mathbf{c}}\ .$$

Proof. Since by definition point p is on the line through $\mathbf{c}$, the projection vector $\mathbf{p}$ has the form $\mathbf{p} = a\mathbf{c}$, where a is some real number. From elementary geometry, the line between p and the endpoint of $\mathbf{b}$ is shortest when it is orthogonal to $\mathbf{c}$:

$$\mathbf{c}^T(\mathbf{b} - a\mathbf{c}) = 0$$

which yields

$$a = \frac{\mathbf{c}^T\mathbf{b}}{\mathbf{c}^T\mathbf{c}}$$

so that

$$\mathbf{p} = a\mathbf{c} = \mathbf{c}\,a = \frac{\mathbf{c}\mathbf{c}^T}{\mathbf{c}^T\mathbf{c}}\,\mathbf{b}$$

as advertised. Δ

2.4 Orthogonal Subspaces and the Rank of a Matrix

Linear transformations map spaces into spaces. It is important to understand exactly what is being mapped into what in order to determine whether a linear system has solutions, and if so how many. This section introduces the notion of orthogonality between spaces, defines the null space and range of a matrix, and its rank. With these tools, we will be able to characterize the solutions to a linear system in section 2.5. In the process, we also introduce a useful procedure (Gram-Schmidt) for orthonormalizing a set of linearly independent vectors.

Two vector spaces A and B are said to be *orthogonal* to one another when every vector in A is orthogonal to every vector in B. If vector space A is a subspace of $\mathbf{R}^m$ for some m, then the *orthogonal complement* of A is the set of all vectors in $\mathbf{R}^m$ that are orthogonal to all the vectors in A.

Notice that complement and orthogonal complement are very different notions. For instance, the complement of the xy plane in $\mathbf{R}^3$ is all of $\mathbf{R}^3$ except the xy plane, while the orthogonal complement of the xy plane is the z axis.

Theorem 2.4.1 *Any basis $\mathbf{a}_1, \ldots, \mathbf{a}_n$ for a subspace A of $\mathbf{R}^m$ can be extended into a basis for $\mathbf{R}^m$ by adding $m - n$ vectors $\mathbf{a}_{n+1}, \ldots, \mathbf{a}_m$.*

Proof. If $n = m$ we are done. If $n < m$, the given basis cannot generate all of $\mathbf{R}^m$, so there must be a vector, call it $\mathbf{a}_{n+1}$, that is linearly independent of $\mathbf{a}_1, \ldots, \mathbf{a}_n$. This argument can be repeated until the basis spans all of $\mathbf{R}^m$, that is, until $m = n$. Δ

Theorem 2.4.2 (Gram-Schmidt) *Given n vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n$, the following construction*

$$\begin{array}{l}
r = 0 \\
\textit{for } j = 1 \textit{ to } n \\
\quad \mathbf{a}'_j = \mathbf{a}_j - \sum_{l=1}^{r} (\mathbf{q}_l^T \mathbf{a}_j)\mathbf{q}_l \\
\quad \textit{if } \|\mathbf{a}'_j\| \neq \mathbf{0} \\
\qquad r = r + 1 \\
\qquad \mathbf{q}_r = \frac{\mathbf{a}'_j}{\|\mathbf{a}'_j\|} \\
\quad \textit{end} \\
\textit{end}
\end{array}$$

yields a set of orthonormal [3] vectors $\mathbf{q}_1 \ldots, \mathbf{q}_r$ that span the same space as $\mathbf{a}_1, \ldots, \mathbf{a}_n$.

[3] Orthonormal means orthogonal and with unit norm.

Proof. We first prove by induction on r that the vectors $\mathbf{q}_r$, are mutually orthonormal. If $r = 1$, there is little to prove. The normalization in the above procedure ensures that $\mathbf{q}_1$ has unit norm. Let us now assume that the procedure above has been performed a number $j-1$ of times sufficient to find $r-1$ vectors $\mathbf{q}_1, \ldots, \mathbf{q}_{m-1}$, and that these vectors are orthonormal (the inductive assumption). Then for any $i < r$ we have

$$\mathbf{q}_i^T \mathbf{a}'_j = \mathbf{q}_i^T \mathbf{a}_j - \sum_{l=1}^{r} (\mathbf{q}_l^T \mathbf{a}_j) \mathbf{q}_i^T \mathbf{q}_l = 0$$

because the term $\mathbf{q}_i^T \mathbf{a}_j$ cancels the i-th term $(\mathbf{q}_i^T \mathbf{a}_j)\mathbf{q}_i^T \mathbf{q}_i$ of the sum (remember that $\mathbf{q}_i^T \mathbf{q}_i = 1$), and the inner products $\mathbf{q}_i^T \mathbf{q}_l$ are zero by the inductive assumption. Because of the explicit normalization step $\mathbf{q}_r = \mathbf{a}'_j / \|\mathbf{a}'_j\|$, the vector $\mathbf{q}_r$, if computed, has unit norm.

Finally, we notice that the vectors $\mathbf{q}_j$ span the same space as the $\mathbf{a}_j$s, because the former are linear combinations of the latter, are orthonormal (and therefore independent), and equal in number to the number of linearly independent vectors in $\mathbf{a}_1, \ldots, \mathbf{a}_n$. Δ

Theorem 2.4.3 *If A is a subspace of $\mathbf{R}^m$ and $A^\perp$ is the orthogonal complement of A in $\mathbf{R}^m$, then*

$$\dim(A) + \dim(A^\perp) = m \ .$$

Proof. Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ be a basis for A. Extend this basis to a basis $\mathbf{a}_1, \ldots, \mathbf{a}_m$ for $\mathbf{R}^m$ (theorem 2.4.1). Orthonormalize this basis by the Gram-Schmidt procedure (theorem 2.4.2) to obtain $\mathbf{q}_1, \ldots, \mathbf{q}_m$. By construction, $\mathbf{q}_1, \ldots, \mathbf{q}_n$ span A. Because the new basis is orthonormal, all vectors generated by $\mathbf{q}_{n+1}, \ldots, \mathbf{q}_m$ are orthogonal to all vectors generated by $\mathbf{q}_1, \ldots, \mathbf{q}_n$, so there is a space of dimension at least $m-n$ that is orthogonal to A. On the other hand, the dimension of this orthogonal space cannot exceed $m-n$, because otherwise we would have more than m vectors in a basis for $\mathbf{R}^m$. Thus, the dimension of the orthogonal space $A^\perp$ is exactly $m-n$, as promised. Δ

We can now start to talk about matrices in terms of the subspaces associated with them. The *null space* null(A) of an $m \times n$ matrix A is the space of all n-dimensional vectors that are orthogonal to the rows of A. The *range* of A is the space of all m-dimensional vectors that are generated by the columns of A. Thus, $\mathbf{x} \in \text{null}(A)$ iff $A\mathbf{x} = 0$, and $\mathbf{b} \in \text{range}(A)$ iff $A\mathbf{x} = \mathbf{b}$ for some $\mathbf{x}$.

From theorem 2.4.3, if null(A) has dimension h, then the space generated by the rows of A has dimension $r = n - h$, that is, A has $n-h$ linearly independent rows. It is not obvious that the space generated by the *columns* of A has also dimension $r = n - h$. This is the point of the following theorem.

Theorem 2.4.4 *The number r of linearly independent columns of any $m \times n$ matrix A is equal to the number of its independent rows, and*

$$r = n - h$$

where $h = \dim(\text{null}(A))$.

Proof. We have already proven that the number of independent rows is $n-h$. Now we show that the number of independent columns is also $n-h$, by constructing a basis for range(A).

Let $\mathbf{v}_1, \ldots, \mathbf{v}_h$ be a basis for null(A), and extend this basis (theorem 2.4.1) into a basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ for $\mathbf{R}^n$. Then we can show that the $n-h$ vectors $A\mathbf{v}_{h+1}, \ldots, A\mathbf{v}_n$ are a basis for the range of A.

First, these $n-h$ vectors generate the range of A. In fact, given an arbitrary vector $\mathbf{b} \in \text{range}(A)$, there must be a linear combination of the columns of A that is equal to $\mathbf{b}$. In symbols, there is an n-tuple $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{b}$. The n-tuple $\mathbf{x}$ itself, being an element of $\mathbf{R}^n$, must be some linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_n$, our basis for $\mathbf{R}^n$:

$$\mathbf{x} = \sum_{j=1}^{n} c_j \mathbf{v}_j \ .$$

Thus,

$$\mathbf{b} = A\mathbf{x} = A\sum_{j=1}^{n} c_j \mathbf{v}_j = \sum_{j=1}^{n} c_j A\mathbf{v}_j = \sum_{j=h+1}^{n} c_j A\mathbf{v}_j$$

since $\mathbf{v}_1, \ldots, \mathbf{v}_h$ span null(A), so that $A\mathbf{v}_j = 0$ for $j = 1, \ldots, h$. This proves that the $n - h$ vectors $A\mathbf{v}_{h+1}, \ldots, A\mathbf{v}_n$ generate range(A).

Second, we prove that the $n-h$ vectors $A\mathbf{v}_{h+1}, \ldots, A\mathbf{v}_n$ are linearly independent. Suppose, *per absurdum*, that they are not. Then there exist numbers $x_{h+1}, \ldots, x_n$, not all zero, such that

$$\sum_{j=h+1}^{n} x_j A\mathbf{v}_j = 0$$

so that

$$A \sum_{j=h+1}^{n} x_j \mathbf{v}_j = 0 \ .$$

But then the vector $\sum_{j=h+1}^{n} x_j \mathbf{v}_j$ is in the null space of A. Since the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_h$ are a basis for null(A), there must exist coefficients $x_1, \ldots, x_h$ such that

$$\sum_{j=h+1}^{n} x_j \mathbf{v}_j = \sum_{j=1}^{h} x_j \mathbf{v}_j \ ,$$

in conflict with the assumption that the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly independent. Δ

Thanks to this theorem, we can define the *rank* of A to be equivalently the number of linearly independent columns or of linearly independent rows of A:

$$\text{rank}(A) = \dim(\text{range}(A)) = n - \dim(\text{null}(A)) \ .$$

2.5 The Solutions of a Linear System

Thanks to the results of the previous sections, we now have a complete picture of the four spaces associated with an $m \times n$ matrix A of rank r and null-space dimension h:

range(A); dimension $r = \text{rank}(A)$
null(A); dimension h
range$(A)^\perp$; dimension $m - r$
null$(A)^\perp$; dimension $r = n - h$.

The space range$(A)^\perp$ is called the *left nullspace* of the matrix, and null$(A)^\perp$ is called the *rowspace* of A. A frequently used synonym for "range" is *column space*. It should be obvious from the meaning of these spaces that

$$\begin{aligned} \text{null}(A)^\perp &= \text{range}(A^T) \\ \text{range}(A)^\perp &= \text{null}(A^T) \end{aligned}$$

where A^T is the *transpose* of A, defined as the matrix obtained by exchanging the rows of A with its columns.

Theorem 2.5.1 *The matrix A transforms a vector $\mathbf{x}$ in its null space into the zero vector, and an arbitrary vector $\mathbf{x}$ into a vector in* range(A).

This allows characterizing the set of solutions to linear system as follows. Let

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

be an $m \times n$ system (m can be less than, equal to, or greater than n). Also, let

$$r = \text{rank}(A)$$

be the number of linearly independent rows or columns of A. Then,

$$\begin{aligned} \mathbf{b} \notin \text{range}(A) &\Rightarrow \text{no solutions} \\ \mathbf{b} \in \text{range}(A) &\Rightarrow \infty^{n-r} \text{ solutions} \end{aligned}$$

with the convention that $\infty^0 = 1$. Here, ∞^k is the cardinality of a k-dimensional vector space.

In the first case above, there can be no linear combination of the columns (no $\mathbf{x}$ vector) that gives $\mathbf{b}$, and the system is said to be *incompatible*. In the second, *compatible* case, three possibilities occur, depending on the relative sizes of r, m, n:

- When $r = n = m$, the system is *invertible*. This means that there is exactly one $\mathbf{x}$ that satisfies the system, since the columns of A span all of $\mathbf{R}^n$. Notice that invertibility depends only on A, not on $\mathbf{b}$.

- When $r = n$ and $m > n$, the system is *redundant*. There are more equations than unknowns, but since $\mathbf{b}$ is in the range of A there is a linear combination of the columns (a vector $\mathbf{x}$) that produces $\mathbf{b}$. In other words, the equations are compatible, and exactly one solution exists. [4]

- When $r < n$ the system is *underdetermined*. This means that the null space is nontrivial (*i.e.*, it has dimension $h > 0$), and there is a space of dimension $h = n - r$ of vectors $\mathbf{x}$ such that $A\mathbf{x} = 0$. Since $\mathbf{b}$ is assumed to be in the range of A, there are solutions $\mathbf{x}$ to $A\mathbf{x} = \mathbf{b}$, but then for any $\mathbf{y} \in \text{null}(A)$ also $\mathbf{x} + \mathbf{y}$ is a solution:

  $$A\mathbf{x} = \mathbf{b}\,,\; A\mathbf{y} = 0 \;\Rightarrow\; A(\mathbf{x} + \mathbf{y}) = \mathbf{b}$$

  and this generates the $\infty^h = \infty^{n-r}$ solutions mentioned above.

Notice that if $r = n$ then n cannot possibly exceed m, so the first two cases exhaust the possibilities for $r = n$. Also, r cannot exceed either m or n. All the cases are summarized in figure 2.3.

Of course, listing all possibilities does not provide an operational method for determining the type of linear system for a given pair $A, \mathbf{b}$. Gaussian elimination, and particularly its version called *reduction to echelon form* is such a method, and is summarized in the next section.

2.6 Gaussian Elimination

Gaussian elimination is an important technique for solving linear systems. In addition to always yielding a solution, no matter whether the system is invertible or not, it also allows determining the rank of a matrix.

Other solution techniques exist for linear systems. Most notably, iterative methods solve systems in a time that depends on the accuracy required, while direct methods, like Gaussian elimination, are done in a finite amount of time that can be bounded given only the size of a matrix. Which method to use depends on

[4]Notice that the technical meaning of "redundant" has a stronger meaning than "with more equations than unknowns." The case $r < n < m$ is possible, has more equations (m) than unknowns (n), admits a solution if $\mathbf{b} \in \text{range}(A)$, but is called "underdetermined" because there are fewer (r) independent equations than there are unknowns (see next item). Thus, "redundant" means "with exactly one solution and with more equations than unknowns."

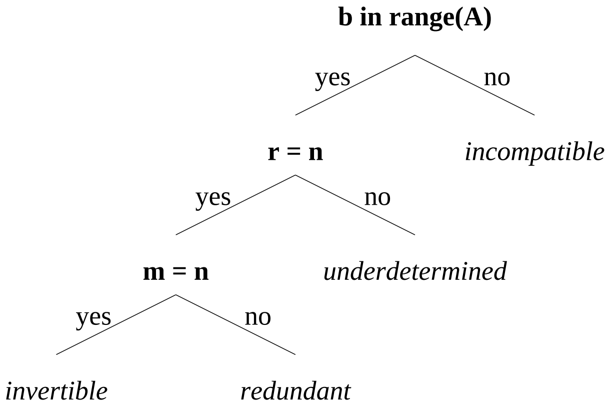

Figure 2.3: Types of linear systems.

the size and structure (e.g., sparsity) of the matrix, whether more information is required about the matrix of the system, and on numerical considerations. More on this in chapter 3.

Consider the $m \times n$ system

$$A\mathbf{x} = \mathbf{b} \tag{2.9}$$

which can be square or rectangular, invertible, incompatible, redundant, or underdetermined. In short, there are no restrictions on the system. Gaussian elimination replaces the rows of this system by linear combinations of the rows themselves until A is changed into a matrix U that is in the so-called *echelon form*. This means that

- Nonzero rows precede rows with all zeros. The first nonzero entry, if any, of a row, is called a *pivot*.
- Below each pivot is a column of zeros.
- Each pivot lies to the right of the pivot in the row above.

The same operations are applied to the rows of A and to those of $\mathbf{b}$, which is transformed to a new vector $\mathbf{c}$, so equality is preserved and solving the final system yields the same solution as solving the original one.

Once the system is transformed into echelon form, we compute the solution $\mathbf{x}$ by backsubstitution, that is, by solving the transformed system

$$U\mathbf{x} = \mathbf{c}\ .$$

2.6.1 Reduction to Echelon Form

The matrix A is reduced to echelon form by a process in $m-1$ steps. The first step is applied to $U^{(1)} = A$ and $\mathbf{c}^{(1)} = \mathbf{b}$. The k-th step is applied to rows $k, \ldots, m$ of $U^{(k)}$ and $\mathbf{c}^{(k)}$ and produces $U^{(k+1)}$ and $\mathbf{c}^{(k+1)}$. The last step produces $U^{(m)} = U$ and $\mathbf{c}^{(m)} = \mathbf{c}$. Initially, the "pivot column index" p is set to one. Here is step k, where u_{ij} denotes entry i, j of $U^{(k)}$:

Skip no-pivot columns If u_{ip} is zero for every $i = k, \ldots, m$, then increment p by 1. If p exceeds n stop.[5]

Row exchange Now $p \leq n$ and u_{ip} is nonzero for some $k \leq i \leq m$. Let l be one such value of i[6]. If $l \neq k$, exchange rows l and k of $U^{(k)}$ and of $\mathbf{c}^{(k)}$.

[5] "Stop" means that the entire algorithm is finished.

[6] Different ways of selecting l here lead to different numerical properties of the algorithm. Selecting the largest entry in the column leads to better round-off properties.

Triangularization The new entry u_{kp} is nonzero, and is called the *pivot*. For $i = k+1, \ldots, m$, subtract row k of $U^{(k)}$ multiplied by u_{ip}/u_{kp} from row i of $U^{(k)}$, and subtract entry k of $\mathbf{c}^{(k)}$ multiplied by u_{ip}/u_{kp} from entry i of $\mathbf{c}^{(k)}$. This zeros all the entries in the column below the pivot, and preserves the equality of left- and right-hand side.

When this process is finished, U is in echelon form. In particular, if the matrix is square and if all columns have a pivot, then U is upper-triangular.

2.6.2 Backsubstitution

A system

$$U\mathbf{x} = \mathbf{c} \tag{2.10}$$

in echelon form is easily solved for $\mathbf{x}$. To see this, we first solve the system symbolically, leaving undetermined variables specified by their name, and then transform this solution procedure into one that can be more readily implemented numerically.

Let r be the index of the last nonzero row of U. Since this is the number of independent rows of U, r is the rank of U. It is also the rank of A, because A and U admit exactly the same solutions and are equal in size. If $r < m$, the last $m - r$ equations yield a subsystem of the following form:

$$\mathbf{0} \begin{bmatrix} x_{r+1} \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} c_{r+1} \\ \vdots \\ c_m \end{bmatrix} .$$

Let us call this the *residual subsystem*. If on the other hand $r = m$ (obviously r cannot exceed m), there is no residual subsystem.

If there is a residual system (*i.e.*, $r < m$) and some of $c_{r+1}, \ldots, c_m$ are nonzero, then the equations corresponding to these nonzero entries are incompatible, because they are of the form $0 = c_i$ with $c_i \neq 0$. Since no vector $\mathbf{x}$ can satisfy these equations, the linear system admits no solutions: it is incompatible.

Let us now assume that either there is no residual system, or if there is one it is compatible, that is, $c_{r+1} = \ldots = c_m = 0$. Then, solutions exist, and they can be determined by *backsubstitution*, that is, by solving the equations starting from the last one and replacing the result in the equations higher up.

Backsubstitutions works as follows. First, remove the residual system, if any. We are left with an $r \times n$ system. In this system, call the variables corresponding to the r columns with pivots the *basic variables*, and call the other $n - r$ the *free variables*. Say that the pivot columns are $j_1, \ldots, j_r$. Then *symbolic backsubstitution* consists of the following sequence:

$$\begin{aligned} &\text{for} \quad i = r \text{ downto } 1 \\ &\qquad x_{j_i} = \frac{1}{u_{ij_i}} \left(c_i - \sum_{l=j_i+1}^{n} u_{il} x_l \right) \\ &\text{end} \end{aligned}$$

This is called symbolic backsubstitution because no numerical values are assigned to free variables. Whenever they appear in the expressions for the basic variables, free variables are specified by name rather than by value. The final result is a solution with as many free parameters as there are free variables. Since any value given to the free variables leaves the equality of system (2.10) satisfied, the presence of free variables leads to an infinity of solutions.

When solving a system in echelon form numerically, however, it is inconvenient to carry around nonnumeric symbol names (the free variables). Here is an equivalent solution procedure that makes this unnecessary. The solution obtained by backsubstitution is an affine function[7] of the free variables, and can therefore

[7] An affine function is a linear function plus a constant.

be written in the form

$$\mathbf{x} = \mathbf{v}_0 + x_{j_1}\mathbf{v}_1 + \ldots + x_{j_{n-r}}\mathbf{v}_{n-r} \tag{2.11}$$

where the x_{j_i} are the free variables. The vector $\mathbf{v}_0$ is the solution when all free variables are zero, and can therefore be obtained by replacing each free variable by zero during backsubstitution. Similarly, the vector $\mathbf{v}_i$ for $i = 1, \ldots, n-r$ can be obtained by solving the homogeneous system

$$U\mathbf{x} = \mathbf{0}$$

with $x_{j_i} = 1$ and all other free variables equal to zero. In conclusion, the general solution can be obtained by running backsubstitution $n-r+1$ times, once for the nonhomogeneous system, and $n-r$ times for the homogeneous system, with suitable values of the free variables. This yields the solution in the form (2.11).

Notice that the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n-r}$ form a basis for the null space of U, and therefore of A.

2.6.3 An Example

An example will clarify both the reduction to echelon form and backsubstitution. Consider the system

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

where

$$U^{(1)} = A = \begin{bmatrix} 1 & 3 & 3 & 2 \\ 2 & 6 & 9 & 5 \\ -1 & -3 & 3 & 0 \end{bmatrix} \quad , \quad \mathbf{c}^{(1)} = \mathbf{b} = \begin{bmatrix} 1 \\ 5 \\ 5 \end{bmatrix} .$$

Reduction to echelon form transforms A and $\mathbf{b}$ as follows. In the first step ($k = 1$), there are no no-pivot columns, so the pivot column index p stays at 1. Throughout this example, we choose a trivial pivot selection rule: we pick the first nonzero entry at or below row k in the pivot column. For $k = 1$, this means that $u_{11}^{(1)} = a_{11} = 1$ is the pivot. In other words, no row exchange is necessary.[8] The triangularization step subtracts row 1 multiplied by 2/1 from row 2, and subtracts row 1 multiplied by -1/1 from row 3. When applied to both $U^{(1)}$ and $\mathbf{c}^{(1)}$ this yields

$$U^{(2)} = \begin{bmatrix} 1 & 3 & 3 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 6 & 2 \end{bmatrix} \quad , \quad \mathbf{c}^{(2)} = \begin{bmatrix} 1 \\ 3 \\ 6 \end{bmatrix} .$$

Notice that now ($k = 2$) the entries $u_{ip}^{(2)}$ are zero for $i = 2, 3$, for both $p = 1$ and $p = 2$, so p is set to 3: the second pivot column is column 3, and $u_{23}^{(2)}$ is nonzero, so no row exchange is necessary. In the triangularization step, row 2 multiplied by 6/3 is subtracted from row 3 for both $U^{(2)}$ and $\mathbf{c}^{(2)}$ to yield

$$U = U^{(3)} = \begin{bmatrix} 1 & 3 & 3 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad , \quad \mathbf{c} = \mathbf{c}^{(3)} = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} .$$

There is one zero row in the left-hand side, and the rank of U and that of A is $r = 2$, the number of nonzero rows. The residual system is $0 = 0$ (compatible), and $r < n = 4$, so the system is underdetermined, with $\infty^{n-r} = \infty^2$ solutions.

In symbolic backsubstitution, the residual subsystem is first deleted. This yields the reduced system

$$\begin{bmatrix} 1 & 3 & 3 & 2 \\ 0 & 0 & 3 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \tag{2.12}$$

[8] Selecting the largest entry in the column at or below row k is a frequent choice, and this would have caused rows 1 and 2 to be switched.

The basic variables are x_1 and x_3, corresponding to the columns with pivots. The other two variables, x_2 and x_4, are free. Backsubstitution applied first to row 2 and then to row 1 yields the following expressions for the pivot variables:

$$\begin{aligned}
x_3 &= \frac{1}{u_{23}}(c_2 - u_{24}x_4) = \frac{1}{3}(3 - x_4) = 1 - \frac{1}{3}x_4 \\
x_1 &= \frac{1}{u_{11}}(c_1 - u_{12}x_2 - u_{13}x_3 - u_{14}x_4) = \frac{1}{1}(1 - 3x_2 - 3x_3 - 2x_4) \\
&= 1 - 3x_2 - (3 - x_4) - 2x_4 = -2 - 3x_2 - x_4
\end{aligned}$$

so the general solution is

$$\mathbf{x} = \begin{bmatrix} -2 - 3x_2 - x_4 \\ x_2 \\ 1 - \frac{1}{3}x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -1 \\ 0 \\ -\frac{1}{3} \\ 1 \end{bmatrix}.$$

This same solution can be found by the numerical backsubstitution method as follows. Solving the reduced system (2.12) with $x_2 = x_4 = 0$ by numerical backsubstitution yields

$$\begin{aligned}
x_3 &= \frac{1}{3}(3 - 1 \cdot 0) = 1 \\
x_1 &= \frac{1}{1}(1 - 3 \cdot 0 - 3 \cdot 1 - 2 \cdot 0) = -2
\end{aligned}$$

so that

$$\mathbf{v}_0 = \begin{bmatrix} -2 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Then $\mathbf{v}_1$ is found by solving the nonzero part (first two rows) of $U\mathbf{x} = \mathbf{0}$ with $x_2 = 1$ and $x_4 = 0$ to obtain

$$\begin{aligned}
x_3 &= \frac{1}{3}(-1 \cdot 0) = 0 \\
x_1 &= \frac{1}{1}(-3 \cdot 1 - 3 \cdot 0 - 2 \cdot 0) = -3
\end{aligned}$$

so that

$$\mathbf{v}_1 = \begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

Finally, solving the nonzero part of $U\mathbf{x} = \mathbf{0}$ with $x_2 = 0$ and $x_4 = 1$ leads to

$$\begin{aligned}
x_3 &= \frac{1}{3}(-1 \cdot 1) = -\frac{1}{3} \\
x_1 &= \frac{1}{1}\left(-3 \cdot 0 - 3 \cdot \left(-\frac{1}{3}\right) - 2 \cdot 1\right) = -1
\end{aligned}$$

so that

$$\mathbf{v}_2 = \begin{bmatrix} -1 \\ 0 \\ -\frac{1}{3} \\ 1 \end{bmatrix}$$

and

$$\mathbf{x} = \mathbf{v}_0 + x_2\mathbf{v}_1 + x_4\mathbf{v}_2 = \begin{bmatrix} -2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -1 \\ 0 \\ -\frac{1}{3} \\ 1 \end{bmatrix}$$

just as before.

As mentioned at the beginning of this section, Gaussian elimination is a *direct* method, in the sense that the answer can be found in a number of steps that depends only on the size of the matrix A. In the next chapter, we study a different method, based on the so-called the Singular Value Decomposition (SVD). This is an *iterative* method, meaning that an exact solution usually requires an infinite number of steps, and the number of steps necessary to find an approximate solution depends on the desired number of correct digits.

This state of affairs would seem to favor Gaussian elimination over the SVD. However, the latter yields a much more complete answer, since it computes bases for all the four spaces mentioned above, as well as a set of quantities, called the *singular values*, which provide great insight into the behavior of the linear transformation represented by the matrix A. Singular values also allow defining a notion of *approximate rank* which is very useful in a large number of applications. It also allows finding approximate solutions when the linear system in question is incompatible. In addition, for reasons that will become apparent in the next chapter, the computation of the SVD is numerically well behaved, much more so than Gaussian elimination. Finally, very efficient algorithms for the SVD exist. For instance, on a regular workstation, one can compute several thousand SVDs of 5×5 matrices in one second. More generally, the number of floating point operations necessary to compute the SVD of an $m \times n$ matrix is $amn^2 + bn^3$ where a, b are small numbers that depend on the details of the algorithm.

Chapter 3

The Singular Value Decomposition

In section 2, we saw that a matrix transforms vectors in its domain into vectors in its range (column space), and vectors in its null space into the zero vector. No nonzero vector is mapped into the left null space, that is, into the orthogonal complement of the range. In this section, we make this statement more specific by showing how *unit* vectors[1] in the rowspace are transformed by matrices. This describes the action that a matrix has on the *magnitudes* of vectors as well. To this end, we first need to introduce the notion of orthogonal matrices, and interpret them geometrically as transformations between systems of orthonormal coordinates. We do this in section 3.1. Then, in section 3.2, we use these new concepts to introduce the all-important concept of the Singular Value Decomposition (SVD). The chapter concludes with some basic applications and examples.

3.1 Orthogonal Matrices

Consider a point P in $\mathbf{R}^n$, with coordinates

$$\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

in a Cartesian reference system. For concreteness, you may want to think of the case $n = 3$, but the following arguments are general. Given any orthonormal basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ for $\mathbf{R}^n$, let

$$\mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

be the vector of coefficients for point P in the new basis. Then for any $i = 1, \ldots, n$ we have

$$\mathbf{v}_i^T \mathbf{p} = \mathbf{v}_i^T \sum_{j=1}^{n} q_j \mathbf{v}_j = \sum_{j=1}^{n} q_j \mathbf{v}_i^T \mathbf{v}_j = q_i \ ,$$

since the $\mathbf{v}_j$ are orthonormal. This is important, and may need emphasis:

If

$$\mathbf{p} = \sum_{j=1}^{n} q_j \mathbf{v}_j$$

[1]Vectors with unit norm.

and the vectors of the basis $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are orthonormal, then the coefficients q_j are the signed magnitudes of the projections of $\mathbf{p}$ onto the basis vectors:

$$q_j = \mathbf{v}_j^T \mathbf{p} . \tag{3.1}$$

We can write all n instances of equation (3.1) by collecting the vectors $\mathbf{v}_j$ into a matrix,

$$V = \left[\begin{array}{ccc} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{array} \right]$$

so that

$$\mathbf{q} = V^T \mathbf{p} . \tag{3.2}$$

Also, we can collect the n^2 equations

$$\mathbf{v}_i^T \mathbf{v}_j = \left\{ \begin{array}{cl} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{array} \right.$$

into the following matrix equation:

$$V^T V = I \tag{3.3}$$

where I is the $n \times n$ identity matrix. Since the inverse of a square matrix V is defined as the matrix V^{-1} such that

$$V^{-1} V = I , \tag{3.4}$$

comparison with equation (3.3) shows that the inverse of an orthogonal matrix V exists, and is equal to the transpose of V:

$$V^{-1} = V^T .$$

Of course, this argument requires V to be full rank, so that the solution V^{-1} to equation (3.4) is unique. However, V is certainly full rank, because it is made of orthonormal columns.

When V is $m \times n$ with $m > n$ and has orthonormal columns, this result is still valid, since equation (3.3) still holds. However, equation (3.4) defines what is now called the *left inverse* of V. In fact, $VV^{-1} = I$ cannot possibly have a solution when $m > n$, because the $m \times m$ identity matrix has m linearly independent [2] columns, while the columns of VV^{-1} are linear combinations of the n columns of V, so VV^{-1} can have at most n linearly independent columns.

For square, full-rank matrices ($r = m = n$), the distinction between left and right inverse vanishes. In fact, suppose that there exist matrices B and C such that $BV = I$ and $VC = I$. Then $B = B(VC) = (BV)C = C$, so the left and the right inverse are the same. We can summarize this discussion as follows:

Theorem 3.1.1 *The left inverse of an orthogonal $m \times n$ matrix V with $m \geq n$ exists and is equal to the transpose of V:*

$$V^{-1} V = V^T V = I .$$

In particular, if $m = n$, the matrix $V^{-1} = V^T$ is also the right inverse of V:

$$V \textit{ square} \quad \Rightarrow \quad V^{-1} V = V^T V = V V^{-1} = V V^T = I .$$

Sometimes, the geometric interpretation of equation (3.2) causes confusion, because two interpretations of it are possible. In the interpretation given above, the point P remains the same, and the underlying reference frame is changed from the elementary vectors $\mathbf{e}_j$ (that is, from the columns of I) to the vectors $\mathbf{v}_j$ (that is, to the columns of V). Alternatively, equation (3.2) can be seen as a transformation, in a fixed reference system, of point P with coordinates $\mathbf{p}$ into a different point Q with coordinates $\mathbf{q}$. This, however,

[2] Nay, orthonormal.

is relativity, and should not be surprising: If you spin clockwise on your feet, or if you stand still and the whole universe spins counterclockwise around you, the result is the same.[3]

Consistently with either of these geometric interpretations, we have the following result:

Theorem 3.1.2 *The norm of a vector* $\mathbf{x}$ *is not changed by multiplication by an orthogonal matrix* V*:*

$$\|V\mathbf{x}\| = \|\mathbf{x}\| \; .$$

Proof.

$$\|V\mathbf{x}\|^2 = \mathbf{x}^T V^T V \mathbf{x} = \mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|^2 \; .$$

Δ

We conclude this section with an obvious but useful consequence of orthogonality. In section 2.3 we defined the projection $\mathbf{p}$ of a vector $\mathbf{b}$ onto another vector $\mathbf{c}$ as the point on the line through $\mathbf{c}$ that is closest to $\mathbf{b}$. This notion of projection can be extended from lines to vector spaces by the following definition: The *projection* $\mathbf{p}$ of a point $\mathbf{b} \in \mathbf{R}^n$ *onto a subspace* C is the point in C that is closest to $\mathbf{b}$.

Also, for *unit* vectors $\mathbf{c}$, the projection matrix is $\mathbf{c}\mathbf{c}^T$ (theorem 2.3.3), and the vector $\mathbf{b} - \mathbf{p}$ is orthogonal to $\mathbf{c}$. An analogous result holds for subspace projection, as the following theorem shows.

Theorem 3.1.3 *Let* U *be an orthogonal matrix. Then the matrix* UU^T *projects any vector* $\mathbf{b}$ *onto* $\text{range}(U)$. *Furthermore, the difference vector between* $\mathbf{b}$ *and its projection* $\mathbf{p}$ *onto* $\text{range}(U)$ *is orthogonal to* $\text{range}(U)$*:*

$$U^T(\mathbf{b} - \mathbf{p}) = \mathbf{0} \; .$$

Proof. A point $\mathbf{p}$ in $\text{range}(U)$ is a linear combination of the columns of U:

$$\mathbf{p} = U\mathbf{x}$$

where $\mathbf{x}$ is the vector of coefficients (as many coefficients as there are columns in U). The squared distance between $\mathbf{b}$ and $\mathbf{p}$ is

$$\|\mathbf{b} - \mathbf{p}\|^2 = (\mathbf{b} - \mathbf{p})^T(\mathbf{b} - \mathbf{p}) = \mathbf{b}^T\mathbf{b} + \mathbf{p}^T\mathbf{p} - 2\mathbf{b}^T\mathbf{p} = \mathbf{b}^T\mathbf{b} + \mathbf{x}^T U^T U \mathbf{x} - 2\mathbf{b}^T U \mathbf{x} \; .$$

Because of orthogonality, $U^T U$ is the identity matrix, so

$$\|\mathbf{b} - \mathbf{p}\|^2 = \mathbf{b}^T\mathbf{b} + \mathbf{x}^T\mathbf{x} - 2\mathbf{b}^T U \mathbf{x} \; .$$

The derivative of this squared distance with respect to $\mathbf{x}$ is the vector

$$2\mathbf{x} - 2U^T\mathbf{b}$$

which is zero iff

$$\mathbf{x} = U^T\mathbf{b} \; ,$$

that is, when

$$\mathbf{p} = U\mathbf{x} = UU^T\mathbf{b}$$

as promised.

For this value of $\mathbf{p}$ the difference vector $\mathbf{b} - \mathbf{p}$ is orthogonal to $\text{range}(U)$, in the sense that

$$U^T(\mathbf{b} - \mathbf{p}) = U^T(\mathbf{b} - UU^T\mathbf{b}) = U^T\mathbf{b} - U^T\mathbf{b} = \mathbf{0} \; .$$

Δ

[3] At least geometrically. One solution may be more efficient than the other in other ways.

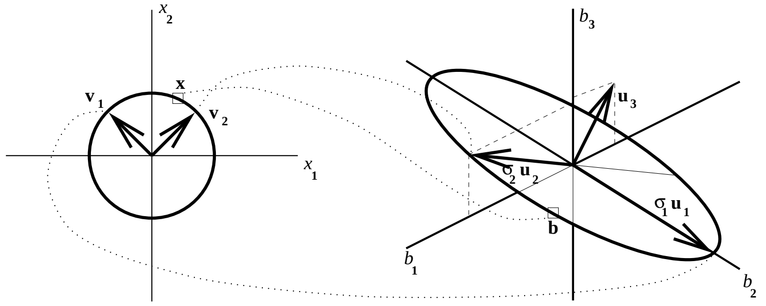

Figure 3.1: The matrix in equation (3.5) maps a circle on the plane into an ellipse in space. The two small boxes are corresponding points.

3.2 The Singular Value Decomposition

In these notes, we have often used geometric intuition to introduce new concepts, and we have then translated these into algebraic statements. This approach is successful when geometry is less cumbersome than algebra, or when geometric intuition provides a strong guiding element. The geometric picture underlying the Singular Value Decomposition is crisp and useful, so we will use geometric intuition again. Here is the main intuition:

> An $m \times n$ matrix A of rank r maps the r-dimensional unit hypersphere in rowspace(A) into an r-dimensional hyperellipse in range(A).

This statement is stronger than saying that A maps rowspace(A) into range(A), because it also describes what happens to the *magnitudes* of the vectors: a hypersphere is stretched or compressed into a hyperellipse, which is a quadratic hypersurface that generalizes the two-dimensional notion of ellipse to an arbitrary number of dimensions. In three dimensions, the hyperellipse is an ellipsoid, in one dimension it is a pair of points. In all cases, the hyperellipse in question is centered at the origin.

For instance, the rank-2 matrix

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} \sqrt{3} & \sqrt{3} \\ -3 & 3 \\ 1 & 1 \end{bmatrix} \tag{3.5}$$

transforms the unit circle on the plane into an ellipse embedded in three-dimensional space. Figure 3.1 shows the map

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

Two diametrically opposite points on the unit circle are mapped into the two endpoints of the major axis of the ellipse, and two other diametrically opposite points on the unit circle are mapped into the two endpoints of the minor axis of the ellipse. The lines through these two pairs of points on the unit circle are always orthogonal. This result can be generalized to any $m \times n$ matrix.

Simple and fundamental as this geometric fact may be, its proof by geometric means is cumbersome. Instead, we will prove it algebraically by first introducing the existence of the SVD and then using the latter to prove that matrices map hyperspheres into hyperellipses.

Theorem 3.2.1 *If A is a real $m \times n$ matrix then there exist orthogonal matrices*

$$\begin{aligned} U &= \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix} \in \mathcal{R}^{m\times m} \\ V &= \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix} \in \mathcal{R}^{n\times n} \end{aligned}$$

such that

$$U^T A V = \Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_p) \in \mathcal{R}^{m\times n}$$

where $p = \min(m, n)$ *and* $\sigma_1 \geq \ldots \geq \sigma_p \geq 0$. *Equivalently,*

$$A = U \Sigma V^T \ .$$

Proof. This proof is adapted from Golub and Van Loan, cited in the introduction to the class notes. Consider all vectors of the form

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

for $\mathbf{x}$ on the unit hypersphere $\|\mathbf{x}\| = 1$, and consider the scalar function $\|A\mathbf{x}\|$. Since $\mathbf{x}$ is defined on a compact set, this scalar function must achieve a maximum value, possibly at more than one point [4]. Let $\mathbf{v}_1$ be one of the vectors on the unit hypersphere in $\mathbf{R}^n$ where this maximum is achieved, and let $\sigma_1 \mathbf{u}_1$ be the corresponding vector $\sigma_1 \mathbf{u}_1 = A\mathbf{v}_1$ with $\|\mathbf{u}_1\| = 1$, so that σ_1 is the length of the corresponding $\mathbf{b} = A\mathbf{v}_1$.

By theorems 2.4.1 and 2.4.2, $\mathbf{u}_1$ and $\mathbf{v}_1$ can be extended into orthonormal bases for $\mathbf{R}^m$ and $\mathbf{R}^n$, respectively. Collect these orthonormal basis vectors into orthogonal matrices U_1 and V_1. Then

$$U_1^T A V_1 = S_1 = \begin{bmatrix} \sigma_1 & \mathbf{w}^T \\ \mathbf{0} & A_1 \end{bmatrix} \ .$$

In fact, the first column of AV_1 is $A\mathbf{v}_1 = \sigma_1 \mathbf{u}_1$, so the first entry of $U_1^T A V_1$ is $\mathbf{u}_1^T \sigma_1 \mathbf{u}_1 = \sigma_1$, and its other entries are $\mathbf{u}_j^T \sigma_1 \mathbf{u}_1 = 0$ because of orthonormality.

The matrix S_1 turns out to have even more structure than this: the row vector $\mathbf{w}^T$ is zero. Consider in fact the length of the vector

$$\frac{1}{\sqrt{\sigma_1^2 + \mathbf{w}^T\mathbf{w}}} S_1 \begin{bmatrix} \sigma_1 \\ \mathbf{w} \end{bmatrix} = \frac{1}{\sqrt{\sigma_1^2 + \mathbf{w}^T\mathbf{w}}} \begin{bmatrix} \sigma_1^2 + \mathbf{w}^T\mathbf{w} \\ A_1\mathbf{w} \end{bmatrix} \ . \tag{3.6}$$

From the last term, we see that the length of this vector is at least $\sqrt{\sigma_1^2 + \mathbf{w}^T\mathbf{w}}$. However, the longest vector we can obtain by premultiplying a unit vector by matrix S_1 has length σ_1. In fact, if $\mathbf{x}$ has unit norm so does $V_1\mathbf{x}$ (theorem 3.1.2). Then, the longest vector of the form $AV_1\mathbf{x}$ has length σ_1 (by definition of σ_1), and again by theorem 3.1.2 the longest vector of the form $S_1\mathbf{x} = U_1^T A V_1 \mathbf{x}$ has still length σ_1. Consequently, the vector in (3.6) cannot be longer than σ_1, and therefore $\mathbf{w}$ must be zero. Thus,

$$U_1^T A V_1 = S_1 = \begin{bmatrix} \sigma_1 & \mathbf{0}^T \\ \mathbf{0} & A_1 \end{bmatrix} \ .$$

The matrix A_1 has one fewer row and column than A. We can repeat the same construction on A_1 and write

$$U_2^T A_1 V_2 = S_2 = \begin{bmatrix} \sigma_2 & \mathbf{0}^T \\ \mathbf{0} & A_2 \end{bmatrix}$$

so that

$$\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & U_2^T \end{bmatrix} U_1^T A V_1 \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & V_2 \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & \mathbf{0}^T \\ 0 & \sigma_2 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{0} & A_2 \end{bmatrix} \ .$$

This procedure can be repeated until A_k vanishes (zero rows or zero columns) to obtain

$$U^T A V = \Sigma$$

[4] Actually, at least at two points: if $A\mathbf{v}_1$ has maximum length, so does $-A\mathbf{v}_1$.

where U^T and V are orthogonal matrices obtained by multiplying together all the orthogonal matrices used in the procedure, and

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_n \end{bmatrix} .$$

By construction, the σ_is are arranged in nonincreasing order along the diagonal of Σ, and are nonnegative.

Since matrices U and V are orthogonal, we can premultiply the matrix product in the theorem by U and postmultiply it by V^T to obtain

$$A = U\Sigma V^T .$$

Δ

We can now review the geometric picture in figure 3.1 in light of the singular value decomposition. In the process, we introduce some nomenclature for the three matrices in the SVD. Consider the map in figure 3.1, represented by equation (3.5), and imagine transforming point $\mathbf{x}$ (the small box at $\mathbf{x}$ on the unit circle) into its corresponding point $\mathbf{b} = A\mathbf{x}$ (the small box on the ellipse). This transformation can be achieved in three steps (see figure 3.2):

1. Write $\mathbf{x}$ in the frame of reference of the two vectors $\mathbf{v}_1, \mathbf{v}_2$ on the unit circle that map into the major axes of the ellipse. There are a few ways to do this, because axis endpoints come in pairs. Just pick one way, but order $\mathbf{v}_1, \mathbf{v}_2$ so they map into the major and the minor axis, in this order. Let us call $\mathbf{v}_1, \mathbf{v}_2$ the two *right singular vectors* of A. The corresponding axis unit vectors $\mathbf{u}_1, \mathbf{u}_2$ on the ellipse are called *left singular vectors.* If we define

   $$V = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} ,$$

   the new coordinates ξ of $\mathbf{x}$ become

   $$\xi = V^T\mathbf{x}$$

   because V is orthogonal.

2. Transform ξ into its image on a "straight" version of the final ellipse. "Straight" here means that the axes of the ellipse are aligned with the y_1, y_2 axes. Otherwise, the "straight" ellipse has the same shape as the ellipse in figure 3.1. If the lengths of the half-axes of the ellipse are σ_1, σ_2 (major axis first), the transformed vector η has coordinates

   $$\eta = \Sigma\xi$$

   where

   $$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix}$$

   is a diagonal matrix. The real, nonnegative numbers σ_1, σ_2 are called the *singular values* of A.

3. Rotate the reference frame in $\mathbf{R}^m = \mathbf{R}^3$ so that the "straight" ellipse becomes the ellipse in figure 3.1. This rotation brings η along, and maps it to $\mathbf{b}$. The components of η are the signed magnitudes of the projections of $\mathbf{b}$ along the unit vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ that identify the axes of the ellipse and the normal to the plane of the ellipse, so

   $$\mathbf{b} = U\eta$$

   where the orthogonal matrix

   $$U = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix}$$

   collects the left singular vectors of A.

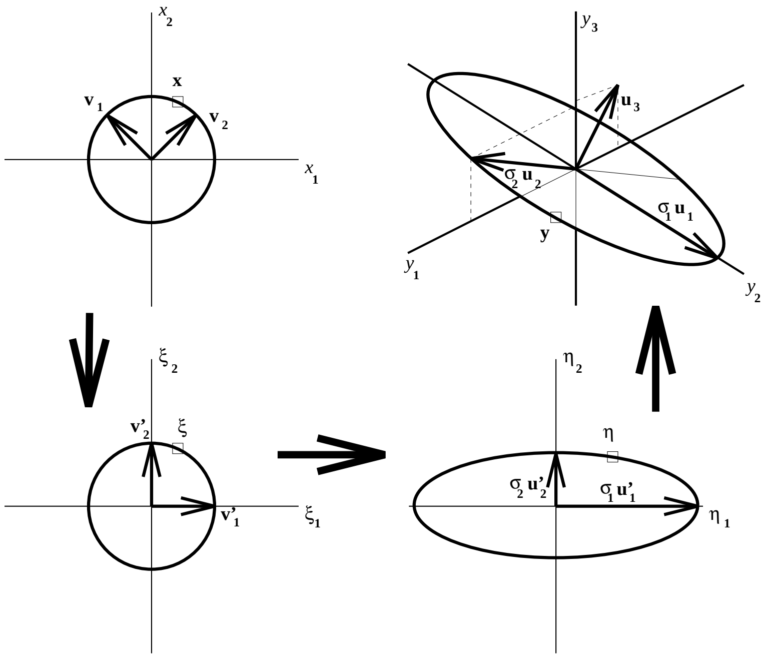

Figure 3.2: Decomposition of the mapping in figure 3.1.

We can concatenate these three transformations to obtain

$$\mathbf{b} = U\Sigma V^T \mathbf{x}$$

or

$$A = U\Sigma V^T$$

since this construction works for any point $\mathbf{x}$ on the unit circle. This is the SVD of A.

The singular value decomposition is "almost unique". There are two sources of ambiguity. The first is in the orientation of the singular vectors. One can flip any right singular vector, provided that the corresponding left singular vector is flipped as well, and still obtain a valid SVD. Singular vectors must be flipped in pairs (a left vector and its corresponding right vector) because the singular values are required to be nonnegative. This is a trivial ambiguity. If desired, it can be removed by imposing, for instance, that the first nonzero entry of every left singular value be positive.

The second source of ambiguity is deeper. If the matrix A maps a hypersphere into another hypersphere, the axes of the latter are not defined. For instance, the identity matrix has an infinity of SVDs, all of the form

$$I = UIU^T$$

where U is any orthogonal matrix of suitable size. More generally, whenever two or more singular values coincide, the subspaces identified by the corresponding left and right singular vectors are unique, but any orthonormal basis can be chosen within, say, the right subspace and yield, together with the corresponding left singular vectors, a valid SVD. Except for these ambiguities, the SVD is unique.

Even in the general case, the singular values of a matrix A are the lengths of the semi-axes of the hyperellipse E defined by

$$E = \{A\mathbf{x} \,:\, \|\mathbf{x}\| = 1\} .$$

The SVD reveals a great deal about the structure of a matrix. If we define r by

$$\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = 0 ,$$

that is, if σ_r is the smallest nonzero singular value of A, then

$$\begin{aligned}
\text{rank}(A) &= r \\
\text{null}(A) &= \text{span}\{\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n\} \\
\text{range}(A) &= \text{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_r\} .
\end{aligned}$$

The sizes of the matrices in the SVD are as follows: U is $m \times m$, Σ is $m \times n$, and V is $n \times n$. Thus, Σ has the same shape and size as A, while U and V are square. However, if $m > n$, the bottom $(m-n) \times n$ block of Σ is zero, so that the last $m - n$ columns of U are multiplied by zero. Similarly, if $m < n$, the rightmost $m \times (n - m)$ block of Σ is zero, and this multiplies the last $n - m$ rows of V. This suggests a "small," equivalent version of the SVD. If $p = \min(m, n)$, we can define $U_p = U(:, 1:p)$, $\Sigma_p = \Sigma(1:p, 1:p)$, and $V_p = V(:, 1:p)$, and write

$$A = U_p \Sigma_p V_p^T$$

where U_p is $m \times p$, Σ_p is $p \times p$, and V_p is $n \times p$.

Moreover, if $p - r$ singular values are zero, we can let $U_r = U(:, 1:r)$, $\Sigma_r = \Sigma(1:r, 1:r)$, and $V_r = V(:, 1:r)$, then we have

$$A = U_r \Sigma_r V_r^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T ,$$

which is an even smaller, *minimal*, SVD.

Finally, both the 2-norm and the Frobenius norm

$$\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$$

and

$$\|A\|_2 = \sup_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}$$

are neatly characterized in terms of the SVD:

$$\begin{aligned}
\|A\|_F^2 &= \sigma_1^2 + \ldots + \sigma_p^2 \\
\|A\|_2 &= \sigma_1 .
\end{aligned}$$

In the next few sections we introduce fundamental results and applications that testify to the importance of the SVD.

3.3 The Pseudoinverse

One of the most important applications of the SVD is the following result.

Theorem 3.3.1 *The minimum-norm least squares solution to a linear system $A\mathbf{x} = \mathbf{b}$, that is, the shortest vector $\mathbf{x}$ that achieves the*

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\| ,$$

is

$$\hat{\mathbf{x}} = V \Sigma^{\dagger} U^T \mathbf{b} \tag{3.7}$$

where

$$\Sigma^\dagger = \begin{bmatrix} 1/\sigma_1 & & & & & 0 & \cdots & 0 \\ & \ddots & & & & & & \\ & & 1/\sigma_r & & & \vdots & & \vdots \\ & & & 0 & & & & \\ & & & & \ddots & & & \\ & & & & & 0 & 0 \cdots & 0 \end{bmatrix}$$

is an $n \times m$ diagonal matrix.

The matrix

$$A^\dagger = V\Sigma^\dagger U^T$$

is called the *pseudoinverse* of A.

Proof. The minimum-norm Least Squares solution to

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

is the shortest vector $\mathbf{x}$ that minimizes

$$\|A\mathbf{x} - \mathbf{b}\|$$

that is,

$$\|U\Sigma V^T\mathbf{x} - \mathbf{b}\| \ .$$

This can be written as

$$\|U(\Sigma V^T\mathbf{x} - U^T\mathbf{b})\| \tag{3.8}$$

because U is an orthogonal matrix, $UU^T = I$. But orthogonal matrices do not change the norm of vectors they are applied to (theorem 3.1.2), so that the last expression above equals

$$\|\Sigma V^T\mathbf{x} - U^T\mathbf{b}\|$$

or, with $\mathbf{y} = V^T\mathbf{x}$ and $\mathbf{c} = U^T\mathbf{b}$,

$$\|\Sigma\mathbf{y} - \mathbf{c}\| \ .$$

In order to find the solution to this minimization problem, let us spell out the last expression. We want to minimize the norm of the following vector:

$$\begin{bmatrix} \sigma_1 & 0 & & \cdots & & 0 \\ 0 & \ddots & & \cdots & & 0 \\ & & \sigma_r & & & \\ \vdots & & & 0 & & \vdots \\ & & & & \ddots & \\ 0 & & & & & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_r \\ y_{r+1} \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} c_1 \\ \vdots \\ c_r \\ c_{r+1} \\ \vdots \\ c_m \end{bmatrix} .$$

The last $m - r$ differences are of the form

$$\mathbf{0} - \begin{bmatrix} c_{r+1} \\ \vdots \\ c_m \end{bmatrix}$$

and do not depend on the unknown $\mathbf{y}$. In other words, there is nothing we can do about those differences: if some or all the c_i for $i = r+1, \ldots, m$ are nonzero, we will not be able to zero these differences, and each of them contributes a *residual* $|c_i|$ to the solution. In each of the first r differences, on the other hand, the last $n - r$ components of $\mathbf{y}$ are multiplied by zeros, so they have no effect on the solution. Thus, there is freedom in their choice. Since we look for the minimum-norm solution, that is, for the shortest vector $\mathbf{x}$, we also want the shortest $\mathbf{y}$, because $\mathbf{x}$ and $\mathbf{y}$ are related by an orthogonal transformation. We therefore set $y_{r+1} = \ldots = y_n = 0$. In summary, the desired $\mathbf{y}$ has the following components:

$$
\begin{aligned}
y_i &= \frac{c_i}{\sigma_i} \quad \text{for } i = 1, \ldots, r \\
y_i &= 0 \quad \text{for } i = r+1, \ldots, n \,.
\end{aligned}
$$

When written as a function of the vector $\mathbf{c}$, this is

$$
\mathbf{y} = \Sigma^+ \mathbf{c} \,.
$$

Thus, the minimum-norm, least-squares solution to the original system is

$$
\hat{\mathbf{x}} = V\mathbf{y} = V\Sigma^+\mathbf{c} = V\Sigma^+U^T\mathbf{b}
$$

as promised. The residual, that is, the norm of $\|A\mathbf{x} - \mathbf{b}\|$ when $\mathbf{x}$ is the solution vector, is the norm of $\Sigma\mathbf{y} - \mathbf{c}$, since this vector is related to $A\mathbf{x} - \mathbf{b}$ by an orthogonal transformation (see equation (3.8)). In conclusion, the square of the residual is

$$
\|A\mathbf{x} - \mathbf{b}\|^2 = \|\Sigma\mathbf{y} - \mathbf{c}\|^2 = \sum_{i=r+1}^{m} c_i^2 = \sum_{i=r+1}^{m} (\mathbf{u}_i^T\mathbf{b})^2
$$

which is the projection of the right-hand side vector $\mathbf{b}$ onto the complement of the range of A. Δ

3.4 SVD Line Fitting

The Singular Value Decomposition of a matrix yields a simple method for fitting a line to a set of points on the plane.

3.4.1 Fitting a Line to a Set of Points

Let $\mathbf{p}_i = (x_i, y_i)^T$ be a set of $m \geq 2$ points on the plane, and let

$$ax + by - c = 0$$

be the equation of a line. If the lefthand side of this equation is multiplied by a nonzero constant, the line does not change. Thus, we can assume without loss of generality that

$$\|\mathbf{n}\| = a^2 + b^2 = 1 \ , \tag{3.9}$$

where the unit vector $\mathbf{n} = (a, b)^T$, orthogonal to the line, is called the *line normal.*

The distance from the line to the origin is $|c|$ (see figure 3.3), and the distance between the line $\mathbf{n}$ and a point $\mathbf{p}_i$ is equal to

$$d_i = |ax_i + by_i - c| = |\mathbf{p}_i^T \mathbf{n} - c| \ . \tag{3.10}$$

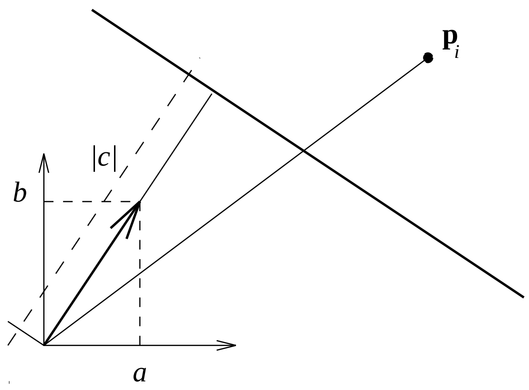

Figure 3.3: The distance between point $\mathbf{p}_i = (x_i, y_i)^T$ and line $ax + by - c = 0$ is $|ax_i + by_i - c|$.

The best-fit line minimizes the sum of the squared distances. Thus, if we let $\mathbf{d} = (d_1, \ldots, d_m)$ and $P = (\mathbf{p}_1 \ldots, \mathbf{p}_m)^T$, the best-fit line achieves the

$$\min_{\|\mathbf{n}\|=1} \|\mathbf{d}\|^2 = \min_{\|\mathbf{n}\|=1} \|P\mathbf{n} - c\mathbf{1}\|^2 \ . \tag{3.11}$$

In equation (3.11), $\mathbf{1}$ is a vector of m ones.

3.4.2 The Best Line Fit

Since the third line parameter c does not appear in the constraint (3.9), at the minimum (3.11) we must have

$$\frac{\partial \|\mathbf{d}\|^2}{\partial c} = 0 \ . \tag{3.12}$$

If we define the centroid $\mathbf{p}$ of all the points $\mathbf{p}_i$ as

$$\mathbf{p} = \frac{1}{m} P^T \mathbf{1} \ ,$$

equation (3.12) yields

$$\begin{aligned}
\frac{\partial \|\mathbf{d}\|^2}{\partial c} &= \frac{\partial}{\partial c}\left(\mathbf{n}^T P^T - c\mathbf{1}^T\right)(P\mathbf{n} - \mathbf{1}c) \\
&= \frac{\partial}{\partial c}\left(\mathbf{n}^T P^T P\mathbf{n} + c^2\mathbf{1}^T\mathbf{1} - 2\mathbf{n}^T P^T c\mathbf{1}\right) \\
&= 2\left(mc - \mathbf{n}^T P^T \mathbf{1}\right) = 0
\end{aligned}$$

from which we obtain

$$c = \frac{1}{m}\mathbf{n}^T P^T \mathbf{1} \ ,$$

that is,

$$c = \mathbf{p}^T \mathbf{n} \ .$$

By replacing this expression into equation (3.11), we obtain

$$\min_{\|\mathbf{n}\|=1} \|\mathbf{d}\|^2 = \min_{\|\mathbf{n}\|=1} \|P\mathbf{n} - \mathbf{1}\mathbf{p}^T\mathbf{n}\|^2 = \min_{\|\mathbf{n}\|=1} \|Q\mathbf{n}\|^2 \ ,$$

where $Q = P - \mathbf{1}\mathbf{p}^T$ collects the *centered* coordinates of the m points. To solve this constrained minimization problem, we recall that if $\mathbf{n}$ is on a circle, the shortest vector of the form $Q\mathbf{n}$ is obtained when $\mathbf{n}$ is the right singular vector $\mathbf{v}_2$ corresponding to the smaller σ_2 of the two singular values of Q. Furthermore, since $Q\mathbf{v}_2$ has norm σ_2, the residue is

$$\min_{\|\mathbf{n}\|=1} \|\mathbf{d}\| = \sigma_2$$

and more specifically the distances d_i are given by

$$\mathbf{d} = \sigma_2 \mathbf{u}_2$$

where $\mathbf{u}_2$ is the left singular vector corresponding to σ_2. In fact, when $\mathbf{n} = \mathbf{v}_2$, the SVD

$$Q = U\Sigma V^T = \sum_{i=1}^{2} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

yields

$$Q\mathbf{n} = Q\mathbf{v}_2 = \sum_{i=1}^{2} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \mathbf{v}_2 = \sigma_2 \mathbf{u}_2$$

because $\mathbf{v}_1$ and $\mathbf{v}_2$ are orthonormal vectors.

To summarize, to fit a line (a, b, c) to a set of m points $\mathbf{p}_i$ collected in the $m \times 2$ matrix $P = (\mathbf{p}_1 \ldots, \mathbf{p}_m)^T$, proceed as follows:

1. compute the centroid of the points ($\mathbf{1}$ is a vector of m ones):

$$\mathbf{p} = \frac{1}{m} P^T \mathbf{1}$$

2. form the matrix of centered coordinates:

$$Q = P - \mathbf{1}\mathbf{p}^T$$

3. compute the SVD of Q:

$$Q = U\Sigma V^T$$

4. the line normal is the second column of the 2×2 matrix V:

$$\mathbf{n} = (a, b)^T = \mathbf{v}_2 \; ,$$

5. the third coefficient of the line is

$$c = \mathbf{p}^T \mathbf{n}$$

6. the residue of the fit is

$$\min_{\|\mathbf{n}\|=1} \|\mathbf{d}\| = \sigma_2$$

The following `matlab` code implements the line fitting method.

```
function [l, residue] = linefit(P)
% check input matrix sizes
[m n] = size(P);
if n ~= 2, error('matrix P must be m x 2'), end
if m < 2, error('Need at least two points'), end
one = ones(m, 1);
% centroid of all the points
p = (P' * one) / m;
% matrix of centered coordinates
Q = P - one * p';
[U Sigma V] = svd(Q);
% the line normal is the second column of V
n = V(:, 2);
% assemble the three line coefficients into a column vector
l = [n ; p' * n];
% the smallest singular value of Q
% measures the residual fitting error
residue = Sigma(2, 2);
```

A useful exercise is to think how this procedure, or something close to it, can be adapted to fit a set of data points in $\mathbf{R}^m$ with an affine subspace of given dimension n. An affine subspace is a linear subspace plus a point, just like an arbitrary line is a line through the origin plus a point. Here "plus" means the following. Let L be a linear space. Then an affine space has the form

$$A = \mathbf{p} + L = \{\mathbf{a} \,|\, \mathbf{a} = \mathbf{p} + \mathbf{l} \text{ and } \mathbf{l} \in L\} \; .$$

Hint: minimizing the distance between a point and a subspace is equivalent to maximizing the norm of the projection of the point onto the subspace. The fitting problem (including fitting a line to a set of points) can be cast either as a maximization or a minimization problem.

Chapter 4

Function Optimization

There are three main reasons why most problems in robotics, vision, and arguably every other science or endeavor take on the form of optimization problems. One is that the desired goal may not be achievable, and so we try to get as close as possible to it. The second reason is that there may be more ways to achieve the goal, and so we can choose one by assigning a quality to all the solutions and selecting the best one. The third reason is that we may not know how to solve the system of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, so instead we minimize the norm $\|\mathbf{f}(\mathbf{x})\|$, which is a scalar function of the unknown vector $\mathbf{x}$.

We have encountered the first two situations when talking about linear systems. The case in which a linear system admits exactly one exact solution is simple but rare. More often, the system at hand is either incompatible (some say overconstrained) or, at the opposite end, underdetermined. In fact, some problems are both, in a sense. While these problems admit no exact solution, they often admit a multitude of approximate solutions. In addition, many problems lead to nonlinear equations.

Consider, for instance, the problem of Structure From Motion (SFM) in computer vision. Nonlinear equations describe how points in the world project onto the images taken by cameras at given positions in space. Structure from motion goes the other way around, and attempts to solve these equations: image points are given, and one wants to determine where the points in the world and the cameras are. Because image points come from noisy measurements, they are not exact, and the resulting system is usually incompatible. SFM is then cast as an optimization problem. On the other hand, the exact system (the one with perfect coefficients) is often close to being underdetermined. For instance, the images may be insufficient to recover a certain shape under a certain motion. Then, an additional criterion must be added to define what a "good" solution is. In these cases, the noisy system admits no exact solutions, but has many approximate ones.

The term "optimization" is meant to subsume both minimization and maximization. However, maximizing the scalar function $f(\mathbf{x})$ is the same as minimizing $-f(\mathbf{x})$, so we consider optimization and minimization to be essentially synonyms. Usually, one is after global minima. However, global minima are hard to find, since they involve a universal quantifier: $\mathbf{x}^*$ is a global minimum of f if for every other $\mathbf{x}$ we have $f(\mathbf{x}) \geq f(\mathbf{x}^*)$. Global minization techniques like simulated annealing have been proposed, but their convergence properties depend very strongly on the problem at hand. In this chapter, we consider local minimization: we pick a starting point $\mathbf{x}_0$, and we descend in the landscape of $f(\mathbf{x})$ until we cannot go down any further. The bottom of the valley is a local minimum.

Local minimization is appropriate if we know how to pick an $\mathbf{x}_0$ that is close to $\mathbf{x}^*$. This occurs frequently in feedback systems. In these systems, we start at a local (or even a global) minimum. The system then evolves and escapes from the minimum. As soon as this occurs, a control signal is generated to bring the system back to the minimum. Because of this immediate reaction, the old minimum can often be used as a starting point $\mathbf{x}_0$ when looking for the new minimum, that is, when computing the required control signal. More formally, we reach the correct minimum $\mathbf{x}^*$ as long as the initial point $\mathbf{x}_0$ is in the *basin of attraction* of $\mathbf{x}^*$, defined as the largest neighborhood of $\mathbf{x}^*$ in which $f(\mathbf{x})$ is convex.

Good references for the discussion in this chapter are *Matrix Computations*, *Practical Optimization*, and

Numerical Recipes in C, all of which are listed with full citations in section 1.4.

4.1 Local Minimization and Steepest Descent

Suppose that we want to find a local minimum for the scalar function f of the vector variable $\mathbf{x}$, starting from an initial point $\mathbf{x}_0$. Picking an appropriate $\mathbf{x}_0$ is crucial, but also very problem-dependent. We start from $\mathbf{x}_0$, and we go downhill. At every step of the way, we must make the following decisions:

- Whether to stop.
- In what direction to proceed.
- How long a step to take.

In fact, most minimization algorithms have the following structure:

$k = 0$
while $\mathbf{x}_k$ is not a minimum
  compute step direction $\mathbf{p}_k$ with $\|\mathbf{p}_k\| = 1$
  compute step size α_k
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $k = k + 1$
end.

Different algorithms differ in how each of these instructions is performed.

It is intuitively clear that the choice of the step size α_k is important. Too small a step leads to slow convergence, or even to lack of convergence altogether. Too large a step causes overshooting, that is, leaping past the solution. The most disastrous consequence of this is that we may leave the basin of attraction, or that we oscillate back and forth with increasing amplitudes, leading to instability. Even when oscillations decrease, they can slow down convergence considerably.

What is less obvious is that the best direction of descent is not necessarily, and in fact is quite rarely, the direction of steepest descent, as we now show. Consider a simple but important case,

$$f(\mathbf{x}) = c + \mathbf{a}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \tag{4.1}$$

where Q is a symmetric, positive definite matrix. *Positive definite* means that for every nonzero $\mathbf{x}$ the quantity $\mathbf{x}^T Q\mathbf{x}$ is positive. In this case, the graph of $f(\mathbf{x}) - c$ is a plane $\mathbf{a}^T\mathbf{x}$ plus a paraboloid.

Of course, if f were this simple, no descent methods would be necessary. In fact the minimum of f can be found by setting its gradient to zero:

$$\frac{\partial f}{\partial \mathbf{x}} = \mathbf{a} + Q\mathbf{x} = 0$$

so that the minimum $\mathbf{x}^*$ is the solution to the linear system

$$Q\mathbf{x} = -\mathbf{a}\ . \tag{4.2}$$

Since Q is positive definite, it is also invertible (why?), and the solution $\mathbf{x}^*$ is unique. However, understanding the behavior of minimization algorithms in this simple case is crucial in order to establish the convergence properties of these algorithms for more general functions. In fact, all smooth functions can be approximated by paraboloids in a sufficiently small neighborhood of any point.

Let us therefore assume that we minimize f as given in equation (4.1), and that at every step we choose the direction of steepest descent. In order to simplify the mathematics, we observe that if we let

$$\tilde{e}(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T Q(\mathbf{x} - \mathbf{x}^*)$$

then we have

$$\tilde{e}(\mathbf{x}) = f(\mathbf{x}) - c + \frac{1}{2}\mathbf{x}^{*T} Q \mathbf{x}^* = f(\mathbf{x}) - f(\mathbf{x}^*) \tag{4.3}$$

so that $\tilde{e}$ and f differ only by a constant. In fact,

$$\tilde{e}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}^T Q \mathbf{x} + \mathbf{x}^{*T} Q \mathbf{x}^* - 2\mathbf{x}^T Q \mathbf{x}^*) = \frac{1}{2}\mathbf{x}^T Q \mathbf{x} + \mathbf{a}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^{*T} Q \mathbf{x}^* = f(\mathbf{x}) - c + \frac{1}{2}\mathbf{x}^{*T} Q \mathbf{x}^*$$

and from equation (4.2) we obtain

$$f(\mathbf{x}^*) = c + \mathbf{a}^T \mathbf{x}^* + \frac{1}{2}\mathbf{x}^{*T} Q \mathbf{x}^* = c - \mathbf{x}^{*T} Q \mathbf{x}^* + \frac{1}{2}\mathbf{x}^{*T} Q \mathbf{x}^* = c - \frac{1}{2}\mathbf{x}^{*T} Q \mathbf{x}^* \ .$$

Since $\tilde{e}$ is simpler, we consider that we are minimizing $\tilde{e}$ rather than f. In addition, we can let

$$\mathbf{y} = \mathbf{x} - \mathbf{x}^* \ ,$$

that is, we can shift the origin of the domain to $\mathbf{x}^*$, and study the function

$$e(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T Q \mathbf{y}$$

instead of f or $\tilde{e}$, without loss of generality. We will transform everything back to f and $\mathbf{x}$ once we are done. Of course, by construction, the new minimum is at

$$\mathbf{y}^* = \mathbf{0}$$

where e reaches a value of zero:

$$e(\mathbf{y}^*) = e(\mathbf{0}) = 0 \ .$$

However, we let our steepest descent algorithm find this minimum by starting from the initial point

$$\mathbf{y}_0 = \mathbf{x}_0 - \mathbf{x}^* \ .$$

At every iteration k, the algorithm chooses the direction of steepest descent, which is in the direction

$$\mathbf{p}_k = -\frac{\mathbf{g}_k}{\|\mathbf{g}_k\|}$$

opposite to the gradient of e evaluated at $\mathbf{y}_k$:

$$\mathbf{g}_k = \mathbf{g}(\mathbf{y}_k) = \left.\frac{\partial e}{\partial \mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}_k} = Q\mathbf{y}_k \ .$$

We select for the algorithm the most favorable step size, that is, the one that takes us from $\mathbf{y}_k$ to the lowest point in the direction of $\mathbf{p}_k$. This can be found by differentiating the function

$$e(\mathbf{y}_k + \alpha \mathbf{p}_k) = \frac{1}{2}(\mathbf{y}_k + \alpha \mathbf{p}_k)^T Q (\mathbf{y}_k + \alpha \mathbf{p}_k)$$

with respect to α, and setting the derivative to zero to obtain the optimal step α_k. We have

$$\frac{\partial e(\mathbf{y}_k + \alpha \mathbf{p}_k)}{\partial \alpha} = (\mathbf{y}_k + \alpha \mathbf{p}_k)^T Q \mathbf{p}_k$$

and setting this to zero yields

$$\alpha_k = -\frac{(Q\mathbf{y}_k)^T \mathbf{p}_k}{\mathbf{p}_k^T Q \mathbf{p}_k} = -\frac{\mathbf{g}_k^T \mathbf{p}_k}{\mathbf{p}_k^T Q \mathbf{p}_k} = \|\mathbf{g}_k\| \frac{\mathbf{p}_k^T \mathbf{p}_k}{\mathbf{p}_k^T Q \mathbf{p}_k} = \|\mathbf{g}_k\| \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k} \ . \tag{4.4}$$

Thus, the basic step of our steepest descent can be written as follows:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \|\mathbf{g}_k\| \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k} \mathbf{p}_k$$

that is,

$$\mathbf{y}_{k+1} = \mathbf{y}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k} \mathbf{g}_k \ . \tag{4.5}$$

How much closer did this step bring us to the solution $\mathbf{y}^* = \mathbf{0}$? In other words, how much smaller is $e(\mathbf{y}_{k+1})$, relative to the value $e(\mathbf{y}_k)$ at the previous step? The answer is, often not much, as we shall now prove. The arguments and proofs below are adapted from D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, 1973.

From the definition of e and from equation (4.5) we obtain

$$\begin{aligned}
\frac{e(\mathbf{y}_k) - e(\mathbf{y}_{k+1})}{e(\mathbf{y}_k)} &= \frac{\mathbf{y}_k^T Q \mathbf{y}_k - \mathbf{y}_{k+1}^T Q \mathbf{y}_{k+1}}{\mathbf{y}_k^T Q \mathbf{y}_k} \\
&= \frac{\mathbf{y}_k^T Q \mathbf{y}_k - \left(\mathbf{y}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k} \mathbf{g}_k\right)^T Q \left(\mathbf{y}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k} \mathbf{g}_k\right)}{\mathbf{y}_k^T Q \mathbf{y}_k} \\
&= \frac{2 \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k} \mathbf{g}_k^T Q \mathbf{y}_k - \left(\frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T Q \mathbf{g}_k}\right)^2 \mathbf{g}_k^T Q \mathbf{g}_k}{\mathbf{y}_k^T Q \mathbf{y}_k} \\
&= \frac{2 \mathbf{g}_k^T \mathbf{g}_k \mathbf{g}_k^T Q \mathbf{y}_k - (\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{y}_k^T Q \mathbf{y}_k \, \mathbf{g}_k^T Q \mathbf{g}_k} \ .
\end{aligned}$$

Since Q is invertible we have

$$\mathbf{g}_k = Q \mathbf{y}_k \quad \Rightarrow \quad \mathbf{y}_k = Q^{-1} \mathbf{g}_k$$

and

$$\mathbf{y}_k^T Q \mathbf{y}_k = \mathbf{g}_k^T Q^{-1} \mathbf{g}_k$$

so that

$$\frac{e(\mathbf{y}_k) - e(\mathbf{y}_{k+1})}{e(\mathbf{y}_k)} = \frac{(\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{g}_k^T Q^{-1} \mathbf{g}_k \, \mathbf{g}_k^T Q \mathbf{g}_k} \ .$$

This can be rewritten as follows by rearranging terms:

$$e(\mathbf{y}_{k+1}) = \left(1 - \frac{(\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{g}_k^T Q^{-1} \mathbf{g}_k \, \mathbf{g}_k^T Q \mathbf{g}_k}\right) e(\mathbf{y}_k) \tag{4.6}$$

so if we can bound the expression in parentheses we have a bound on the rate of convergence of steepest descent. To this end, we introduce the following result.

Lemma 4.1.1 (Kantorovich inequality) *Let Q be a positive definite, symmetric, $n \times n$ matrix. For any vector $\mathbf{y}$ there holds*

$$\frac{(\mathbf{y}^T \mathbf{y})^2}{\mathbf{y}^T Q^{-1} \mathbf{y} \, \mathbf{y}^T Q \mathbf{y}} \geq \frac{4 \sigma_1 \sigma_n}{(\sigma_1 + \sigma_n)^2}$$

where σ_1 and σ_n are, respectively, the largest and smallest singular values of Q.

Proof. Let

$$Q = U\Sigma U^T$$

be the singular value decomposition of the symmetric (hence $V = U$) matrix Q. Because Q is positive definite, all its singular values are strictly positive, since the smallest of them satisfies

$$\sigma_n = \min_{\|\mathbf{y}\|=1} \mathbf{y}^T Q \mathbf{y} > 0$$

by the definition of positive definiteness. If we let

$$\mathbf{z} = U^T \mathbf{y}$$

we have

$$\frac{(\mathbf{y}^T\mathbf{y})^2}{\mathbf{y}^T Q^{-1}\mathbf{y}\,\mathbf{y}^T Q\mathbf{y}} = \frac{(\mathbf{y}^T U^T U\mathbf{y})^2}{\mathbf{y}^T U\Sigma^{-1}U^T\mathbf{y}\,\mathbf{y}^T U\Sigma U^T\mathbf{y}} = \frac{(\mathbf{z}^T\mathbf{z})^2}{\mathbf{z}^T\Sigma^{-1}\mathbf{z}\,\mathbf{z}^T\Sigma\mathbf{z}} = \frac{1/\sum_{i=1}^n \theta_i\sigma_i}{\sum_{i=1}^n \theta_i/\sigma_i} = \frac{\phi(\sigma)}{\psi(\sigma)} \tag{4.7}$$

where the coefficients

$$\theta_i = \frac{z_i^2}{\|\mathbf{z}\|^2}$$

add up to one. If we let

$$\sigma = \sum_{i=1}^n \theta_i\sigma_i\ , \tag{4.8}$$

then the numerator $\phi(\sigma)$ in (4.7) is $1/\sigma$. Of course, there are many ways to choose the coefficients θ_i to obtain a particular value of σ. However, each of the singular values σ_j can be obtained by letting $\theta_j = 1$ and all other θ_i to zero. Thus, the values $1/\sigma_j$ for $j = 1, \ldots, n$ are all on the curve $1/\sigma$. The denominator $\psi(\sigma)$ in (4.7) is a convex combination of points on this curve. Since $1/\sigma$ is a convex function of σ, the values of the denominator $\psi(\sigma)$ of (4.7) must be in the shaded area in figure 4.1. This area is delimited from above by the straight line that connects point $(\sigma_1, 1/\sigma_1)$ with point $(\sigma_n, 1/\sigma_n)$, that is, by the line with ordinate

$$\lambda(\sigma) = (\sigma_1 + \sigma_n - \sigma)/(\sigma_1\sigma_n)\ .$$

For the same vector of coefficients θ_i, the values of $\phi(\sigma)$, $\psi(\sigma)$, and $\lambda(\sigma)$ are on the vertical line corresponding to the value of σ given by (4.8). Thus an appropriate bound is

$$\frac{\phi(\sigma)}{\psi(\sigma)} \geq \min_{\sigma_1\leq\sigma\leq\sigma_n} \frac{\phi(\sigma)}{\lambda(\sigma)} = \min_{\sigma_1\leq\sigma\leq\sigma_n} \frac{1/\sigma}{(\sigma_1+\sigma_n-\sigma)/(\sigma_1\sigma_n)}\ .$$

The minimum is achieved at $\sigma = (\sigma_1 + \sigma_n)/2$, yielding the desired result. Δ

Thanks to this lemma, we can state the main result on the convergence of the method of steepest descent.

Theorem 4.1.2 *Let*

$$f(\mathbf{x}) = c + \mathbf{a}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T Q\mathbf{x}$$

be a quadratic function of $\mathbf{x}$, with Q symmetric and positive definite. For any $\mathbf{x}_0$, the method of steepest descent

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{g}_k^T Q\mathbf{g}_k}\mathbf{g}_k \tag{4.9}$$

where

$$\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k} = \mathbf{a} + Q\mathbf{x}_k$$

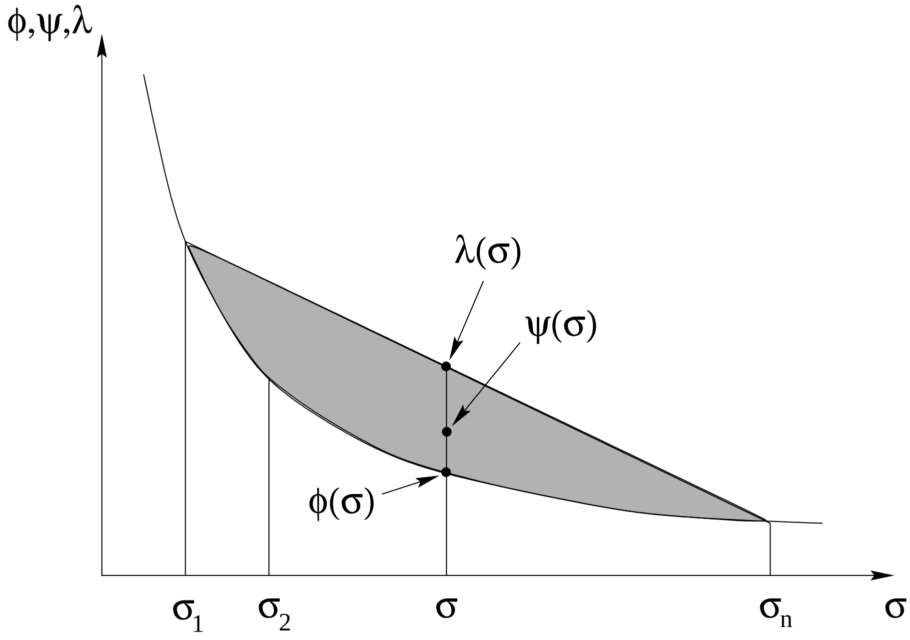

Figure 4.1: Kantorovich inequality.

converges to the unique minimum point

$$\mathbf{x}^* = -Q^{-1}\mathbf{a}$$

of f. Furthermore, at every step k there holds

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq \left(\frac{\sigma_1 - \sigma_n}{\sigma_1 + \sigma_n}\right)^2 (f(\mathbf{x}_k) - f(\mathbf{x}^*))$$

where σ_1 and σ_n are, respectively, the largest and smallest singular value of Q.

Proof. From the definitions

$$\mathbf{y} = \mathbf{x} - \mathbf{x}^* \quad \text{and} \quad e(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T Q \mathbf{y} \tag{4.10}$$

we immediately obtain the expression for steepest descent in terms of f and $\mathbf{x}$. By equations (4.3) and (4.6) and the Kantorovich inequality we obtain

$$\begin{aligned} f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) &= e(\mathbf{y}_{k+1}) = \left(1 - \frac{(\mathbf{g}_k^T \mathbf{g}_k)^2}{\mathbf{g}_k^T Q^{-1} \mathbf{g}_k \, \mathbf{g}_k^T Q \mathbf{g}_k}\right) e(\mathbf{y}_k) \leq \left(1 - \frac{4\sigma_1\sigma_n}{(\sigma_1+\sigma_n)^2}\right) e(\mathbf{y}_k) & (4.11) \\ &= \left(\frac{\sigma_1 - \sigma_n}{\sigma_1 + \sigma_n}\right)^2 (f(\mathbf{x}_k) - f(\mathbf{x}^*)) \, . & (4.12) \end{aligned}$$

Since the ratio in the last term is smaller than one, it follows immediately that $f(\mathbf{x}_k) - f(\mathbf{x}^*) \to 0$ and hence, since the minimum of f is unique, that $\mathbf{x}_k \to \mathbf{x}^*$. Δ

The ratio $\kappa(Q) = \sigma_1/\sigma_n$ is called the *condition number* of Q. The larger the condition number, the closer the fraction $(\sigma_1 - \sigma_n)/(\sigma_1 + \sigma_n)$ is to unity, and the slower convergence. It is easily seen why this happens in

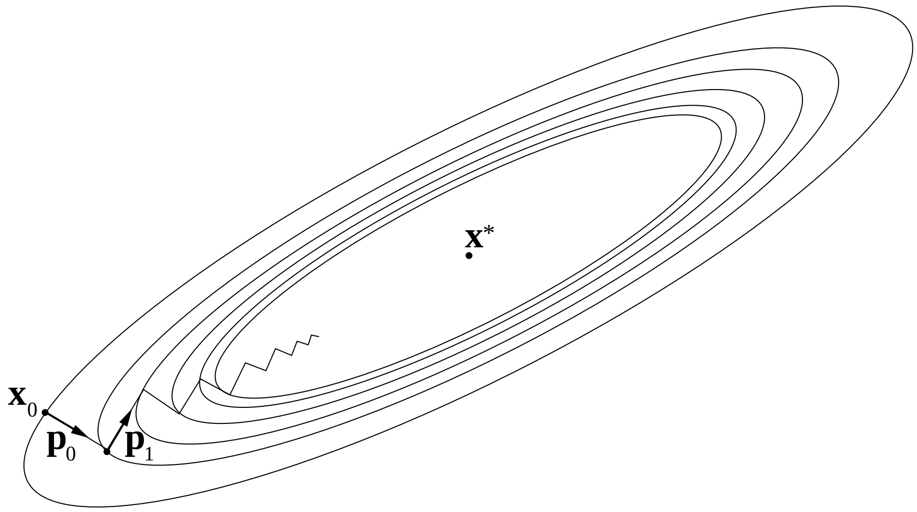

Figure 4.2: Trajectory of steepest descent.

the case in which $\mathbf{x}$ is a two-dimensional vector, as in figure 4.2, which shows the trajectory $\mathbf{x}_k$ superimposed on a set of isocontours of $f(\mathbf{x})$.

There is one good, but very precarious case, namely, when the starting point $\mathbf{x}_0$ is at one apex (tip of either axis) of an isocontour ellipse. In that case, one iteration will lead to the minimum $\mathbf{x}^*$. In all other cases, the line in the direction $\mathbf{p}_k$ of steepest descent, which is orthogonal to the isocontour at $\mathbf{x}_k$, will not pass through $\mathbf{x}^*$. The minimum of f along that line is tangent to some other, lower isocontour. The next step is orthogonal to the latter isocontour (that is, parallel to the gradient). Thus, at every step the steepest descent trajectory is forced to make a ninety-degree turn. If isocontours were circles ($\sigma_1 = \sigma_n$) centered at $\mathbf{x}^*$, then the first turn would make the new direction point to $\mathbf{x}^*$, and minimization would get there in just one more step. This case, in which $\kappa(Q) = 1$, is consistent with our analysis, because then

$$\frac{\sigma_1 - \sigma_n}{\sigma_1 + \sigma_n} = 0 \; .$$

The more elongated the isocontours, that is, the greater the condition number $\kappa(Q)$, the farther away a line orthogonal to an isocontour passes from $\mathbf{x}^*$, and the more steps are required for convergence.

For general (that is, non-quadratic) f, the analysis above applies once $\mathbf{x}_k$ gets close enough to the minimum, so that f is well approximated by a paraboloid. In this case, Q is the matrix of second derivatives of f with respect to $\mathbf{x}$, and is called the *Hessian* of f. In summary, steepest descent is good for functions that have a well conditioned Hessian near the minimum, but can become arbitrarily slow for poorly conditioned Hessians.

To characterize the speed of convergence of different minimization algorithms, we introduce the notion of the *order of convergence*. This is defined as the largest value of q for which the

$$\lim_{k \to \infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^q}$$

is finite. If β is this limit, then close to the solution (that is, for large values of k) we have

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \approx \beta \|\mathbf{x}_k - \mathbf{x}^*\|^q$$

for a minimization method of order q. In other words, the distance of $\mathbf{x}_k$ from $\mathbf{x}^*$ is reduced by the q-th power at every step, so the higher the order of convergence, the better. Theorem 4.1.2 implies that steepest descent has at best a linear order of convergence. In fact, the residuals $|f(\mathbf{x}_k) - f(\mathbf{x}^*)|$ in the *values* of the function being minimized converge linearly. Since the gradient of f approaches zero when $\mathbf{x}_k$ tends to $\mathbf{x}^*$, the *arguments* $\mathbf{x}_k$ to f can converge to $\mathbf{x}^*$ even more slowly.

To complete the steepest descent algorithm we need to specify how to check whether a minimum has been reached. One criterion is to check whether the value of $f(\mathbf{x}_k)$ has significantly decreased from $f(\mathbf{x}_{k-1})$. Another is to check whether $\mathbf{x}_k$ is significantly different from $\mathbf{x}_{k-1}$. Close to the minimum, the derivatives of f are close to zero, so $|f(\mathbf{x}_k) - f(\mathbf{x}_{k-1})|$ may be very small but $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|$ may still be relatively large. Thus, the check on $\mathbf{x}_k$ is more stringent, and therefore preferable in most cases. In fact, usually one is interested in the value of $\mathbf{x}^*$, rather than in that of $f(\mathbf{x}^*)$. In summary, the steepest descent algorithm can be stopped when

$$\|\mathbf{x}_k - \mathbf{x}_{k-1}\| < \epsilon$$

where the positive constant ϵ is provided by the user.

In our analysis of steepest descent, we used the Hessian Q in order to compute the optimal step size α (see equation (4.4)). We used Q because it was available, but its computation during steepest descent would in general be overkill. In fact, only gradient information is necessary to find $\mathbf{p}_k$, and a line search in the direction of $\mathbf{p}_k$ can be used to determine the step size α_k. In contrast, the Hessian of $f(\mathbf{x})$ requires computing $\binom{n}{2}$ second derivatives if $\mathbf{x}$ is an n-dimensional vector.

Using line search to find α_k guarantees that a minimum in the direction $\mathbf{p}_k$ is actually reached even when the parabolic approximation is inadequate. Here is how line search works.

Let

$$h(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{p}_k) \tag{4.13}$$

be the scalar function of one variable that is obtained by restricting the function f to the line through the current point $\mathbf{x}_k$ and in the direction of $\mathbf{p}_k$. Line search first determines two points a, c that bracket the desired minimum α_k, in the sense that $a \le \alpha_k \le c$, and then picks a point between a and c, say, $b = (a+c)/2$. The only difficulty here is to find c. In fact, we can set $a = 0$, corresponding through equation (4.13) to the starting point $\mathbf{x}_k$. A point c that is on the opposite side of the minimum with respect to a can be found by increasing α through values $\alpha_1 = a, \alpha_2, \ldots$ until α_i is greater than α_{i-1}. Then, if we can assume that h is convex between α_1 and α_i, we can set $c = \alpha_i$. In fact, the derivative of h at a is negative, so the function is initially decreasing, but it is increasing between α_{i-1} and $\alpha_i = c$, so the minimum must be somewhere between a and c. Of course, if we cannot assume convexity, we may find the wrong minimum, but there is no general-purpose fix to this problem.

Line search now proceeds by shrinking the bracketing triple (a, b, c) until $c - a$ is smaller than the desired accuracy in determining α_k. Shrinking works as follows:

```
if b − a > c − b
    u = (a + b)/2
    if f(u) > f(b)
        (a, b, c) = (u, b, c)
    otherwise
        (a, b, c) = (a, u, b)
    end
otherwise
    u = (b + c)/2
    if f(u) > f(b)
        (a, b, c) = (a, b, u)
    otherwise
        (a, b, c) = (b, u, c)
    end
end.
```

It is easy to see that in each case the bracketing triple (a, b, c) preserves the property that $f(b) \leq f(a)$ and $f(b) \leq f(c)$, and therefore the minimum is somewhere between a and c. In addition, at every step the interval (a, c) shrinks to $3/4$ of its previous size, so line search will find the minimum in a number of steps that is logarithmic in the desired accuracy.

4.2 Newton's Method

If a function can be well approximated by a paraboloid in the region in which minimization is performed, the analysis in the previous section suggests a straight-forward fix to the slow convergence of steepest descent. In fact, equation (4.2) tells us how to jump in one step from the starting point $\mathbf{x}_0$ to the minimum $\mathbf{x}^*$. Of course, when $f(\mathbf{x})$ is not exactly a paraboloid, the new value $\mathbf{x}_1$ will be different from $\mathbf{x}^*$. Consequently, iterations are needed, but convergence can be expected to be faster. This is the idea of Newton's method, which we now summarize. Let

$$f(\mathbf{x}_k + \Delta\mathbf{x}) \approx f(\mathbf{x}_k) + \mathbf{g}_k^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T Q_k \Delta\mathbf{x}_k \tag{4.14}$$

be the first terms of the Taylor series expansion of f about the current point $\mathbf{x}_k$, where

$$\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k}$$

and

$$Q_k = Q(\mathbf{x}_k) = \left.\frac{\partial^2 f}{\partial \mathbf{x}\partial \mathbf{x}^T}\right|_{\mathbf{x}=\mathbf{x}_k} = \left[\begin{array}{ccc} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{array}\right]_{\mathbf{x}=\mathbf{x}_k}$$

are the gradient and Hessian of f evaluated at the current point $\mathbf{x}_k$. Notice that even when f is a paraboloid, the gradient $\mathbf{g}_k$ is different from $\mathbf{a}$ as used in equation (4.1). In fact, $\mathbf{a}$ and Q are the coefficients of the Taylor expansion of f around point $\mathbf{x} = 0$, while $\mathbf{g}_k$ and Q_k are the coefficients of the Taylor expansion of f around the *current* point $\mathbf{x}_k$. In other words, gradient and Hessian are constantly reevaluated in Newton's method.

To the extent that approximation (4.14) is valid, we can set the derivatives of $f(\mathbf{x}_k + \Delta\mathbf{x})$ with respect to $\Delta\mathbf{x}$ to zero, and obtain, analogously to equation (4.2), the linear system

$$Q_k \Delta\mathbf{x} = -\mathbf{g}_k \; , \tag{4.15}$$

whose solution $\Delta\mathbf{x}_k = \alpha_k \mathbf{p}_k$ yields at the same time the step direction $\mathbf{p}_k = \Delta\mathbf{x}_k / \|\Delta\mathbf{x}_k\|$ and the step size $\alpha_k = \|\Delta\mathbf{x}_k\|$. The direction is of course undefined once the algorithm has reached a minimum, that is, when $\alpha_k = 0$.

A minimization algorithm in which the step direction $\mathbf{p}_k$ and size α_k are defined in this manner is called *Newton's method*. The corresponding $\mathbf{p}_k$ is termed the *Newton direction*, and the step defined by equation (4.15) is the *Newton step*.

The greater speed of Newton's method over steepest descent is borne out by analysis: while steepest descent has a linear order of convergence, Newton's method is quadratic. In fact, let

$$\mathbf{y}(\mathbf{x}) = \mathbf{x} - Q(\mathbf{x})^{-1}\mathbf{g}(\mathbf{x})$$

be the place reached by a Newton step starting at $\mathbf{x}$ (see equation (4.15)), and suppose that at the minimum $\mathbf{x}^*$ the Hessian $Q(\mathbf{x}^*)$ is nonsingular. Then

$$\mathbf{y}(\mathbf{x}^*) = \mathbf{x}^*$$

because $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$, and

$$\mathbf{x}_{k+1} - \mathbf{x}^* = \mathbf{y}(\mathbf{x}_k) - \mathbf{x}^* = \mathbf{y}(\mathbf{x}_k) - \mathbf{y}(\mathbf{x}^*) \ .$$

From the mean-value theorem, we have

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| = \|\mathbf{y}(\mathbf{x}_k) - \mathbf{y}(\mathbf{x}^*)\| \leq \left\| \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T} \right]_{\mathbf{x}=\mathbf{x}^*} (\mathbf{x}_k - \mathbf{x}^*) \right\| + \frac{1}{2} \left| \frac{\partial^2 \mathbf{y}}{\partial \mathbf{x} \partial \mathbf{x}^T} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \|\mathbf{x}_k - \mathbf{x}^*\|^2$$

where $\hat{\mathbf{x}}$ is some point on the line between $\mathbf{x}^*$ and $\mathbf{x}_k$. Since $\mathbf{y}(\mathbf{x}^*) = \mathbf{x}^*$, the first derivatives of $\mathbf{y}$ at $\mathbf{x}^*$ are zero, so that the first term in the right-hand side above vanishes, and

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq c \, \|\mathbf{x}_k - \mathbf{x}^*\|^2$$

where c depends on third-order derivatives of f near $\mathbf{x}^*$. Thus, the convergence rate of Newton's method is of order at least two.

For a quadratic function, as in equation (4.1), steepest descent takes many steps to converge, while Newton's method reaches the minimum in one step. However, this single iteration in Newton's method is more expensive, because it requires both the gradient $\mathbf{g}_k$ and the Hessian Q_k to be evaluated, for a total of $n + \binom{n}{2}$ derivatives. In addition, the Hessian must be inverted, or, at least, system (4.15) must be solved. For very large problems, in which the dimension n of $\mathbf{x}$ is thousands or more, storing and manipulating a Hessian can be prohibitive. In contrast, steepest descent requires the gradient $\mathbf{g}_k$ for selecting the step direction $\mathbf{p}_k$, and a line search in the direction $\mathbf{p}_k$ to find the step size. The method of conjugate gradients, discussed in the next section, is motivated by the desire to accelerate convergence with respect to the steepest descent method, but without paying the storage cost of Newton's method.

4.3 Conjugate Gradients

Newton's method converges faster (quadratically) than steepest descent (linear convergence rate) because it uses more information about the function f being minimized. Steepest descent locally approximates the function with planes, because it only uses gradient information. All it can do is to go downhill. Newton's method approximates f with paraboloids, and then jumps at every iteration to the lowest point of the current approximation. The bottom line is that fast convergence requires work that is equivalent to evaluating the Hessian of f.

Prima facie, the method of conjugate gradients discussed in this section seems to violate this principle: it achieves fast, superlinear convergence, similarly to Newton's method, but it only requires gradient information. This paradox, however, is only apparent. Conjugate gradients works by taking n steps for each of the steps in Newton's method. It effectively solves the linear system (4.2) of Newton's method, but it does so by a sequence of n one-dimensional minimizations, each requiring one gradient computation and one line search.

Overall, the work done by conjugate gradients is equivalent to that done by Newton's method. However, system (4.2) is never constructed explicitly, and the matrix Q is never stored. This is very important in cases where $\mathbf{x}$ has thousands or even millions of components. These high-dimensional problems arise typically from the discretization of partial differential equations. Say for instance that we want to compute the motion of points in an image as a consequence of camera motion. Partial differential equations relate image intensities over space and time to the motion of the underlying image features. At every pixel in the image, this motion, called the *motion field*, is represented by a vector whose magnitude and direction describe the velocity of the image feature at that pixel. Thus, if an image has, say, a quarter of a million pixels, there are $n = 500,000$ unknown motion field values. Storing and inverting a $500,000 \times 500,000$ Hessian is out of the question. In cases like these, conjugate gradients saves the day.

The conjugate gradients method described in these notes is the so-called Polak-Ribière variation. It will be introduced in three steps. First, it will be developed for the simple case of minimizing a quadratic function with positive-definite and known Hessian. This quadratic function $f(\mathbf{x})$ was introduced in equation (4.1).

We know that in this case minimizing $f(\mathbf{x})$ is equivalent to solving the linear system (4.2). Rather than an iterative method, conjugate gradients is a direct method for the quadratic case. This means that the number of iterations is fixed. Specifically, the method converges to the solution in n steps, where n is the number of components of $\mathbf{x}$. Because of the equivalence with a linear system, conjugate gradients for the quadratic case can also be seen as an alternative method for solving a linear system, although the version presented here will only work if the matrix of the system is symmetric and positive definite.

Second, the assumption that the Hessian Q in expression (4.1) is known will be removed. As discussed above, this is the main reason for using conjugate gradients.

Third, the conjugate gradients method will be extended to general functions $f(\mathbf{x})$. In this case, the method is no longer direct, but iterative, and the cost of finding the minimum depends on the desired accuracy. This occurs because the Hessian of f is no longer a constant, as it was in the quadratic case. As a consequence, a certain property that holds in the quadratic case is now valid only approximately. In spite of this, the convergence rate of conjugate gradients is superlinear, somewhere between Newton's method and steepest descent. Finding tight bounds for the convergence rate of conjugate gradients is hard, and we will omit this proof. We rely instead on the intuition that conjugate gradients solves system (4.2), and that the quadratic approximation becomes more and more valid as the algorithm converges to the minimum. If the function f starts to behave like a quadratic function early, that is, if f is nearly quadratic in a large neighborhood of the minimum, convergence is fast, as it requires close to the n steps that are necessary in the quadratic case, and each of the steps is simple. This combination of fast convergence, modest storage requirements, and low computational cost per iteration explains the popularity of conjugate gradients methods for the optimization of functions of a large number of variables.

4.3.1 The Quadratic Case

Suppose that we want to minimize the quadratic function

$$f(\mathbf{x}) = c + \mathbf{a}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T Q\mathbf{x} \tag{4.16}$$

where Q is a symmetric, positive definite matrix, and $\mathbf{x}$ has n components. As we saw in our discussion of steepest descent, the minimum $\mathbf{x}^*$ is the solution to the linear system

$$Q\mathbf{x} = -\mathbf{a}\ . \tag{4.17}$$

We know how to solve such a system. However, all the methods we have seen so far involve explicit manipulation of the matrix Q. We now consider an alternative solution method that does not need Q, but only the quantity

$$\mathbf{g}_k = Q\mathbf{x}_k + \mathbf{a}$$

that is, the gradient of $f(\mathbf{x})$, evaluated at n different points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. We will see that the conjugate gradients method requires n gradient evaluations and n line searches *in lieu* of each $n \times n$ matrix inversion in Newton's method.

Formal proofs can be found in Elijah Polak, *Optimization — Algorithms and consistent approximations*, Springer, NY, 1997. The arguments offered below appeal to intuition.

Consider the case $n = 3$, in which the variable $\mathbf{x}$ in $f(\mathbf{x})$ is a three-dimensional vector. Then the quadratic function $f(\mathbf{x})$ is constant over ellipsoids, called *isosurfaces*, centered at the minimum $\mathbf{x}^*$. How can we start from a point $\mathbf{x}_0$ on one of these ellipsoids and reach $\mathbf{x}^*$ by a finite sequence of one-dimensional searches? In connection with steepest descent, we noticed that for poorly conditioned Hessians orthogonal directions lead to many small steps, that is, to slow convergence.

When the ellipsoids are spheres, on the other hand, this works much better. The first step takes from $\mathbf{x}_0$ to $\mathbf{x}_1$, and the line between $\mathbf{x}_0$ and $\mathbf{x}_1$ is tangent to an isosurface at $\mathbf{x}_1$. The next step is in the direction of the gradient, so that the new direction $\mathbf{p}_1$ is orthogonal to the previous direction $\mathbf{p}_0$. This would then take us to $\mathbf{x}^*$ right away. Suppose however that we cannot afford to compute this special direction $\mathbf{p}_1$ orthogonal to $\mathbf{p}_0$, but that we can only compute *some* direction $\mathbf{p}_1$ orthogonal to $\mathbf{p}_0$ (there is an $n-1$-dimensional space of such directions!). It is easy to see that in that case n steps will take us to $\mathbf{x}^*$. In fact, since isosurfaces are spheres, each line minimization is independent of the others: The first step yields the minimum in the space spanned by $\mathbf{p}_0$, the second step then yields the minimum in the space spanned by $\mathbf{p}_0$ and $\mathbf{p}_1$, and so forth. After n steps we must be done, since $\mathbf{p}_0 \ldots, \mathbf{p}_{n-1}$ span the whole space.

In summary, any set of orthogonal directions, with a line search in each direction, will lead to the minimum for spherical isosurfaces. Given an arbitrary set of ellipsoidal isosurfaces, there is a one-to-one mapping with a spherical system: if $Q = U\Sigma U^T$ is the SVD of the symmetric, positive definite matrix Q, then we can write

$$\frac{1}{2}\mathbf{x}^T Q\mathbf{x} = \frac{1}{2}\mathbf{y}^T\mathbf{y}$$

where

$$\mathbf{y} = \Sigma^{1/2}U^T\mathbf{x} \; . \tag{4.18}$$

Consequently, there must be a condition for the original problem (in terms of Q) that is equivalent to orthogonality for the spherical problem. If two directions $\mathbf{q}_i$ and $\mathbf{q}_j$ are orthogonal in the spherical context, that is, if

$$\mathbf{q}_i^T\mathbf{q}_j = 0 \; ,$$

what does this translate into in terms of the directions $\mathbf{p}_i$ and $\mathbf{p}_j$ for the ellipsoidal problem? We have

$$\mathbf{q}_{i,j} = \Sigma^{1/2}U^T\mathbf{p}_{i,j} \; ,$$

so that orthogonality for $\mathbf{q}_{i,j}$ becomes

$$\mathbf{p}_i^T U\Sigma^{1/2}\Sigma^{1/2}U^T\mathbf{p}_j = 0$$

or

$$\mathbf{p}_i^T Q\mathbf{p}_j = 0 \; . \tag{4.19}$$

This condition is called *Q-conjugacy*, or *Q-orthogonality*: if equation (4.19) holds, then $\mathbf{p}_i$ and $\mathbf{p}_j$ are said to be Q-conjugate or Q-orthogonal to each other. We will henceforth simply say "conjugate" for brevity.

In summary, if we can find n directions $\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}$ that are mutually conjugate, and if we do line minimization along each direction $\mathbf{p}_k$, we reach the minimum in at most n steps. Of course, we cannot use the transformation (4.18) in the algorithm, because Σ and especially U^T are too large. So now we need to find a method for generating n conjugate directions without using either Q or its SVD. We do this in two steps. First, we find conjugate directions whose definitions do involve Q. Then, in the next subsection, we rewrite these expressions without Q.

Here is the procedure, due to Hestenes and Stiefel (*Methods of conjugate gradients for solving linear systems*, J. Res. Bureau National Standards, section B, Vol 49, pp. 409-436, 1952), which also incorporates the steps from $\mathbf{x}_0$ to $\mathbf{x}_n$:

$\mathbf{g}_0 = \mathbf{g}(\mathbf{x}_0)$
$\mathbf{p}_0 = -\mathbf{g}_0$
for $k = 0\ldots, n-1$
    $\alpha_k = \arg\min_{\alpha\geq 0} f(\mathbf{x}_k + \alpha\mathbf{p}_k)$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$
    $\mathbf{g}_{k+1} = \mathbf{g}(\mathbf{x}_{k+1})$
    $\gamma_k = \frac{\mathbf{g}_{k+1}^T Q\mathbf{p}_k}{\mathbf{p}_k^T Q\mathbf{p}_k}$
    $\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \gamma_k\mathbf{p}_k$
end

where

$$\mathbf{g}_k = \mathbf{g}(\mathbf{x}_k) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_k}$$

is the gradient of f at $\mathbf{x}_k$.

It is simple to see that $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$ are conjugate. In fact,

$$\begin{aligned}
\mathbf{p}_k^T Q \mathbf{p}_{k+1} &= \mathbf{p}_k^T Q(-\mathbf{g}_{k+1} + \gamma_k \mathbf{p}_k) \\
&= -\mathbf{p}_k^T Q \mathbf{g}_{k+1} + \frac{\mathbf{g}_{k+1}^T Q \mathbf{p}_k}{\mathbf{p}_k^T Q \mathbf{p}_k} \mathbf{p}_k^T Q \mathbf{p}_k \\
&= -\mathbf{p}_k^T Q \mathbf{g}_{k+1} + \mathbf{g}_{k+1}^T Q \mathbf{p}_k = 0 \; .
\end{aligned}$$

It is somewhat more cumbersome to show that $\mathbf{p}_i$ and $\mathbf{p}_{k+1}$ for $i = 0, \ldots, k$ are also conjugate. This can be done by induction. The proof is based on the observation that the vectors $\mathbf{p}_k$ are found by a generalization of Gram-Schmidt (theorem 2.4.2) to produce conjugate rather than orthogonal vectors. Details can be found in Polak's book mentioned earlier.

4.3.2 Removing the Hessian

The algorithm shown in the previous subsection is a correct conjugate gradients algorithm. However, it is computationally inadequate because the expression for γ_k contains the Hessian Q, which is too large. We now show that γ_k can be rewritten in terms of the gradient values $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$ only. To this end, we notice that

$$\mathbf{g}_{k+1} = \mathbf{g}_k + \alpha_k Q \mathbf{p}_k \; ,$$

or

$$\alpha_k Q \mathbf{p}_k = \mathbf{g}_{k+1} - \mathbf{g}_k \; .$$

In fact,

$$\mathbf{g}(\mathbf{x}) = \mathbf{a} + Q\mathbf{x}$$

so that

$$\mathbf{g}_{k+1} = \mathbf{g}(\mathbf{x}_{k+1}) = \mathbf{g}(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \mathbf{a} + Q(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = \mathbf{g}_k + \alpha_k Q \mathbf{p}_k \; .$$

We can therefore write

$$\gamma_k = \frac{\mathbf{g}_{k+1}^T Q \mathbf{p}_k}{\mathbf{p}_k^T Q \mathbf{p}_k} = \frac{\mathbf{g}_{k+1}^T \alpha_k Q \mathbf{p}_k}{\mathbf{p}_k^T \alpha_k Q \mathbf{p}_k} = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{p}_k^T (\mathbf{g}_{k+1} - \mathbf{g}_k)} \; ,$$

and Q has disappeared.

This expression for γ_k can be further simplified by noticing that

$$\mathbf{p}_k^T \mathbf{g}_{k+1} = 0$$

because the line along $\mathbf{p}_k$ is tangent to an isosurface at $\mathbf{x}_{k+1}$, while the gradient $\mathbf{g}_{k+1}$ is orthogonal to the isosurface at $\mathbf{x}_{k+1}$. Similarly,

$$\mathbf{p}_{k-1}^T \mathbf{g}_k = 0 \; .$$

Then, the denominator of γ_k becomes

$$\mathbf{p}_k^T (\mathbf{g}_{k+1} - \mathbf{g}_k) = -\mathbf{p}_k^T \mathbf{g}_k = (\mathbf{g}_k - \gamma_{k-1} \mathbf{p}_{k-1})^T \mathbf{g}_k = \mathbf{g}_k^T \mathbf{g}_k \; .$$

In conclusion, we obtain the *Polak-Ribière formula*

$$\gamma_k = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^T \mathbf{g}_k} \; .$$

4.3.3 Extension to General Functions

We now know how to minimize the quadratic function (4.16) in n steps, without ever constructing the Hessian explicitly. When the function $f(\mathbf{x})$ is arbitrary, the same algorithm can be used.

However, n iterations will not suffice. In fact, the Hessian, which was constant for the quadratic case, now is a function of $\mathbf{x}_k$. Strictly speaking, we then lose conjugacy, since $\mathbf{p}_k$ and $\mathbf{p}_{k+1}$ are associated to different Hessians. However, as the algorithm approaches the minimum $\mathbf{x}^*$, the quadratic approximation becomes more and more valid, and a few cycles of n iterations each will achieve convergence.

Chapter 5

Eigenvalues and Eigenvectors

Given a linear transformation

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

the singular value decomposition $A = U\Sigma V^T$ of A transforms the domain of the transformation via the matrix V^T and its range via the matrix U^T so that the transformed system is diagonal. In fact, the equation $\mathbf{b} = U\Sigma V^T\mathbf{x}$ can be written as follows

$$U^T\mathbf{b} = \Sigma V^T\mathbf{x} ,$$

that is,

$$\mathbf{c} = \Sigma\mathbf{y}$$

where

$$\mathbf{y} = V^T\mathbf{x} \quad \text{and} \quad \mathbf{c} = U^T\mathbf{b} ,$$

and where Σ is diagonal. This is a fundamental transformation to use whenever the domain and the range of A are separate spaces. Often, however, domain and range are intimately related to one another even independently of the transformation A. The most important example is perhaps that of a system of linear differential equations, of the form

$$\dot{\mathbf{x}} = A\mathbf{x}$$

where A is $n \times n$. For this equation, the fact that A is square is not a coincidence. In fact, $\mathbf{x}$ is assumed to be a function of some real scalar variable t (often time), and $\dot{\mathbf{x}}$ is the derivative of $\mathbf{x}$ with respect to t:

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} .$$

In other words, there is an intimate, pre-existing relation between $\mathbf{x}$ and $\dot{\mathbf{x}}$, and one cannot change coordinates for $\mathbf{x}$ without also changing those for $\dot{\mathbf{x}}$ accordingly. In fact, if V is an orthogonal matrix and we define

$$\mathbf{y} = V^T\mathbf{x} ,$$

then the definition of $\dot{\mathbf{x}}$ forces us to transform $\dot{\mathbf{x}}$ by V^T as well:

$$\frac{d\, V^T\mathbf{x}}{dt} = V^T\frac{d\mathbf{x}}{dt} = V^T\dot{\mathbf{x}} .$$

In brief, the SVD does nothing useful for systems of linear differential equations, because it diagonalizes A by two different transformations, one for the domain and one for the range, while we need a single transformation. Ideally, we would like to find an orthogonal matrix S and a diagonal matrix Λ such that

$$A = S\Lambda S^T \tag{5.1}$$

so that if we define

$$\mathbf{y} = S^T\mathbf{x}$$

we can write the equivalent but *diagonal* differential system

$$\dot{\mathbf{y}} = \Lambda\mathbf{y} \ .$$

This is now much easier to handle, because it is a system of n independent, scalar differential equations, which can be solved separately. The solutions can then be recombined through

$$\mathbf{x} = S\mathbf{y} \ .$$

We will see all of this in greater detail soon.

Unfortunately, writing A in the form (5.1) is not always possible. This stands to reason, because now we are imposing stronger constraints on the terms of the decomposition. It is like doing an SVD but with the additional constraint $U = V$. If we refer back to figure 3.1, now the circle and the ellipse live in the same space, and the constraint $U = V$ implies that the vectors $\mathbf{v}_i$ on the circle that map into the axes $\sigma_i\mathbf{u}_i$ of the ellipse are parallel to the axes themselves. This will only occur for very special matrices.

In order to make a decomposition like (5.1) possible, we weaken the constraints in several ways:

- the elements of S and Λ are allowed to be complex, rather than real;
- the elements on the diagonal of Λ are allowed to be negative; in fact, they can be even non-real;
- S is required to be only invertible, rather than orthogonal.

To distinguish invertible from orthogonal matrices we use the symbol Q for invertible and S for orthogonal. In some cases, it will be possible to diagonalize A by orthogonal transformations S and S^T. Finally, for complex matrices we generalize the notion of transpose by introducing the *Hermitian* operator: The matrix Q^H (pronounced "Q Hermitian") is defined to be the complex conjugate of the transpose of Q. If Q happens to be real, conjugate transposition becomes simply transposition, so the Hermitian is a generalization of the transpose. A matrix S is said to be *unitary* if

$$S^H S = SS^H = I \ ,$$

so unitary generalizes orthogonal for complex matrices. Unitary matrices merely rotate or flip vectors, in the sense that they do not alter the vectors' norms. For complex vectors, the norm squared is defined as

$$\|\mathbf{x}\|^2 = \mathbf{x}^H\mathbf{x} \ ,$$

and if S is unitary we have

$$\|S\mathbf{x}\|^2 = \mathbf{x}^H S^H S\mathbf{x} = \mathbf{x}^H\mathbf{x} = \|\mathbf{x}\|^2 \ .$$

Furthermore, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are mutually *orthogonal*, in the sense that

$$\mathbf{x}_1^H\mathbf{x}_2 = 0 \ ,$$

then $S\mathbf{x}_1$ and $S\mathbf{x}_2$ are orthogonal as well:

$$\mathbf{x}_1^H S^H S\mathbf{x}_2 = \mathbf{x}_1^H\mathbf{x}_2 = 0 \ .$$

In contrast, a nonunitary transformation Q can change the norms of vectors, as well as the inner products between vectors. A matrix that is equal to its Hermitian is called a Hermitian matrix.

In summary, in order to diagonalize a square matrix A from a system of linear differential equations we generally look for a decomposition of A of the form

$$A = Q\Lambda Q^{-1} \tag{5.2}$$

where Q and Λ are complex, Q is invertible, and Λ is diagonal. For some special matrices, this may specialize to

$$A = S\Lambda S^H$$

with unitary S.

Whenever two matrices A and B, diagonal or not, are related by

$$A = QBQ^{-1} ,$$

they are said to be *similar* to each other, and the transformation of B into A (and vice versa) is called a *similarity* transformation.

The equation $A = Q\Lambda Q^{-1}$ can be rewritten as follows:

$$AQ = Q\Lambda$$

or separately for every column of Q as follows:

$$A\mathbf{q}_i = \lambda_i \mathbf{q}_i \tag{5.3}$$

where

$$Q = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{bmatrix} \quad \text{and} \quad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) .$$

Thus, the columns of $\mathbf{q}_i$ of Q and the diagonal entries λ_i of Λ are solutions of the *eigenvalue/eigenvector* equation

$$A\mathbf{x} = \lambda\mathbf{x} , \tag{5.4}$$

which is how eigenvalues and eigenvectors are usually introduced. In contrast, we have *derived* this equation from the requirement of diagonalizing a matrix by a similarity transformation. The columns of Q are called *eigenvectors*, and the diagonal entries of Λ are called eigenvalues.

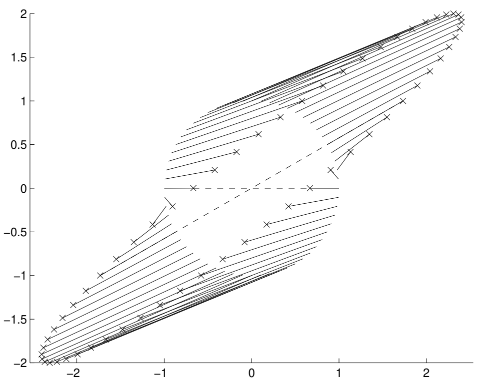

Figure 5.1: Effect of the transformation (5.5) on a sample of points on the unit circle. The dashed lines are vectors that do not change direction under the transformation.

That real eigenvectors and eigenvalues do not always exist can be clarified by considering the eigenvalue problem from a geometrical point of view in the $n = 2$ case. As we know, an invertible linear transformation transforms the unit circle into an ellipse. Each point on the unit circle is transformed into some point on the ellipse. Figure 5.1 shows the effect of the transformation represented by the matrix

$$A = \begin{bmatrix} 2/3 & 4/\sqrt{3} \\ 0 & 2 \end{bmatrix} \tag{5.5}$$

for a sample of points on the unit circle. Notice that there are many transformations that map the unit circle into the same ellipse. In fact, the circle in figure 5.1 can be rotated, pulling the solid lines along. Each rotation yields another matrix A, but the resulting ellipse is unchanged. In other words, the curve-to-curve transformation from circle to ellipse is unique, but the point-to-point transformation is not. Matrices represent point-to-point transformations.

The eigenvalue problem amounts to finding axes $\mathbf{q}_1, \mathbf{q}_2$ that are mapped into themselves by the original transformation A (see equation (5.3)). In figure 5.1, the two eigenvectors are shown as dashed lines. Notice that they do not correspond to the axes of the ellipse, and that they are not orthogonal. Equation (5.4) is homogeneous in $\mathbf{x}$, so $\mathbf{x}$ can be assumed to be a unit vector without loss of generality.

Given that the directions of the input vectors are generally changed by the transformation A, as evident from figure 5.1, it is not obvious whether the eigenvalue problem admits a solution at all. We will see that the answer depends on the matrix A, and that a rather diverse array of situations may arise. In some cases, the eigenvalues and their eigenvectors exist, but they are complex. The geometric intuition is hidden, and the problem is best treated as an algebraic one. In other cases, all eigenvalues exist, perhaps all real, but not enough eigenvectors can be found, and the matrix A cannot be diagonalized. In particularly good cases, there are n real, orthonormal eigenvectors. In bad cases, we have to give up the idea of diagonalizing A, and we can only triangularize it. This turns out to be good enough for solving linear differential systems, just as triangularization was sufficient for solving linear algebraic systems.

5.1 Computing Eigenvalues and Eigenvectors Algebraically

Let us rewrite the eigenvalue equation

$$A\mathbf{x} = \lambda\mathbf{x}$$

as follows:

$$(A - \lambda I)\mathbf{x} = 0 \; . \tag{5.6}$$

This is a homogeneous, square system of equations, which admits nontrivial solutions iff the matrix $A - \lambda I$ is rank-deficient. A square matrix B is rank-deficient iff its *determinant*,

$$\det(B) = \begin{cases} b_{11} & \text{if } B \text{ is } 1 \times 1 \\ \sum_{i=1}^{n} (-1)^{i+1} b_{i1} \det(B_{i1}) & \text{otherwise} \end{cases}$$

is zero. In this expression, B_{ij} is the *algebraic complement* of entry b_{ij}, defined as the $(n-1) \times (n-1)$ matrix obtained by removing row i and column j from B.

Volumes have been written about the properties of the determinant. For our purposes, it is sufficient to recall the following properties from linear algebra:

- $\det(B) = \det(B^T)$;
- $\det([\ \mathbf{b}_1 \ \cdots \ \mathbf{b}_n\ ]) = 0$ iff $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are linearly dependent;
- $\det([\ \mathbf{b}_1 \ \cdots \ \mathbf{b}_i \ \cdots \ \mathbf{b}_j \ \cdots \ \mathbf{b}_n\ ]) = -\det([\ \mathbf{b}_1 \ \cdots \ \mathbf{b}_j \ \cdots \ \mathbf{b}_i \ \cdots \ \mathbf{b}_n\ ])$;
- $\det(BC) = \det(B)\det(C)$.

Thus, for system (5.6) to admit nontrivial solutions, we need

$$\det(A - \lambda I) = 0 \; . \tag{5.7}$$

From the definition of determinant, it follows, by very simple induction, that the left-hand side of equation (5.7) is a polynomial of degree n in λ, and that the coefficient of λ^n is 1. Therefore, equation (5.7), which is called the *characteristic equation* of A, has n complex solutions, in the sense that

$$\det(A - \lambda I) = (-1)^n (\lambda - \lambda_1) \cdot \ldots \cdot (\lambda - \lambda_n)$$

where some of the λ_i may coincide. In other words, an $n \times n$ matrix has at most n distinct eigenvalues. The case of exactly n distinct eigenvalues is of particular interest, because of the following results.

Theorem 5.1.1 *Eigenvectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$ corresponding to distinct eigenvalues $\lambda_1, \ldots, \lambda_k$ are linearly independent.*

Proof. Suppose that $c_1\mathbf{x}_1 + \ldots + c_k\mathbf{x}_k = 0$ where the $\mathbf{x}_i$ are eigenvectors of a matrix A. We need to show that then $c_1 = \ldots = c_k = 0$. By multiplying by A we obtain

$$c_1 A\mathbf{x}_1 + \ldots + c_k A\mathbf{x}_k = 0$$

and because $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are eigenvectors corresponding to eigenvalues $\lambda_1, \ldots, \lambda_k$, we have

$$c_1\lambda_1\mathbf{x}_1 + \ldots + c_k\lambda_k\mathbf{x}_k = 0 \ . \tag{5.8}$$

However, from

$$c_1\mathbf{x}_1 + \ldots + c_k\mathbf{x}_k = 0$$

we also have

$$c_1\lambda_k\mathbf{x}_1 + \ldots + c_k\lambda_k\mathbf{x}_k = 0$$

and subtracting this equation from equation (5.8) we have

$$c_1(\lambda_1 - \lambda_k)\mathbf{x}_1 + \ldots + c_{k-1}(\lambda_{k-1} - \lambda_k)\mathbf{x}_{k-1} = 0 \ .$$

Thus, we have reduced the summation to one containing $k-1$ terms. Since all λ_i are distinct, the differences in parentheses are all nonzero, and we can replace each $\mathbf{x}_i$ by $\mathbf{x}'_i = (\lambda_i - \lambda_k)\mathbf{x}_i$, which is still an eigenvector of A:

$$c_1\mathbf{x}'_1 + \ldots + c_{k-1}\mathbf{x}'_k = 0 \ .$$

We can repeat this procedure until only one term remains, and this forces $c_1 = 0$, so that

$$c_2\mathbf{x}_2 + \ldots + c_k\mathbf{x}_k = 0$$

This entire argument can be repeated for the last equation, therefore forcing $c_2 = 0$, and so forth.

In summary, the equation $c_1\mathbf{x}_1 + \ldots + c_k\mathbf{x}_k = 0$ implies that $c_1 = \ldots = c_k = 0$, that is, that the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are linearly independent. Δ

For Hermitian matrices (and therefore for real symmetric matrices as well), the situation is even better.

Theorem 5.1.2 *A Hermitian matrix has real eigenvalues.*

Proof. A matrix A is Hermitian iff $A = A^H$. Let λ and $\mathbf{x}$ be an eigenvalue of A and a corresponding eigenvector:

$$A\mathbf{x} = \lambda\mathbf{x} \ . \tag{5.9}$$

By taking the Hermitian we obtain

$$\mathbf{x}^H A^H = \lambda^* \mathbf{x}^H \ .$$

Since $A = A^H$, the last equation can be rewritten as follows:

$$\mathbf{x}^H A = \lambda^* \mathbf{x}^H \ . \tag{5.10}$$

If we multiply equation (5.9) from the left by $\mathbf{x}^H$ and equation (5.10) from the right by $\mathbf{x}$, we obtain

$$\begin{aligned} \mathbf{x}^H A\mathbf{x} &= \lambda\mathbf{x}^H\mathbf{x} \\ \mathbf{x}^H A\mathbf{x} &= \lambda^*\mathbf{x}^H\mathbf{x} \end{aligned}$$

which implies that

$$\lambda \mathbf{x}^H \mathbf{x} = \lambda^* \mathbf{x}^H \mathbf{x}$$

Since $\mathbf{x}$ is an eigenvector, the scalar $\mathbf{x}^H \mathbf{x}$ is nonzero, so that we have

$$\lambda = \lambda^*$$

as promised. Δ

Corollary 5.1.3 *A real and symmetric matrix has real eigenvalues.*

Proof. A real and symmetric matrix is Hermitian. Δ

Theorem 5.1.4 *Eigenvectors corresponding to distinct eigenvalues of a Hermitian matrix are mutually orthogonal.*

Proof. Let λ and μ be two distinct eigenvalues of A, and let $\mathbf{x}$ and $\mathbf{y}$ be corresponding eigenvectors:

$$\begin{aligned} A\mathbf{x} &= \lambda \mathbf{x} \\ A\mathbf{y} &= \mu \mathbf{y} \;\Rightarrow\; \mathbf{y}^H A = \mu \mathbf{y}^H \end{aligned}$$

because $A = A^H$ and from theorem 5.1.2 $\mu = \mu^*$. If we multiply these two equations by $\mathbf{y}^H$ from the left and $\mathbf{x}$ from the right, respectively, we obtain

$$\begin{aligned} \mathbf{y}^H A\mathbf{x} &= \lambda \mathbf{y}^H \mathbf{x} \\ \mathbf{y}^H A\mathbf{x} &= \mu \mathbf{y}^H \mathbf{x} \,, \end{aligned}$$

which implies

$$\lambda \mathbf{y}^H \mathbf{x} = \mu \mathbf{y}^H \mathbf{x}$$

or

$$(\lambda - \mu)\mathbf{y}^H \mathbf{x} = 0 \,.$$

Since the two eigenvalues are distinct, $\lambda - \mu$ is nonzero, and we must have $\mathbf{y}^H \mathbf{x} = 0$. Δ

Corollary 5.1.5 *An $n \times n$ Hermitian matrix with n distinct eigenvalues admits n orthonormal eigenvectors.*

Proof. From theorem 5.1.4, the eigenvectors of an $n \times n$ Hermitian matrix with n distinct eigenvalues are all mutually orthogonal. Since the eigenvalue equation $A\mathbf{x} = \lambda \mathbf{x}$ is homogeneous in $\mathbf{x}$, the vector $\mathbf{x}$ can be normalized without violating the equation. Consequently, the eigenvectors can be made to be orthonormal. Δ

In summary, any square matrix with n distinct eigenvalues can be diagonalized by a similarity transformation, and any square Hermitian matrix with n distinct eigenvalues can be diagonalized by a unitary similarity transformation.

Notice that the converse is not true: a matrix can have coincident eigenvalues and still admit n independent, and even orthonormal, eigenvectors. For instance, the $n \times n$ identity matrix has n equal eigenvalues but n orthonormal eigenvectors (which can be chosen in infinitely many ways).

The examples in section 5.2 show that when some eigenvalues coincide, rather diverse situations can arise concerning the eigenvectors. First, however, we point out a simple but fundamental fact about the eigenvalues of a triangular matrix.

Theorem 5.1.6 *The determinant of a triangular matrix is the product of the elements on its diagonal.*

Proof. This follows immediately from the definition of determinant. Without loss of generality, we can assume a triangular matrix B to be upper-triangular, for otherwise we can repeat the argument for the transpose, which because of the properties above has the same eigenvalues. Then, the only possibly nonzero b_{i1} of the matrix B is b_{11}, and the summation in the definition of determinant given above reduces to a single term:

$$\det(B) = \begin{cases} b_{11} & \text{if } B \text{ is } 1 \times 1 \\ b_{11} \det(B_{11}) & \text{otherwise} \end{cases} .$$

By repeating the argument for B_{11} and so forth until we are left with a single scalar, we obtain

$$\det(B) = b_{11} \cdot \ldots \cdot b_{nn} .$$

Δ

Corollary 5.1.7 *The eigenvalues of a triangular matrix are the elements on its diagonal.*

Proof. The eigenvalues of a matrix A are the solutions of the equation

$$\det(A - \lambda I) = 0 .$$

If A is triangular, so is $B = A - \lambda I$, and from the previous theorem we obtain

$$\det(A - \lambda I) = (a_{11} - \lambda) \cdot \ldots \cdot (a_{nn} - \lambda)$$

which is equal to zero for

$$\lambda = a_{11}, \ldots, a_{nn} .$$

Δ

Note that diagonal matrices are triangular, so this result holds for diagonal matrices as well.

5.2 Good and Bad Matrices

Solving differential equations becomes much easier when matrices have a full set of orthonormal eigenvectors. For instance, the matrix

$$A = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.11}$$

has eigenvalues 2 and 1 and eigenvectors

$$\mathbf{s}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{s}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} .$$

Matrices with n orthonormal eigenvectors are called *normal*. Normal matrices are good news, because then the $n \times n$ system of differential equations

$$\dot{\mathbf{x}} = A\mathbf{x}$$

has solution

$$\mathbf{x}(t) = \sum_{i=1}^{n} c_i \mathbf{s}_i \, e^{\lambda_i t} = S \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_n t} \end{bmatrix} \mathbf{c}$$

where $S = [\mathbf{s}_1 \cdots \mathbf{s}_n]$ are the eigenvectors, λ_i are the eigenvalues, and the vector $\mathbf{c}$ of constants c_i is

$$\mathbf{c} = S^H \mathbf{x}(0) .$$

More compactly,

$$\mathbf{x}(t) = S \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_n t} \end{bmatrix} S^H \mathbf{x}(0) .$$

Fortunately these matrices occur frequently in practice. However, not all matrices are as good as these. First, there may still be a complete set of n eigenvectors, but they may not be orthonormal. An example of such a matrix is

$$\begin{bmatrix} 2 & -1 \\ 0 & 1 \end{bmatrix}$$

which has eigenvalues 2 and 1 and nonorthogonal eigenvectors

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \mathbf{q}_2 = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} .$$

This is conceptually only a slight problem, because the unitary matrix S is replaced by an invertible matrix Q, and the solution becomes

$$\mathbf{x}(t) = Q \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_n t} \end{bmatrix} Q^{-1} \mathbf{x}(0) .$$

Computationally this is more expensive, because a computation of a Hermitian is replaced by a matrix inversion.

However, things can be worse yet, and a full set of eigenvectors may fail to exist, as we now show.

A necessary condition for an $n \times n$ matrix to be *defective*, that is, to have fewer than n eigenvectors, is that it have repeated eigenvalues. In fact, we have seen (theorem 5.1.1) that a matrix with distinct eigenvalues (zero or nonzero does not matter) has a full set of eigenvectors (perhaps nonorthogonal, but independent). The simplest example of a defective matrix is

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

which has double eigenvalue 0 and only eigenvector $[1\,0]^T$, while

$$\begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix}$$

has double eigenvalue 3 and only eigenvector $[1\,0]^T$, so zero eigenvalues are not the problem.

However, repeated eigenvalues are not a sufficient condition for defectiveness, as the identity matrix proves.

How bad can a matrix be? Here is a matrix that is singular, has fewer than n eigenvectors, and the eigenvectors it has are not orthogonal. It belongs to the scum of all matrices:

$$A = \begin{bmatrix} 0 & 2 & -1 \\ 0 & 2 & 1 \\ 0 & 0 & 2 \end{bmatrix} .$$

Its eigenvalues are 0, because the matrix is singular, and 2, repeated twice. A has to have a repeated eigenvalue if it is to be defective. Its two eigenvectors are

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} , \qquad \mathbf{q}_2 = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

corresponding to eigenvalues 0 and 2 in this order, and there is no $\mathbf{q}_3$. Furthermore, $\mathbf{q}_1$ and $\mathbf{q}_2$ are not orthogonal to each other.

5.3 Computing Eigenvalues and Eigenvectors Numerically

The examples above have shown that not every $n \times n$ matrix admits n independent eigenvectors, so some matrices cannot be diagonalized by similarity transformations. Fortunately, these matrices can be *triangularized* by similarity transformations, as we now show. We will show later on that this allows solving systems of linear differential equations regardless of the structure of the system's matrix of coefficients.

It is important to notice that if a matrix A is triangularized by similarity transformations,

$$T = Q^{-1} A Q \ ,$$

then the eigenvalues of the triangular matrix T are equal to those of the original matrix A. In fact, if

$$A\mathbf{x} = \lambda \mathbf{x} \ ,$$

then

$$QTQ^{-1}\mathbf{x} = \lambda \mathbf{x} \ ,$$

that is,

$$T\mathbf{y} = \lambda \mathbf{y}$$

where

$$\mathbf{y} = Q^{-1}\mathbf{x} \ ,$$

so λ is also an eigenvalue for T. The eigenvectors, however, are changed according to the last equation.

The Schur decomposition does even better, since it triangularizes any square matrix A by a *unitary* (possibly complex) transformation:

$$T = S^H A S \ .$$

This transformation is equivalent to factoring A into the product

$$A = S T S^H \ ,$$

and this product is called the *Schur decomposition* of A. Numerically stable and efficient algorithms exist for the Schur decomposition. In this note, we will not study these algorithms, but only show that all square matrices admit a Schur decomposition.

5.3.1 Rotations into the x_1 Axis

An important preliminary fact concerns vector rotations. Let $\mathbf{e}_1$ be the first column of the identity matrix. It is intuitively obvious that any nonzero real vector $\mathbf{x}$ can be rotated into a vector parallel to $\mathbf{e}_1$. Formally, take any orthogonal matrix S whose first column is

$$\mathbf{s}_1 = \frac{\mathbf{x}}{\|\mathbf{x}\|} \ .$$

Since $\mathbf{s}_1^T \mathbf{x} = \mathbf{x}^T\mathbf{x}/\|\mathbf{x}\| = \|\mathbf{x}\|$, and since all the other $\mathbf{s}_j$ are orthogonal to $\mathbf{s}_1$, we have

$$S^T\mathbf{x} = \begin{bmatrix} \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_n^T \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{s}_1^T\mathbf{x} \\ \vdots \\ \mathbf{s}_n^T\mathbf{x} \end{bmatrix} = \begin{bmatrix} \|\mathbf{x}\| \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

which is parallel to $\mathbf{e}_1$ as desired. It may be less obvious that a *complex* vector $\mathbf{x}$ can be transformed into a *real* vector parallel to $\mathbf{e}_1$ by a unitary transformation. But the trick is the same: let

$$\mathbf{s}_1 = \frac{\mathbf{x}}{\|\mathbf{x}\|} .$$

Now $\mathbf{s}_1$ may be complex. We have $\mathbf{s}_1^H \mathbf{x} = \mathbf{x}^H \mathbf{x} / \|\mathbf{x}\| = \|\mathbf{x}\|$, and

$$S^H \mathbf{x} = \begin{bmatrix} \mathbf{s}_1^H \\ \vdots \\ \mathbf{s}_n^H \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{s}_1^H \mathbf{x} \\ \vdots \\ \mathbf{s}_n^H \mathbf{x} \end{bmatrix} = \begin{bmatrix} \|\mathbf{x}\| \\ 0 \\ \vdots \\ 0 \end{bmatrix} ,$$

just about like before. We are now ready to triangularize an arbitrary square matrix A.

5.3.2 The Schur Decomposition

The Schur decomposition theorem is the cornerstone of eigenvalue computations. It states that any square matrix can be triangularized by unitary transformations. The diagonal elements of a triangular matrix are its eigenvalues, and unitary transformations preserve eigenvalues. Consequently, if the Schur decomposition of a matrix can be computed, its eigenvalues can be determined. Moreover, as we will see later, a system of linear differential equations can be solved regardless of the structure of the matrix of its coefficients.

Lemma 5.3.1 *If A is an $n \times n$ matrix and λ and $\mathbf{x}$ are an eigenvalue of A and its corresponding eigenvector,*

$$A\mathbf{x} = \lambda \mathbf{x} \tag{5.12}$$

then there is a transformation

$$T = U^H A U$$

where U is a unitary, $n \times n$ matrix, such that

$$T = \left[\begin{array}{c|c} \lambda & \\ 0 & \\ \vdots & C \\ 0 & \end{array}\right] .$$

Proof. Let U be a unitary transformation that transforms the (possibly complex) eigenvector $\mathbf{x}$ of A into a real vector on the x_1 axis:

$$\mathbf{x} = U \begin{bmatrix} r \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where r is the nonzero norm of $\mathbf{x}$. By substituting this into (5.12) and rearranging we have

$$\begin{aligned} AU \begin{bmatrix} r \\ 0 \\ \vdots \\ 0 \end{bmatrix} &= \lambda U \begin{bmatrix} r \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ U^H A U \begin{bmatrix} r \\ 0 \\ \vdots \\ 0 \end{bmatrix} &= \lambda \begin{bmatrix} r \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{aligned}$$

$$U^H A U \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \lambda \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$T \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \lambda \\ 0 \\ \vdots \\ 0 \end{bmatrix} .$$

The last left-hand side is the first column of T, and the corresponding right-hand side is of the form required by the lemma. Δ

Theorem 5.3.2 (Schur) *If A is any $n \times n$ matrix then there exists a unitary $n \times n$ matrix S such that*

$$S^H A S = T$$

where T is triangular. Furthermore, S can be chosen so that the eigenvalues λ_i of A appear in any order along the diagonal of T.

Proof. By induction. The theorem obviously holds for $n = 1$:

$$1\,A\,1 = 1\ .$$

Suppose it holds for all matrices of order $n-1$. Then from the lemma there exists a unitary U such that

$$U^H A U = \left[\begin{array}{c|c} \lambda & \\ 0 & \\ \vdots & C \\ 0 & \end{array} \right]$$

where λ is any eigenvalue of A. Partition C into a row vector and an $(n-1) \times (n-1)$ matrix G:

$$C = \begin{bmatrix} \mathbf{w}^H \\ G \end{bmatrix} .$$

By the inductive hypothesis, there is a unitary matrix V such that $V^H G V$ is a Schur decomposition of G. Let

$$S = U \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & V & \\ 0 & & & \end{bmatrix} .$$

Clearly, S is a unitary matrix, and $S^H A S$ is upper-triangular. Since the elements on the diagonal of a triangular matrix are the eigenvalues, $S^H A S$ is the Schur decomposition of A. Because we can pick any eigenvalue as λ, the order of eigenvalues can be chosen arbitrarily. Δ

This theorem does not say how to compute the Schur decomposition, only that it exists. Fortunately, there is a stable and efficient algorithm to compute the Schur decomposition. This is the preferred way to compute eigenvalues numerically.

5.4 Eigenvalues/Vectors and Singular Values/Vectors

In this section we prove a few additional important properties of eigenvalues and eigenvectors. In the process, we also establish a link between singular values/vectors and eigenvalues/vectors. While this link is very important, it is useful to remember that eigenvalues/vectors and singular values/vectors are conceptually and factually very distinct entities (recall figure 5.1).

First, a general relation between determinant and eigenvalues.

Theorem 5.4.1 *The determinant of a matrix is equal to the product of its eigenvalues.*

Proof. The proof is very simple, given the Schur decomposition. In fact, we know that the eigenvalues of a matrix A are equal to those of the triangular matrix in the Schur decomposition of A. Furthermore, we know from theorem 5.1.6 that the determinant of a triangular matrix is the product of the elements on its diagonal. If we recall that a unitary matrix has determinant 1 or -1, that the determinants of S and S^H are the same, and that the determinant of a product of matrices is equal to the product of the determinants, the proof is complete. Δ

We saw that an $n \times n$ Hermitian matrix with n distinct eigenvalues admits n orthonormal eigenvectors (corollary 5.1.5). The assumption of distinct eigenvalues made the proof simple, but is otherwise unnecessary. In fact, now that we have the Schur decomposition, we can state the following stronger result.

Theorem 5.4.2 (Spectral theorem) *Every Hermitian matrix can be diagonalized by a unitary matrix, and every real symmetric matrix can be diagonalized by an orthogonal matrix:*

$$\begin{aligned} A = A^H \quad &\Rightarrow \quad A = S\Lambda S^H \\ A \text{ real}, \; A = A^T \quad &\Rightarrow \quad A = S\Lambda S^T, S \text{ real} . \end{aligned}$$

In either case, Λ is real and diagonal.

Proof. We already know that Hermitian matrices (and therefore real and symmetric ones) have real eigenvalues (theorem 5.1.2), so Λ is real. Let now

$$A = STS^H$$

be the Schur decomposition of A. Since A is Hermitian, so is T. In fact, $T = S^H AS$, and

$$T^H = (S^H AS)^H = S^H A^H S = S^H AS = T \, .$$

But the only way that T can be both triangular and Hermitian is for it to be diagonal, because $0^* = 0$. Thus, the Schur decomposition of a Hermitian matrix is in fact a diagonalization, and this is the first equation of the theorem (the diagonal of a Hermitian matrix must be real).

Let now A be real and symmetric. All that is left to prove is that then its eigenvectors are real. But eigenvectors are the solution of the homogeneous system (5.6), which is both real and rank-deficient, and therefore admits nontrivial real solutions. Thus, S is real, and $S^H = S^T$. Δ

In other words, a Hermitian matrix, real or not, with distinct eigenvalues or not, has real eigenvalues and n orthonormal eigenvectors. If in addition the matrix is real, so are its eigenvectors.

We recall that a real matrix A such that for every nonzero $\mathbf{x}$ we have $\mathbf{x}^T A\mathbf{x} > 0$ is said to be *positive definite*. It is *positive semidefinite* if for every nonzero $\mathbf{x}$ we have $\mathbf{x}^T A\mathbf{x} \geq 0$. Notice that a positive definite matrix is also positive semidefinite. Positive definite or semidefinite matrices arise in the solution of overconstrained linear systems, because $A^T A$ is positive semidefinite for every A (lemma 5.4.5). They also occur in geometry through the equation of an ellipsoid,

$$\mathbf{x}^T Q \mathbf{x} = 1$$

in which Q is positive definite. In physics, positive definite matrices are associated to quadratic forms $\mathbf{x}^T Q \mathbf{x}$ that represent energies or second-order moments of mass or force distributions. Their physical meaning makes them positive definite, or at least positive semidefinite (for instance, energies cannot be negative). The following result relates eigenvalues/vectors with singular values/vectors for positive semidefinite matrices.

Theorem 5.4.3 *The eigenvalues of a real, symmetric, positive semidefinite matrix A are equal to its singular values. The eigenvectors of A are also its singular vectors, both left and right.*

Proof. From the previous theorem, $A = S\Lambda S^T$, where both Λ and S are real. Furthermore, the entries in Λ are nonnegative. In fact, from

$$A\mathbf{s}_i = \lambda \mathbf{s}_i$$

we obtain

$$\mathbf{s}_i^T A \mathbf{s}_i = \mathbf{s}_i^T \lambda \mathbf{s}_i = \lambda \mathbf{s}_i^T \mathbf{s}_i = \lambda \|\mathbf{s}_i\|^2 = \lambda \ .$$

If A is positive semidefinite, then $\mathbf{x}^T A \mathbf{x} \geq 0$ for any nonzero $\mathbf{x}$, and in particular $\mathbf{s}_i^T A \mathbf{s}_i \geq 0$, so that $\lambda \geq 0$.

But

$$A = S\Lambda S^T$$

with nonnegative diagonal entries in Λ is the singular value decomposition $A = U\Sigma V^T$ of A with $\Sigma = \Lambda$ and $U = V = S$. Recall that the eigenvalues in the Schur decomposition can be arranged in any desired order along the diagonal. Δ

Theorem 5.4.4 *A real, symmetric matrix is positive semidefinite iff all its eigenvalues are nonnegative. It is positive definite iff all its eigenvalues are positive.*

Proof. Theorem 5.4.3 implies one of the two directions: If A is real, symmetric, and positive semidefinite, then its eigenvalues are nonnegative. If the proof of that theorem is repeated with the strict inequality, we also obtain that if A is real, symmetric, and positive definite, then its eigenvalues are positive.

Conversely, we show that if all eigenvalues λ of a real and symmetric matrix A are positive (nonnegative) then A is positive definite (semidefinite). To this end, let $\mathbf{x}$ be any nonzero vector. Since real and symmetric matrices have n orthonormal eigenvectors (theorem 5.4.2), we can use these eigenvectors $\mathbf{s}_1, \ldots, \mathbf{s}_n$ as an orthonormal basis for $\mathbf{R}^n$, and write

$$\mathbf{x} = c_1 \mathbf{s}_1 + \ldots + c_n \mathbf{s}_n$$

with

$$c_i = \mathbf{x}^T \mathbf{s}_i \ .$$

But then

$$\begin{aligned} \mathbf{x}^T A \mathbf{x} &= \mathbf{x}^T A (c_1 \mathbf{s}_1 + \ldots + c_n \mathbf{s}_n) = \mathbf{x}^T (c_1 A \mathbf{s}_1 + \ldots + c_n A \mathbf{s}_n) \\ &= \mathbf{x}^T (c_1 \lambda_1 \mathbf{s}_1 + \ldots + c_n \lambda_n \mathbf{s}_n) = c_1 \lambda_1 \mathbf{x}^T \mathbf{s}_1 + \ldots + c_n \lambda_n \mathbf{x}^T \mathbf{s}_n \\ &= \lambda_1 c_1^2 + \ldots + \lambda_n c_n^2 > 0 \text{ (or } \geq 0) \end{aligned}$$

because the λ_i are positive (nonnegative) and not all c_i can be zero. Since $\mathbf{x}^T A \mathbf{x} > 0$ (or ≥ 0) for every nonzero $\mathbf{x}$, A is positive definite (semidefinite). Δ

Theorem 5.4.3 establishes one connection between eigenvalues/vectors and singular values/vectors: for symmetric, positive definite matrices, the concepts coincide. This result can be used to introduce a less direct link, but for arbitrary matrices.

Lemma 5.4.5 *$A^T A$ is positive semidefinite.*

Proof. For any nonzero $\mathbf{x}$ we can write $\mathbf{x}^T A^T A\mathbf{x} = \|A\mathbf{x}\|^2 \geq 0$. Δ

Theorem 5.4.6 *The eigenvalues of $A^T A$ with $m \geq n$ are the squares of the singular values of A; the eigenvectors of $A^T A$ are the right singular vectors of A. Similarly, for $m \leq n$, the eigenvalues of AA^T are the squares of the singular values of A, and the eigenvectors of AA^T are the left singular vectors of A.*

Proof. If $m \geq n$ and $A = U\Sigma V^T$ is the SVD of A, we have

$$A^T A = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$$

which is in the required format to be a (diagonal) Schur decomposition with $S = U$ and $T = \Lambda = \Sigma^2$. Similarly, for $m \leq n$,

$$AA^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T$$

is a Schur decomposition with $S = V$ and $T = \Lambda = \Sigma^2$. Δ

We have seen that important classes of matrices admit a full set of orthonormal eigenvectors. The theorem below characterizes the class of *all* matrices with this property, that is, the class of all normal matrices. To prove the theorem, we first need a lemma.

Lemma 5.4.7 *If for an $n \times n$ matrix B we have $BB^H = B^H B$, then for every $i = 1, \ldots, n$, the norm of the i-th row of B equals the norm of its i-th column.*

Proof. From $BB^H = B^H B$ we deduce

$$\|B\mathbf{x}\|^2 = \mathbf{x}^H B^H B\mathbf{x} = \mathbf{x}^H BB^H\mathbf{x} = \|B^H\mathbf{x}\|^2 . \tag{5.13}$$

If $\mathbf{x} = \mathbf{e}_i$, the i-th column of the $n \times n$ identity matrix, $B\mathbf{e}_i$ is the i-th column of B, and $B^H\mathbf{e}_i$ is the i-th column of B^H, which is the conjugate of the i-th row of B. Since conjugation does not change the norm of a vector, the equality (5.13) implies that the i-th column of B has the same norm as the i-th row of B. Δ

Theorem 5.4.8 *An $n \times n$ matrix is normal if an only if it commutes with its Hermitian:*

$$AA^H = A^H A .$$

Proof. Let $A = STS^H$ be the Schur decomposition of A. Then,

$$AA^H = STS^H ST^H S^H = STT^H S^H \quad \text{and} \quad A^H A = ST^H S^H STS^H = ST^H TS^H .$$

Because S is invertible (even unitary), we have $AA^H = A^H A$ if and only if $TT^H = T^H T$.

However, a triangular matrix T for which $TT^H = T^H T$ must be diagonal. In fact, from the lemma, the norm of the i-th row of T is equal to the norm of its i-th column. Let $i = 1$. Then, the first column of T has norm $|t_{11}|$. The first row has first entry t_{11}, so the only way that its norm can be $|t_{11}|$ is for all other entries in the first row to be zero. We now proceed through $i = 2, \ldots, n$, and reason similarly to conclude that T must be diagonal.

The converse is also obviously true: if T is diagonal, then $TT^H = T^H T$. Thus, $AA^H = A^H A$ if and only if T is diagonal, that is, if and only if A can be diagonalized by a unitary similarity transformation. This is the definition of a normal matrix. Δ

Corollary 5.4.9 *A triangular, normal matrix must be diagonal.*

Proof. We proved this in the proof of theorem 5.4.8. Δ

Checking that $A^H A = AA^H$ is much easier than computing eigenvectors, so theorem 5.4.8 is a very useful characterization of normal matrices. Notice that Hermitian (and therefore also real symmetric) matrices commute trivially with their Hermitians, but so do, for instance, unitary (and therefore also real orthogonal) matrices:

$$UU^H = U^H U = I \; .$$

Thus, Hermitian, real symmetric, unitary, and orthogonal matrices are all normal.

Chapter 6

Ordinary Differential Systems

In this chapter we use the theory developed in chapter 5 in order to solve systems of first-order linear differential equations with constant coefficients. These systems have the following form:

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}(t) \tag{6.1}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{6.2}$$

where $\mathbf{x} = \mathbf{x}(t)$ is an n-dimensional vector function of time t, the dot denotes differentiation, the coefficients a_{ij} in the $n \times n$ matrix A are constant, and the vector function $\mathbf{b}(t)$ is a function of time. The equation (6.2), in which $\mathbf{x}_0$ is a known vector, defines the *initial value* of the solution.

First, we show that *scalar* differential equations of order greater than one can be reduced to *systems* of first-order differential equations. Then, in section 6.2, we recall a general result for the solution of first-order differential systems from the elementary theory of differential equations. In section 6.3, we make this result more specific by showing that the solution to a homogeneous system is a linear combination of exponentials multiplied by polynomials in t. This result is based on the Schur decomposition introduced in chapter 5, which is numerically preferable to the more commonly used Jordan canonical form. Finally, in sections 6.4 and 6.5, we set up and solve a particular differential system as an illustrative example.

6.1 Scalar Differential Equations of Order Higher than One

The first-order system (6.1) subsumes also the case of a scalar differential equation of order n, possibly greater than 1,

$$\frac{d^n y}{dt^n} + c_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \ldots + c_1\frac{dy}{dt} + c_0 y = b(t) \ . \tag{6.3}$$

In fact, such an equation can be reduced to a first-order system of the form (6.1) by introducing the n-dimensional vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y \\ \frac{dy}{dt} \\ \vdots \\ \frac{d^{n-1}y}{dt^{n-1}} \end{bmatrix} .$$

With this definition, we have

$$\begin{aligned} \frac{d^i y}{dt^i} &= x_{i+1} \qquad \text{for } i = 0, \ldots, n-1 \\ \frac{d^n y}{dt^n} &= \frac{dx_n}{dt} , \end{aligned}$$

and $\mathbf{x}$ satisfies the additional $n-1$ equations

$$x_{i+1} = \frac{dx_i}{dt} \tag{6.4}$$

for $i = 1, \ldots, n-1$. If we write the original system (6.3) together with the $n-1$ differential equations (6.4), we obtain the first-order system

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}(t)$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -c_0 & -c_1 & -c_2 & \cdots & -c_{n-1} \end{bmatrix}$$

is the so-called *companion matrix* of (6.3) and

$$\mathbf{b}(t) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b(t) \end{bmatrix} .$$

6.2 General Solution of a Linear Differential System

We know from the general theory of differential equations that a general solution of system (6.1) with initial condition (6.2) is given by

$$\mathbf{x}(t) = \mathbf{x}_h(t) + \mathbf{x}_p(t)$$

where $\mathbf{x}_h(t)$ is the solution of the homogeneous system

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned}$$

and $\mathbf{x}_p(t)$ is a particular solution of

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + \mathbf{b}(t) \\ \mathbf{x}(0) &= \mathbf{0} . \end{aligned}$$

The two solution components $\mathbf{x}_h$ and $\mathbf{x}_p$ can be written by means of the *matrix exponential*, introduced in the following.

For the scalar exponential $e^{\lambda t}$ we can write a Taylor series expansion

$$e^{\lambda t} = 1 + \frac{\lambda t}{1!} + \frac{\lambda^2 t^2}{2!} + \cdots = \sum_{j=0}^{\infty} \frac{\lambda^j t^j}{j!} .$$

Usually[1], in calculus classes, the exponential is introduced by other means, and the Taylor series expansion above is proven as a property.

For matrices, the exponential e^Z of a matrix $Z \in \mathbf{R}^{n \times n}$ is instead *defined* by the infinite series expansion

$$e^Z = I + \frac{Z}{1!} + \frac{Z^2}{2!} + \cdots = \sum_{j=0}^{\infty} \frac{Z^j}{j!} .$$

[1]Not always. In some treatments, the exponential is *defined* through its Taylor series.

Here I is the $n \times n$ identity matrix, and the general term $Z^j/j!$ is simply the matrix Z raised to the jth power divided by the scalar $j!$. It turns out that this infinite sum converges (to an $n \times n$ matrix which we write as e^Z) for every matrix Z. Substituting $Z = At$ gives

$$e^{At} = I + \frac{At}{1!} + \frac{A^2t^2}{2!} + \frac{A^3t^3}{3!} + \cdots = \sum_{j=0}^{\infty} \frac{A^j t^j}{j!}. \tag{6.5}$$

Differentiating both sides of (6.5) gives

$$\begin{aligned} \frac{de^{At}}{dt} &= A + \frac{A^2t}{1!} + \frac{A^3t^2}{2!} + \cdots \\ &= A\left(I + \frac{At}{1!} + \frac{A^2t^2}{2!} + \cdots\right) \\ \frac{de^{At}}{dt} &= Ae^{At}. \end{aligned}$$

Thus, for any vector $\mathbf{w}$, the function $\mathbf{x}_h(t) = e^{At}\mathbf{w}$ satisfies the homogeneous differential system

$$\dot{\mathbf{x}}_h = A\mathbf{x}_h \ .$$

By using the initial values (6.2) we obtain $\mathbf{v} = \mathbf{x}_0$, and

$$\mathbf{x}_h(t) = e^{At}\mathbf{x}(0) \tag{6.6}$$

is a solution to the differential system (6.1) with $\mathbf{b}(t) = \mathbf{0}$ and initial values (6.2). It can be shown that this solution is unique.

From the elementary theory of differential equations, we also know that a particular solution to the nonhomogeneous ($\mathbf{b}(t) \neq \mathbf{0}$) equation (6.1) is given by

$$\mathbf{x}_p(t) = \int_0^t e^{A(t-s)}\,\mathbf{b}(s)\,ds \ .$$

This is easily verified, since by differentiating this expression for $\mathbf{x}_p$ we obtain

$$\dot{\mathbf{x}}_p = Ae^{At}\int_0^t e^{-As}\,b(s)\,ds + e^{At}\,e^{-At}\,\mathbf{b}(t) = A\mathbf{x}_p + \mathbf{b}(t) \ ,$$

so $\mathbf{x}_p$ satisfies equation (6.1).

In summary, we have the following result.

The solution to

$$\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}(t) \tag{6.7}$$

with initial value

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{6.8}$$

is

$$\mathbf{x}(t) = \mathbf{x}_h(t) + \mathbf{x}_p(t) \tag{6.9}$$

where

$$\mathbf{x}_h(t) = e^{At}\mathbf{x}(0) \tag{6.10}$$

and

$$\mathbf{x}_p(t) = \int_0^t e^{A(t-s)}\,\mathbf{b}(s)\,ds \ . \tag{6.11}$$

Since we now have a formula for the general solution to a linear differential system, we seem to have all we need. However, we do not know how to compute the matrix exponential. The naive solution to use the definition (6.5) requires too many terms for a good approximation. As we have done for the SVD and the Schur decomposition, we will only point out that several methods exist for computing a matrix exponential, but we will not discuss how this is done[2]. In a fundamental paper on the subject, *Nineteen dubious ways to compute the exponential of a matrix* (SIAM Review, vol. 20, no. 4, pp. 801-36), Cleve Moler and Charles Van Loan discuss a large number of different methods, pointing out that no one of them is appropriate for all situations. A full discussion of this matter is beyond the scope of these notes.

When the matrix A is constant, as we currently assume, we can be much more specific about the structure of the solution (6.9) of system (6.7), and particularly so about the solution $\mathbf{x}_h(t)$ to the homogeneous part. Specifically, the matrix exponential (6.10) can be written as a linear combination, with constant vector coefficients, of scalar exponentials multiplied by polynomials. In the general theory of linear differential systems, this is shown via the Jordan canonical form. However, in the paper cited above, Moler and Van Loan point out that the Jordan form cannot be computed reliably, and small perturbations in the data can change the results dramatically. Fortunately, a similar result can be found through the Schur decomposition introduced in chapter 5. The next section shows how to do this.

6.3 Structure of the Solution

For the homogeneous case $\mathbf{b}(t) = \mathbf{0}$, consider the first order system of linear differential equations

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} && (6.12) \\ \mathbf{x}(0) &= \mathbf{x}_0 \, . && (6.13) \end{aligned}$$

Two cases arise: either A admits n distinct eigenvalues, or is does not. In chapter 5, we have seen that if (but not only if) A has n distinct eigenvalues then it has n linearly independent eigenvectors (theorem 5.1.1), and we have shown how to find $\mathbf{x}_h(t)$ by solving an eigenvalue problem. In section 6.3.1, we briefly review this solution. Then, in section 6.3.2, we show how to compute the homogeneous solution $\mathbf{x}_h(t)$ in the extreme case of an $n \times n$ matrix A with n *coincident* eigenvalues.

To be sure, we have seen that matrices with coincident eigenvalues can still have a full set of linearly independent eigenvectors (see for instance the identity matrix). However, the solution procedure we introduce in section 6.3.2 for the case of n coincident eigenvalues can be applied regardless to how many linearly independent eigenvectors exist. If the matrix has a full complement of eigenvectors, the solution obtained in section 6.3.2 is the same as would be obtained with the method of section 6.3.1.

Once these two extreme cases (nondefective matrix or all-coincident eigenvalues) have been handled, we show a general procedure in section 6.3.3 for solving a homogeneous or nonhomogeneous differential system for any, square, constant matrix A, defective or not. This procedure is based on backsubstitution, and produces a result analogous to that obtained via Jordan decomposition for the homogeneous part $\mathbf{x}_h(t)$ of the solution. However, since it is based on the numerically sound Schur decomposition, the method of section 6.3.3 is superior in practice. For a nonhomogeneous system, the procedure can be carried out analytically if the functions in the right-hand side vector $\mathbf{b}(t)$ can be integrated.

6.3.1 A is Not Defective

In chapter 5 we saw how to find the homogeneous part $\mathbf{x}_h(t)$ of the solution when A has a full set of n linearly independent eigenvectors. This result is briefly reviewed in this section for convenience.[3]

If A is not defective, then it has n linearly independent eigenvectors $\mathbf{q}_1, \ldots, \mathbf{q}_n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$. Let

$$Q = \left[\begin{array}{ccc} \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{array} \right] .$$

[2]In Matlab, `expm(A)` is the matrix exponential of `A`.

[3]Parts of this subsection and of the following one are based on notes written by Scott Cohen.

This square matrix is invertible because its columns are linearly independent. Since $A\mathbf{q}_i = \lambda_i \mathbf{q}_i$, we have

$$AQ = Q\Lambda, \tag{6.14}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ is a square diagonal matrix with the eigenvalues of A on its diagonal. Multiplying both sides of (6.14) by Q^{-1} on the right, we obtain

$$A = Q\Lambda Q^{-1}. \tag{6.15}$$

Then, system (6.12) can be rewritten as follows:

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} \\ \dot{\mathbf{x}} &= Q\Lambda Q^{-1}\mathbf{x} \\ Q^{-1}\dot{\mathbf{x}} &= \Lambda Q^{-1}\mathbf{x} \\ \dot{\mathbf{y}} &= \Lambda \mathbf{y}, \end{aligned} \tag{6.16}$$

where $\mathbf{y} = Q^{-1}\mathbf{x}$. The last equation (6.16) represents n uncoupled, homogeneous, differential equations $\dot{y}_i = \lambda_i y_i$. The solution is

$$\mathbf{y}_h(t) = e^{\Lambda t}\mathbf{y}(0),$$

where

$$e^{\Lambda t} = \mathrm{diag}(e^{\lambda_1 t}, \ldots, e^{\lambda_n t}).$$

Using the relation $\mathbf{x} = Q\mathbf{y}$, and the consequent relation $\mathbf{y}(0) = Q^{-1}\mathbf{x}(0)$, we see that the solution to the homogeneous system (6.12) is

$$\mathbf{x}_h(t) = Qe^{\Lambda t}Q^{-1}\mathbf{x}(0).$$

If A is normal, that is, if it has n orthonormal eigenvectors $q_1, \ldots q_n$, then Q is replaced by the Hermitian matrix $S = \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_n \end{bmatrix}$, Q^{-1} is replaced by S^H, and the solution to (6.12) becomes

$$\mathbf{x}_h(t) = Se^{\Lambda t}S^H\mathbf{x}(0).$$

6.3.2 A Has n Coincident Eigenvalues

When $A = Q\Lambda Q^{-1}$, we derived that the solution to (6.12) is $\mathbf{x}_h(t) = Qe^{\Lambda t}Q^{-1}\mathbf{x}(0)$. Comparing with (6.6), it should be the case that

$$e^{Q(\Lambda t)Q^{-1}} = Qe^{\Lambda t}Q^{-1}.$$

This follows easily from the definition of e^Z and the fact that $(Q(\Lambda t)Q^{-1})^j = Q(\Lambda t)^j Q^{-1}$. Similarly, if $A = S\Lambda S^H$, where S is Hermitian, then the solution to (6.12) is $\mathbf{x}_h(t) = Se^{\Lambda t}S^H\mathbf{x}(0)$, and

$$e^{S(\Lambda t)S^H} = Se^{\Lambda t}S^H.$$

How can we compute the matrix exponential in the extreme case in which A has n coincident eigenvalues, regardless of the number of its linearly independent eigenvectors? In any case, A admits a Schur decomposition

$$A = STS^H$$

(theorem 5.3.2). We recall that S is a unitary matrix and T is upper triangular with the eigenvalues of A on its diagonal. Thus we can write T as

$$T = \Lambda + N,$$

where Λ is diagonal and N is *strictly* upper triangular. The solution (6.6) in this case becomes

$$\mathbf{x}_h(t) = e^{S(Tt)S^H}\mathbf{x}(0) = Se^{Tt}S^H\mathbf{x}(0) = Se^{\Lambda t + Nt}S^H\mathbf{x}(0).$$

Thus we can compute (6.6) if we can compute $e^{Tt} = e^{\Lambda t + Nt}$. This turns out to be almost as easy as computing $e^{\Lambda t}$ when the diagonal matrix Λ is a multiple of the identity matrix:

$$\Lambda = \lambda I$$

that is, when all the eigenvalues of A coincide. In fact, in this case, Λt and Nt commute:

$$\Lambda t \, Nt = \lambda I t \, Nt = \lambda t \, Nt = Nt \, \lambda t = Nt \, \lambda I t = Nt \, \Lambda t \, .$$

It can be shown that if two matrices Z_1 and Z_2 commute, that is if

$$Z_1 Z_2 = Z_2 Z_1 \; ,$$

then

$$e^{Z_1 + Z_2} = e^{Z_1} e^{Z_2} = e^{Z_2} e^{Z_1} \; .$$

Thus, in our case, we can write

$$e^{\Lambda t + Nt} = e^{\Lambda t} e^{Nt}.$$

We already know how to compute $e^{\Lambda t}$, so it remains to show how to compute e^{Nt}. The fact that Nt is strictly upper triangular makes the computation of this matrix exponential much simpler than for a general matrix Z.

Suppose, for example, that N is 4×4. Then N has three nonzero superdiagonals, N^2 has two nonzero superdiagonals, N^3 has one nonzero superdiagonal, and N^4 is the zero matrix:

$$\begin{aligned} N &= \begin{bmatrix} 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow N^2 = \begin{bmatrix} 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \\ N^3 &= \begin{bmatrix} 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow N^4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} . \end{aligned}$$

In general, for a strictly upper triangular $n \times n$ matrix, we have $N^j = 0$ for all $j \geq n$ (i.e., N is nilpotent of order n). Therefore,

$$e^{Nt} = \sum_{j=0}^{\infty} \frac{N^j t^j}{j!} = \sum_{j=0}^{n-1} \frac{N^j t^j}{j!}$$

is simply a finite sum, and the exponential reduces to a matrix polynomial.

In summary, the general solution to the homogeneous differential system (6.12) with initial value (6.13) when the $n \times n$ matrix A has n coincident eigenvalues is given by

$$\mathbf{x}_h(t) = S e^{\Lambda t} \sum_{j=0}^{n-1} \frac{N^j t^j}{j!} S^H \mathbf{x}_0 \tag{6.17}$$

where

$$A = S(\Lambda + N)S^H$$

is the Schur decomposition of A,

$$\Lambda = \lambda I$$

is a multiple of the identity matrix containing the coincident eigenvalues of A on its diagonal, and N is strictly upper triangular.

6.3.3 The General Case

We are now ready to solve the linear differential system

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + \mathbf{b}(t) && (6.18)\\ \mathbf{x}(0) &= \mathbf{x}_0 && (6.19)\end{aligned}$$

in the general case of a constant matrix A, defective or not, with arbitrary $\mathbf{b}(t)$. In fact, let $A = STS^H$ be the Schur decomposition of A, and consider the transformed system

$$\dot{\mathbf{y}}(t) = T\mathbf{y}(t) + \mathbf{c}(t) \tag{6.20}$$

where

$$\mathbf{y}(t) = S^H\mathbf{x}(t) \quad \text{and} \quad \mathbf{c}(t) = S^H\mathbf{b}(t)\ . \tag{6.21}$$

The triangular matrix T can always be written in the following form:

$$T = \begin{bmatrix} T_{11} & \cdots & \cdots & T_{1k} \\ 0 & T_{22} & \cdots & T_{2k} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & T_{kk} \end{bmatrix}$$

where the diagonal blocks T_{ii} for $i = 1, \ldots, k$ are of size $n_i \times n_i$ (possibly 1×1) and contain all-coincident eigenvalues. The remaining nonzero blocks T_{ij} with $i < j$ can be in turn bundled into matrices

$$R_i = \begin{bmatrix} T_{i,i+1} & \cdots & T_{i,k} \end{bmatrix}$$

that contain everything to the right of the corresponding T_{ii}. The vector $\mathbf{c}(t)$ can be partitioned correspondingly as follows

$$\mathbf{c}(t) = \begin{bmatrix} \mathbf{c}_1(t) \\ \vdots \\ \mathbf{c}_k(t) \end{bmatrix}$$

where $\mathbf{c}_i$ has n_i entries, and the same can be done for

$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{y}_1(t) \\ \vdots \\ \mathbf{y}_k(t) \end{bmatrix}$$

and for the initial values

$$\mathbf{y}(0) = \begin{bmatrix} \mathbf{y}_1(0) \\ \vdots \\ \mathbf{y}_k(0) \end{bmatrix} .$$

The triangular system (6.20) can then be solved by backsubstitution as follows:

for $i = k$ down to 1
    if $i < k$
        $\mathbf{d}_i(t) = R_i\mathbf{y}_{i+1}(t)$
    else
        $\mathbf{d}_i(t) = \mathbf{0}$ (an n_k-dimensional vector of zeros)
    end
    $T_{ii} = \lambda_i I + N_i$ (diagonal and strictly upper-triangular part of T_{ii})
    $\mathbf{y}_i(t) = e^{\lambda_i It}\sum_{j=0}^{n_i-1}\frac{N_i^j t^j}{j!}\mathbf{y}_i(0) + \int_0^t \left(e^{\lambda_i I(t-s)}\sum_{j=0}^{n_i-1}\frac{N_i^j(t-s)^j}{j!}\right)(\mathbf{c}_i(s) - \mathbf{d}_i(s))\,ds$
end.

In this procedure, the expression for $\mathbf{y}_i(t)$ is a direct application of equations (6.9), (6.10), (6.11), and (6.17) with $S = I$. In the general case, the applicability of this routine depends on whether the integral in the expression for $\mathbf{y}_i(t)$ can be computed analytically. This is certainly the case when $\mathbf{b}(t)$ is a constant vector $\mathbf{b}$, because then the integrand is a linear combination of exponentials multiplied by polynomials in $t - s$, which can be integrated by parts.

The solution $\mathbf{x}(t)$ for the original system (6.18) is then

$$\mathbf{x}(t) = S\mathbf{y}(t) .$$

As an illustration, we consider a very small example, the 2×2 homogeneous, triangular case,

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} . \tag{6.22}$$

When $t_{11} = t_{22} = \lambda$, we obtain

$$\mathbf{y}(t) = e^{\lambda I t} \begin{bmatrix} 1 & t_{12}t \\ 0 & 1 \end{bmatrix} \mathbf{y}(0) .$$

In scalar form, this becomes

$$\begin{aligned} y_1(t) &= (y_1(0) + t_{12}y_2(0)\,t)\,e^{\lambda t} \\ y_2(t) &= y_2(0)\,e^{\lambda t} , \end{aligned}$$

and it is easy to verify that this solution satisfies the differential system (6.22).

When $t_{11} = \lambda_1 \neq t_{22} = \lambda_2$, we could solve the system by finding the eigenvectors of T, since we know that in this case two linearly independent eigenvectors exist (theorem 5.1.1). Instead, we apply the backsubstitution procedure introduced in this section. The second equation of the system,

$$\dot{y}_2(t) = t_{22}y_2$$

has solution

$$y_2(t) = y_2(0)\,e^{\lambda_2 t} .$$

We then have

$$d_1(t) = t_{12}y_2(t) = t_{12}y_2(0)\,e^{\lambda_2 t}$$

and

$$\begin{aligned} y_1(t) &= y_1(0)e^{\lambda_1 t} - \int_0^t e^{\lambda_1(t-s)}d_1(s)\,ds \\ &= y_1(0)e^{\lambda_1 t} - t_{12}y_2(0)\,e^{\lambda_1 t}\int_0^t e^{-\lambda_1 s}e^{\lambda_2 s}\,ds \\ &= y_1(0)e^{\lambda_1 t} - t_{12}y_2(0)\,e^{\lambda_1 t}\int_0^t e^{(\lambda_2-\lambda_1)s}\,ds \\ &= y_1(0)e^{\lambda_1 t} - \frac{t_{12}y_2(0)}{\lambda_2-\lambda_1}\,e^{\lambda_1 t}(e^{(\lambda_2-\lambda_1)t} - 1) \\ &= y_1(0)e^{\lambda_1 t} - \frac{t_{12}y_2(0)}{\lambda_2-\lambda_1}\,(e^{\lambda_2 t} - e^{\lambda_1 t}) \end{aligned}$$

Exercise: verify that this solution satisfies both the differential equation (6.22) and the initial value equation $\mathbf{y}(0) = \mathbf{y}_0$.

Thus, the solutions to system (6.22) for $t_{11} = t_{22}$ and for $t_{11} \neq t_{22}$ have different forms. While $y_2(t)$ is the same in both cases, we have

$$\begin{aligned} y_1(t) &= y_1(0)\,e^{\lambda t} + t_{12}y_2(0)\,t\,e^{\lambda t} \quad \text{if} \quad t_{11} = t_{22} \\ y_1(t) &= y_1(0)e^{\lambda_1 t} - \frac{t_{12}y_2(0)}{\lambda_2-\lambda_1}\,(e^{\lambda_2 t} - e^{\lambda_1 t}) \quad \text{if} \quad t_{11} \neq t_{22} . \end{aligned}$$

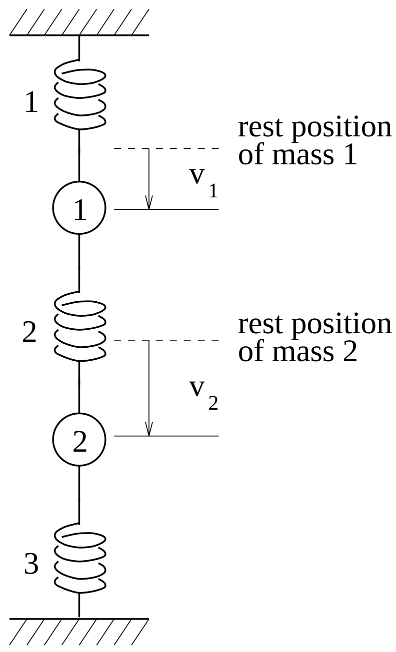

Figure 6.1: A system of masses and springs. In the absence of external forces, the two masses would assume the positions indicated by the dashed lines.

This would seem to present numerical difficulties when $t_{11} \approx t_{22}$, because the solution would suddenly switch from one form to the other as the difference between t_{11} and t_{22} changes from about zero to exactly zero or viceversa. This, however, is not a problem. In fact,

$$\lim_{\lambda_1 \to \lambda} \frac{e^{\lambda t} - e^{\lambda_1 t}}{\lambda - \lambda_1} = t\, e^{\lambda t} ,$$

and the transition between the two cases is smooth.

6.4 A Concrete Example

In this section we set up and solve a more concrete example of a system of differential equations. The initial system has two second-order equations, and is transformed into a first-order system with four equations. The 4×4 matrix of the resulting system has an interesting structure, which allows finding eigenvalues and eigenvectors analytically with a little trick. The point of this section is to show how to transform the complex formal solution of the differential system, computed with any of the methods described above, into a real solution in a form appropriate to the problem at hand.

Consider the mechanical system in figure 6.1. Suppose that we want to study the evolution of the system over time. Since forces are proportional to accelerations, because of Newton's law, and since accelerations are second derivatives of position, the new equations are differential. Because differentiation occurs only with respect to one variable, time, these are *ordinary* differential equations, as opposed to partial.

In the following we write the differential equations that describe this system. Two linear differential equations of the second order[4] result. We will then transform these into four linear differential equations of the first order.

[4] Recall that the order of a differential equation is the highest degree of derivative that appears in it.

By Hooke's law, the three springs exert forces that are proportional to the springs' elongations:

$$\begin{aligned} f_1 &= c_1 v_1 \\ f_2 &= c_2(v_2 - v_1) \\ f_3 &= -c_3 v_2 \end{aligned}$$

where the c_i are the positive spring constants (in newtons per meter).

The accelerations of masses 1 and 2 (springs are assumed to be massless) are proportional to their accelerations, according to Newton's second law:

$$\begin{aligned} m_1 \ddot{v}_1 &= -f_1 + f_2 = -c_1 v_1 + c_2(v_2 - v_1) = -(c_1 + c_2)v_1 + c_2 v_2 \\ m_2 \ddot{v}_2 &= -f_2 + f_3 = -c_2(v_2 - v_1) - c_3 v_2 = c_2 v_1 - (c_2 + c_3)v_2 \end{aligned}$$

or, in matrix form,

$$\ddot{\mathbf{v}} = B\mathbf{v} \tag{6.23}$$

where

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -\frac{c_1+c_2}{m_1} & \frac{c_2}{m_1} \\ \frac{c_2}{m_2} & -\frac{c_2+c_3}{m_2} \end{bmatrix} .$$

We also assume that initial conditions

$$\mathbf{v}(0) \quad \text{and} \quad \dot{\mathbf{v}}(0) \tag{6.24}$$

are given, which specify positions and velocities of the two masses at time $t = 0$.

To solve the second-order system (6.23), we will first transform it to a system of four first-order equations. As shown in the introduction to this chapter, the trick is to introduce variables to denote the first-order derivatives of $\mathbf{v}$, so that second-order derivatives of $\mathbf{v}$ are first-order derivatives of the new variables. For uniformity, we define four new variables

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} \tag{6.25}$$

so that

$$u_3 = \dot{v}_1 \quad \text{and} \quad u_4 = \dot{v}_2 \, ,$$

while the original system (6.23) becomes

$$\begin{bmatrix} \dot{u}_3 \\ \dot{u}_4 \end{bmatrix} = B \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} .$$

We can now gather these four first-order differential equations into a single system as follows:

$$\dot{\mathbf{u}} = A\mathbf{u} \tag{6.26}$$

where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \multicolumn{2}{c}{\multirow{2}{*}{B}} & 0 & 0 \\ & & 0 & 0 \end{bmatrix} .$$

Likewise, the initial conditions (6.24) are replaced by the (known) vector

$$\mathbf{u}(0) = \begin{bmatrix} \mathbf{v}(0) \\ \dot{\mathbf{v}}(0) \end{bmatrix} .$$

In the next section we solve equation (6.26).

6.5 Solution of the Example

Not all matrices have a full set of linearly independent eigenvectors. With the system of springs in figure 6.1, however, we are lucky. The eigenvalues of A are solutions to the equation

$$A\mathbf{x} = \lambda\mathbf{x} \;, \tag{6.27}$$

where we recall that

$$A = \begin{bmatrix} 0 & I \\ B & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -\frac{c_1+c_2}{m_1} & \frac{c_2}{m_1} \\ \frac{c_2}{m_2} & -\frac{c_2+c_3}{m_2} \end{bmatrix} .$$

Here, the zeros in A are 2×2 matrices of zeros, and I is the 2×2 identity matrix. If we partition the vector $\mathbf{x}$ into its upper and lower halves $\mathbf{y}$ and $\mathbf{z}$,

$$\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} ,$$

we can write

$$A\mathbf{x} = \begin{bmatrix} 0 & I \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ B\mathbf{y} \end{bmatrix}$$

so that the eigenvalue equation (6.27) can be written as the following pair of equations:

$$\begin{aligned} \mathbf{z} &= \lambda\mathbf{y} \\ B\mathbf{y} &= \lambda\mathbf{z} \;, \end{aligned} \tag{6.28}$$

which yields

$$B\mathbf{y} = \mu\mathbf{y} \quad \text{with} \quad \mu = \lambda^2 \;.$$

In other words, the eigenvalues of A are the square roots of the eigenvalues of B: if we denote the two eigenvalues of B as μ_1 and μ_2, then the eigenvalues of A are

$$\lambda_1 = \sqrt{\mu_1} \qquad \lambda_2 = -\sqrt{\mu_1} \qquad \lambda_3 = \sqrt{\mu_2} \qquad \lambda_4 = -\sqrt{\mu_2} \;.$$

The eigenvalues μ_1 and μ_2 of B are the solutions of

$$\det(B - \mu I) = \left(\frac{c_1+c_2}{m_1} + \mu\right)\left(\frac{c_2+c_3}{m_2} + \mu\right) - \frac{c_2^2}{m_1 m_2} = \mu^2 + 2\alpha\mu + \beta = 0$$

where

$$\alpha = \frac{1}{2}\left(\frac{c_1+c_2}{m_1} + \frac{c_2+c_3}{m_2}\right) \quad \text{and} \quad \beta = \frac{c_1c_2 + c_1c_3 + c_2c_3}{m_1 m_2}$$

are positive constants that depend on the elastic properties of the springs and on the masses. We then obtain

$$\mu_{1,2} = -\alpha \pm \gamma \;,$$

where

$$\gamma = \sqrt{\alpha^2 - \beta} = \sqrt{\frac{1}{4}\left(\frac{c_1+c_2}{m_1} - \frac{c_2+c_3}{m_2}\right)^2 + \frac{c_2^2}{m_1 m_2}} \;.$$

The constant γ is real because the radicand is nonnegative. We also have that $\alpha \leq \gamma$, so that the two solutions $\mu_{1,2}$ are real and negative, and the four eigenvalues of A,

$$\lambda_1 = \sqrt{-\alpha+\gamma}\,, \quad \lambda_2 = -\sqrt{-\alpha+\gamma}\,, \tag{6.29}$$

$$\lambda_3 = \sqrt{-\alpha-\gamma}\,, \quad \lambda_4 = -\sqrt{-\alpha-\gamma} \tag{6.30}$$

come in nonreal, complex-conjugate pairs. This is to be expected, since our system of springs obviously exhibits an oscillatory behavior.

Also the eigenvectors of A can be derived from those of B. In fact, from equation (6.28) we see that if $\mathbf{y}$ is an eigenvector of B corresponding to eigenvalue $\mu = \lambda^2$, then there are two corresponding eigenvectors for A of the form

$$\mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \pm\lambda\mathbf{y} \end{bmatrix} . \tag{6.31}$$

The four eigenvectors of B are the solutions of

$$(B - (-\alpha \pm \gamma)I)\mathbf{y} = 0 . \tag{6.32}$$

Since $\pm(-\alpha \pm \gamma)$ are eigenvalues of B, the determinant of this equation is zero, and the two scalar equations in (6.32) must be linearly dependent. The first equation reads

$$-\left(\frac{c_1 + c_2}{m_1} - \alpha \pm \gamma\right) y_1 + \frac{c_2}{m_1} y_2 = 0$$

and is obviously satisfied by any vector of the form

$$\mathbf{y} = k \begin{bmatrix} \frac{c_2}{m_1} \\ \frac{c_1+c_2}{m_1} - \alpha \pm \gamma \end{bmatrix}$$

where k is an arbitrary constant. For $k \neq 0$, $\mathbf{y}$ denotes the two eigenvectors of B, and from equation (6.31) the four eigenvectors of A are proportional to the four columns of the following matrix:

$$Q = \begin{bmatrix} \frac{c_2}{m_1} & \frac{c_2}{m_1} & \frac{c_2}{m_1} & \frac{c_2}{m_1} \\ a + \lambda_1^2 & a + \lambda_2^2 & a + \lambda_3^2 & a + \lambda_4^2 \\ \lambda_1 \frac{c_2}{m_1} & \lambda_2 \frac{c_2}{m_1} & \lambda_3 \frac{c_2}{m_1} & \lambda_4 \frac{c_2}{m_1} \\ \lambda_1 \left(a + \lambda_1^2\right) & \lambda_2 \left(a + \lambda_2^2\right) & \lambda_3 \left(a + \lambda_3^2\right) & \lambda_4 \left(a + \lambda_4^2\right) \end{bmatrix} \tag{6.33}$$

where

$$a = \frac{c_1 + c_2}{m_1} .$$

The general solution to the first-order differential system (6.26) is then given by equation (6.17). Since we just found four distinct eigenvectors, however, we can write more simply

$$\mathbf{u}(t) = Q e^{\Lambda t} Q^{-1} \mathbf{u}(0) \tag{6.34}$$

where

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{bmatrix} .$$

In these expressions, the values of λ_i are given in equations (6.30), and Q is in equation (6.33).

Finally, the solution to the original, second-order system (6.23) can be obtained from equation (6.25) by noticing that $\mathbf{v}$ is equal to the first two components of $\mathbf{u}$.

This completes the solution of our system of differential equations. However, it may be useful to add some algebraic manipulation in order to show that the solution is indeed oscillatory. As we see in the following, the masses' motions can be described by the superposition of two sinusoids whose frequencies depend on the physical constants involved (masses and spring constants). The amplitudes and phases of the sinusoids, on the other hand, depend on the initial conditions.

To simplify our manipulation, we note that

$$\mathbf{u}(t) = Q e^{\Lambda t} \mathbf{w} ,$$

where we defined

$$\mathbf{w} = Q^{-1}\mathbf{u}(0) \ . \tag{6.35}$$

We now leave the constants in $\mathbf{w}$ unspecified, and derive the general solution $\mathbf{v}(t)$ for the original, second-order problem. Numerical values for the constants can be found from the initial conditions $\mathbf{u}(0)$ by equation (6.35). We have

$$\mathbf{v}(t) = Q(1:2,:)e^{\Lambda t}\mathbf{w} \ ,$$

where $Q(1:2,:)$ denotes the first two rows of Q. Since

$$\lambda_2 = -\lambda_1 \text{ and } \quad \lambda_4 = -\lambda_3$$

(see equations (6.30)), we have

$$Q(1:2,:) = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_2 \end{bmatrix}$$

where we defined

$$\mathbf{q}_1 = \begin{bmatrix} \frac{c_2}{m_1} \\ \frac{c_1+c_2}{m_1} + \lambda_1^2 \end{bmatrix} \quad \text{and} \quad \mathbf{q}_2 = \begin{bmatrix} \frac{c_2}{m_1} \\ \frac{c_1+c_2}{m_1} + \lambda_3^2 \end{bmatrix} \ .$$

Thus, we can write

$$\mathbf{v}(t) = \mathbf{q}_1 \left(k_1 e^{\lambda_1 t} + k_2 e^{-\lambda_1 t}\right) + \mathbf{q}_2 \left(k_3 e^{\lambda_3 t} + k_4 e^{-\lambda_3 t}\right) \ .$$

Since the λs are imaginary but $\mathbf{v}(t)$ is real, the k_i must come in complex-conjugate pairs:

$$k_1 = k_2^* \quad \text{and} \quad k_3 = k_4^* \ . \tag{6.36}$$

In fact, we have

$$\mathbf{v}(0) = \mathbf{q}_1(k_1 + k_2) + \mathbf{q}_2(k_3 + k_4)$$

and from the derivative

$$\dot{\mathbf{v}}(t) = \mathbf{q}_1\lambda_1 \left(k_1 e^{\lambda_1 t} - k_2 e^{-\lambda_1 t}\right) + \mathbf{q}_2\lambda_3 \left(k_3 e^{\lambda_3 t} - k_4 e^{-\lambda_3 t}\right)$$

we obtain

$$\dot{\mathbf{v}}(0) = \mathbf{q}_1\lambda_1(k_1 - k_2) + \mathbf{q}_2\lambda_3(k_3 - k_4) \ .$$

Since the vectors $\mathbf{q}_i$ are independent (assuming that the mass c_2 is nonzero), this means that

$$\begin{array}{ll} k_1 + k_2 \text{ is real} & k_1 - k_2 \text{ is purely imaginary} \\ k_3 + k_4 \text{ is real} & k_3 - k_4 \text{ is purely imaginary} \ , \end{array}$$

from which equations (6.36) follow.

Finally, by using the relation

$$\frac{e^{jx} + e^{-jx}}{2} = \cos x \ ,$$

and simple trigonometry we obtain

$$\mathbf{v}(t) = \mathbf{q}_1 A_1 \cos(\omega_1 t + \phi_1) + \mathbf{q}_2 A_2 \cos(\omega_2 t + \phi_2)$$

where

$$\begin{aligned} \omega_1 &= \sqrt{\alpha - \gamma} = \sqrt{\frac{1}{2}(a+b) - \sqrt{\frac{1}{4}(a-b)^2 + \frac{c_2^2}{m_1 m_2}}} \\ \omega_2 &= \sqrt{\alpha + \gamma} = \sqrt{\frac{1}{2}(a+b) + \sqrt{\frac{1}{4}(a-b)^2 + \frac{c_2^2}{m_1 m_2}}} \end{aligned}$$

and

$$a = \frac{c_1 + c_2}{m_1} \quad , \quad b = \frac{c_2 + c_3}{m_2} \; .$$

Notice that these two frequencies depend only on the configuration of the system, and not on the initial conditions.

The amplitudes A_i and phases ϕ_i, on the other hand, depend on the constants k_i as follows:

$$\begin{aligned} A_1 &= |k_1| \; , & A_2 &= |k_3| \\ \phi_1 &= \arctan_2(\mathsf{Im}(k_1), \mathsf{Re}(k_1)) & \phi_2 &= \arctan_2(\mathsf{Im}(k_3), \mathsf{Re}(k_3)) \end{aligned}$$

where Re, Im denote the real and imaginary part and where the two-argument function $\arctan_2$ is defined as follows for $(x, y) \neq (0, 0)$

$$\arctan_2(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \pi + \arctan(\frac{y}{x}) & \text{if } x < 0 \\ \frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \end{cases}$$

and is undefined for $(x, y) = (0, 0)$. This function returns the arctangent of y/x (notice the order of the arguments) in the proper quadrant, and extends the function by continuity along the y axis.

The two constants k_1 and k_3 can be found from the given initial conditions $\mathbf{v}(0)$ and $\dot{\mathbf{v}}(0)$ from equations (6.35) and (6.25).

Chapter 7

Stochastic State Estimation

Perhaps the most important part of studying a problem in robotics or vision, as well as in most other sciences, is to determine a good model for the phenomena and events that are involved. For instance, studying manipulation requires defining models for how a robot arm can move and for how it interacts with the world. Analyzing image motion implies defining models for how points move in space and how this motion projects onto the image. When motion is involved, as is very often the case, models take on frequently the form of *dynamic systems*. A dynamic system is a mathematical description of a quantity that evolves over time. The theory of dynamic systems is both rich and fascinating. Although in this chapter we will barely scratch its surface, we will consider one of its most popular and useful aspects, the theory of state estimation, in the particular form of *Kalman filtering*. To this purpose, an informal definition of a dynamic system is given in the next section. The definition is then illustrated by setting up the dynamic system equations for a simple but realistic application, that of modeling the trajectory of an enemy mortar shell. In sections 7.3 through 7.5, we will develop the theory of the Kalman filter, and in section 7.6 we will see that the shell can be shot down before it hits us. As discussed in section 7.7, Kalman filtering has intimate connections with the theory of algebraic linear systems we have developed in chapters 2 and 3.

7.1 Dynamic Systems

In its most general meaning, the term *system* refers to some physical entity on which some action is performed by means of an input u. The system reacts to this input and produces an output y (see figure 7.1).

A *dynamic* system is a system whose phenomena occur over time. One often says that a system *evolves over time*. Simple examples of a dynamic system are the following:

- An electric circuit, whose input is the current in a given branch and whose output is a voltage across a pair of nodes.

- A chemical reactor, whose inputs are the external temperature, the temperature of the gas being supplied, and the supply rate of the gas. The output can be the temperature of the reaction product.

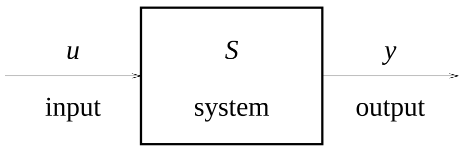

Figure 7.1: A general system.

- A mass suspended from a spring. The input is the force applied to the mass and the output is the position of the mass.

In all these examples, what is input and what is output is a choice that depends on the application. Also, all the quantities in the examples vary continuously with time. In other cases, as for instance for switching networks and computers, it is more natural to consider time as a discrete variable. If time varies continuously, the system is said to be *continuous*; if time varies discretely, the system is said to be *discrete*.

7.1.1 State

Given a dynamic system, continuous or discrete, the modeling problem is to somehow correlate inputs (causes) with outputs (effects). The examples above suggest that the output at time t cannot be determined in general by the value assumed by the input quantity at the same point in time. Rather, the output is the result of the entire history of the system. An effort of abstraction is therefore required, which leads to postulating a new quantity, called the *state*, which summarizes information about the past and the present of the system. Specifically, the value $\mathbf{x}(t)$ taken by the state at time t must be sufficient to determine the output at the same point in time. Also, knowledge of both $\mathbf{x}(t_1)$ and $\mathbf{u}_{[t_1,t_2)}$, that is, of the state at time t_1 and the input over the interval $t_1 \leq t < t_2$, must allow computing the state (and hence the output) at time t_2. For the mass attached to a spring, for instance, the state could be the position and velocity of the mass. In fact, the laws of classical mechanics allow computing the new position and velocity of the mass at time t_2 given its position and velocity at time t_1 and the forces applied over the interval $[t_1, t_2)$. Furthermore, in this example, the output $\mathbf{y}$ of the system happens to coincide with one of the two state variables, and is therefore always deducible from the latter.

Thus, in a dynamic system the input affects the state, and the output is a function of the state. For a discrete system, the way that the input changes the state at time instant number k into the new state at time instant $k+1$ can be represented by a simple equation:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, k)$$

where f is some function that represents the change, and $\mathbf{u}_k$ is the input at time k. Similarly, the relation between state and output can be expressed by another function:

$$\mathbf{y}_k = h(\mathbf{x}_k, k) \ .$$

A *discrete dynamic system* is completely described by these two equations and an initial state $\mathbf{x}_0$. In general, all quantities are vectors.

For continuous systems, time does not come in quanta, so one cannot compute $\mathbf{x}_{k+1}$ as a function of $\mathbf{x}_k$, $\mathbf{u}_k$, and k, but rather compute $\mathbf{x}(t_2)$ as a functional ϕ of $\mathbf{x}(t_1)$ and the entire input $\mathbf{u}$ over the interval $[t_1, t_2)$:

$$\mathbf{x}(t_2) = \phi(\mathbf{x}(t_1), \mathbf{u}(\cdot), t_1, t_2)$$

where $\mathbf{u}(\cdot)$ represents the entire function $\mathbf{u}$, not just one of its values. A description of the system in terms of functions, rather than functionals, can be given in the case of a *regular system*, for which the functional ϕ is continuous, differentiable, and with continuous first derivative. In that case, one can show that there exists a function f such that the state $\mathbf{x}(t)$ of the system satisfies the differential equation

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t)$$

where the dot denotes differentiation with respect to time. The relation from state to output, on the other hand, is essentially the same as for the discrete case:

$$\mathbf{y}(t) = h(\mathbf{x}(t), t) \ .$$

Specifying the initial state $\mathbf{x}_0$ completes the definition of a continuous dynamic system.

7.1.2 Uncertainty

The systems defined in the previous section are called *deterministic*, since the evolution is exactly determined once the initial state $\mathbf{x}$ at time 0 is known. Determinism implies that both the evolution function f and the output function h are known exactly. This is, however, an unrealistic state of affairs. In practice, the laws that govern a given physical system are known up to some uncertainty. In fact, the equations themselves are simple abstractions of a complex reality. The coefficients that appear in the equations are known only approximately, and can change over time as a result of temperature changes, component wear, and so forth. A more realistic model then allows for some inherent, unresolvable uncertainty in both f and h. This uncertainty can be represented as *noise* that perturbs the equations we have presented so far. A discrete system then takes on the following form:

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k, k) + \eta_k \\ \mathbf{y}_k &= h(\mathbf{x}_k, k) + \xi_k \end{aligned}$$

and for a continuous system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t), t) + \eta(t) \\ \mathbf{y}(t) &= h(\mathbf{x}(t), t) + \xi(t) \ . \end{aligned}$$

Without loss of generality, the noise distributions can be assumed to have zero mean, for otherwise the mean can be incorporated into the deterministic part, that is, in either f or h. The mean may not be known, but this is a different story: in general the parameters that enter into the definitions of f and h must be estimated by some method, and the mean perturbations are no different.

A common assumption, which is sometimes valid and always simplifies the mathematics, is that η and ξ are zero-mean Gaussian random variables with known covariance matrices Q and R, respectively.

7.1.3 Linearity

The mathematics becomes particularly simple when both the evolution function f and the output function h are linear. Then, the system equations become

$$\begin{aligned} \mathbf{x}_{k+1} &= F_k \mathbf{x}_k + G_k \mathbf{u}_k + \eta_k \\ \mathbf{y}_k &= H_k \mathbf{x}_k + \xi_k \end{aligned}$$

for the discrete case, and

$$\begin{aligned} \dot{\mathbf{x}}(t) &= F(t)\mathbf{x}(t) + G(t)\mathbf{u}(t) + \eta(t) \\ \mathbf{y}(t) &= H(t)\mathbf{x}(t) + \xi(t) \end{aligned}$$

for the continuous one. It is useful to specify the sizes of the matrices involved. We assume that the input $\mathbf{u}$ is a vector in $\mathcal{R}^p$, the state $\mathbf{x}$ is in $\mathcal{R}^n$, and the output $\mathbf{y}$ is in $\mathcal{R}^m$. Then, the *state propagation matrix* F is $n \times n$, the *input matrix* G is $n \times p$, and the *output matrix* H is $m \times n$. The covariance matrix Q of the *system noise* η is $n \times n$, and the covariance matrix of the output noise ξ is $m \times m$.

7.2 An Example: the Mortar Shell

In this section, the example of the mortar shell will be discussed in order to see some of the technical issues involved in setting up the equations of a dynamic system. In particular, we consider discretization issues because the physical system is itself continuous, but we choose to model it as a discrete system for easier implementation on a computer.

In sections 7.3 through 7.5, we consider the *state estimation* problem: given observations of the output $\mathbf{y}$ over an interval of time, we want to determine the state $\mathbf{x}$ of the system. This is a very important task. For instance, in the case of the mortar shell, the state is the (initially unknown) position and velocity of the shell, while the output is a set of observations made by a tracking system. Estimating the state then leads to enough knowledge about the shell to allow driving an antiaircraft gun to shoot the shell down in mid-flight.

You spotted an enemy mortar installation about thirty kilometers away, on a hill that looks about 0.5 kilometers higher than your own position. You want to track incoming projectiles with a Kalman filter so you can aim your guns accurately. You do not know the initial velocity of the projectiles, so you just guess some values: 0.6 kilometers/second for the horizontal component, 0.1 kilometers/second for the vertical component. Thus, your estimate of the initial state of the projectile is

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} \dot{d} \\ d \\ \dot{z} \\ z \end{bmatrix} = \begin{bmatrix} -0.6 \\ 30 \\ 0.1 \\ 0.5 \end{bmatrix}$$

where d is the horizontal coordinate, z is the vertical, you are at $(0,0)$, and dots denote derivatives with respect to time.

From your high-school physics, you remember that the laws of motion for a ballistic trajectory are the following:

$$d(t) = d(0) + \dot{d}(0)t \tag{7.1}$$

$$z(t) = z(0) + \dot{z}(0)t - \frac{1}{2}gt^2 \tag{7.2}$$

where g is the gravitational acceleration, equal to 9.8×10^{-3} kilometers per second squared. Since you do not trust your physics much, and you have little time to get ready, you decide to ignore air drag. Because of this, you introduce a state update covariance matrix $Q = 0.1 I_4$, where I_4 is the 4×4 identity matrix.

All you have to track the shells is a camera pointed at the mortar that will rotate so as to keep the projectile at the center of the image, where you see a blob that increases in size as the projectile gets closer. Thus, the aiming angle of the camera gives you elevation information about the projectile's position, and the size of the blob tells you something about the distance, given that you know the actual size of the projectiles used and all the camera parameters. The projectile's elevation is

$$e = 1000 \frac{z}{d} \tag{7.3}$$

when the projectile is at (d, z). Similarly, the size of the blob in pixels is

$$s = \frac{1000}{\sqrt{d^2 + z^2}} \; . \tag{7.4}$$

You do not have very precise estimates of the noise that corrupts e and s, so you guess measurement covariances $R_e = R_s = 1000$, which you put along the diagonal of a 2×2 diagonal measurement covariance matrix R.

7.2.1 The Dynamic System Equation

Equations (7.1) and (7.2) are continuous. Since you are taking measurements every $dt = 0.2$ seconds, you want to discretize these equations. For the z component, equation (7.2) yields

$$\begin{aligned} z(t+dt) - z(t) &= z(0) + \dot{z}(0)(t+dt) - \frac{1}{2}g(t+dt)^2 - \left[z(0) + \dot{z}(0)t - \frac{1}{2}gt^2 \right] \\ &= (\dot{z}(0) - gt)dt - \frac{1}{2}g(dt)^2 \\ &= \dot{z}(t)dt - \frac{1}{2}g(dt)^2 \; , \end{aligned}$$

since $\dot{z}(0) - gt = \dot{z}(t)$.

Consequently, if $t + dt$ is time instant $k + 1$ and t is time instant k, you have

$$z_{k+1} = z_k + \dot{z}_k dt - \frac{1}{2} g (dt)^2 \ . \tag{7.5}$$

The reasoning for the horizontal component d is the same, except that there is no acceleration:

$$d_{k+1} = d_k + \dot{d}_k dt \ . \tag{7.6}$$

Equations (7.5) and (7.6) can be rewritten as a single system update equation

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + Gu$$

where

$$\mathbf{x}_k = \begin{bmatrix} \dot{d}_k \\ d_k \\ \dot{z}_k \\ z_k \end{bmatrix}$$

is the state, the 4×4 matrix F depends on dt, the control scalar u is equal to $-g$, and the 4×1 control matrix G depends on dt. The two matrices F and G are as follows:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ dt & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & dt & 1 \end{bmatrix} \qquad G = \begin{bmatrix} 0 \\ 0 \\ dt \\ \frac{-dt^2}{2} \end{bmatrix} \ .$$

7.2.2 The Measurement Equation

The two nonlinear equations (7.3) and (7.4) express the available measurements as a function of the true values of the projectile coordinates d and z. We want to replace these equations with linear approximations. To this end, we develop both equations as Taylor series around the current estimate and truncate them after the linear term. From the elevation equation (7.3), we have

$$e_k = 1000\frac{z}{d} \approx 1000 \left[\frac{\hat{z}_k}{\hat{d}_k} + \frac{z - \hat{z}_k}{\hat{d}_k} - \frac{\hat{z}_k}{\hat{d}_k^2}(d - \hat{d}_k) \right] \ ,$$

so that after simplifying we can redefine the measurement to be the discrepancy from the estimated value:

$$e'_k = e_k - 1000\frac{\hat{z}_k}{\hat{d}_k} \approx 1000(\frac{z}{\hat{d}_k} - \frac{\hat{z}_k}{\hat{d}_k^2} d) \ . \tag{7.7}$$

We can proceed similarly for equation (7.4):

$$s_k = \frac{1000}{\sqrt{d^2 + z^2}} \approx \frac{1000}{\sqrt{\hat{d}_k^2 + \hat{z}_k^2}} - \frac{1000\hat{d}_k}{(\hat{d}_k^2 + \hat{z}_k^2)^{3/2}}(d - \hat{d}_k) - \frac{1000\hat{z}_k}{(\hat{d}_k^2 + \hat{z}_k^2)^{3/2}}(z - \hat{z}_k)$$

and after simplifying:

$$s'_k = s_k - \frac{2000}{\sqrt{\hat{d}^2 + \hat{z}^2}} \approx -1000 \left[\frac{\hat{d}_k}{(\hat{d}_k^2 + \hat{z}_k^2)^{3/2}} d + \frac{\hat{z}_k}{(\hat{d}_k^2 + \hat{z}_k^2)^{3/2}} z \right] \ . \tag{7.8}$$

The two measurements e'_k and s'_k just defined can be collected into a single measurement vector

$$\mathbf{y}_k = \begin{bmatrix} e'_k \\ s'_k \end{bmatrix} ,$$

and the two approximate measurement equations (7.7) and (7.8) can be written in the matrix form

$$\mathbf{y}_k = H_k \mathbf{x}_k \tag{7.9}$$

where the measurement matrix H_k depends on the current state estimate $\hat{\mathbf{x}}_k$:

$$H_k = -1000 \begin{bmatrix} 0 & \frac{\hat{d}_k}{(\hat{d}_k^2+\hat{z}_k^2)^{3/2}} & 0 & \frac{\hat{z}_k}{(\hat{d}_k^2+\hat{z}_k^2)^{3/2}} \\ 0 & \frac{\hat{z}_k}{\hat{d}_k^2} & 0 & -\frac{1}{\hat{d}_k} \end{bmatrix}$$

As the shell approaches us, we frantically start studying state estimation, and in particular Kalman filtering, in the hope to build a system that lets us shoot down the shell before it hits us. The next few sections will be read under this impending threat.

Knowing the model for the mortar shell amounts to knowing the laws by which the object moves and those that relate the position of the projectile to our observations. So what else is there left to do? From the observations, we would like to know where the mortar shell is right now, and perhaps predict where it will be in a few seconds, so we can direct an antiaircraft gun to shoot down the target. In other words, we want to know $\mathbf{x}_k$, the state of the dynamic system. Clearly, knowing $\mathbf{x}_0$ instead is equivalent, at least when the dynamics of the system are known exactly (the system noise η_k is zero). In fact, from $\mathbf{x}_0$ we can simulate the system up until time t, thereby determining $\mathbf{x}_k$ as well. Most importantly, we do not want to have all the observations before we shoot: we would be dead by then. A scheme that refines an initial estimation of the state as new observations are acquired is called a *recursive*[1] *state estimation* system. The *Kalman filter* is one of the most versatile schemes for recursive state estimations. The original paper by Kalman (R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME Journal Basic Engineering*, 82:34–45, 1960) is still one of the most readable treatments of this subject from the point of view of stochastic estimation.

Even without noise, a single observation $\mathbf{y}_k$ may not be sufficient to determine the state $\mathbf{x}_k$ (in the example, one observation happens to be sufficient). This is a very interesting aspect of state estimation. It is really the ensemble of all observations that let one estimate the state, and yet observations are processed one at a time, as they become available. A classical example of this situation in computer vision is the reconstruction of three-dimensional shape from a sequence of images. A single image is two-dimensional, so by itself it conveys no three-dimensional information. Kalman filters exist that recover shape information from a sequence of images. See for instance L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, 3(3):209-236, September 1989; and T.J. Broida, S. Chandrashekhar, and R. Chellappa, "Recursive 3-D motion estimation from a monocular image sequence," *IEEE Transactions on Aerospace and Electronic Systems*, 26(4):639–656, July 1990.

Here, we introduce the Kalman filter from the simpler point of view of least squares estimation, since we have developed all the necessary tools in the first part of this course. The next section defines the state estimation problem for a discrete dynamic system in more detail. Then, section 7.4 defines the essential notions of estimation theory that are necessary to understand the quantitative aspects of Kalman filtering. Section 7.5 develops the equation of the Kalman filter, and section 7.6 reconsiders the example of the mortar shell. Finally, section 7.7 establishes a connection between the Kalman filter and the solution of a linear system.

[1]The term "recursive" in the systems theory literature corresponds loosely to "incremental" or "iterative" in computer science.

7.3 State Estimation

In this section, the estimation problem is defined in some more detail. Given a discrete dynamic system

$$\mathbf{x}_{k+1} = F_k \mathbf{x}_k + G_k \mathbf{u}_k + \eta_k \tag{7.10}$$
$$\mathbf{y}_k = H_k \mathbf{x}_k + \xi_k \tag{7.11}$$

where the system noise η_k and the measurement noise ξ_k are Gaussian variables,

$$\begin{aligned} \eta_k &\sim \mathcal{N}(0, Q_k) \\ \xi_k &\sim \mathcal{N}(0, R_k) \ , \end{aligned}$$

as well as a (possibly completely wrong) estimate $\hat{\mathbf{x}}_0$ of the initial state and an initial covariance matrix P_0 of the estimate $\hat{\mathbf{x}}_0$, the Kalman filter computes the optimal estimate $\hat{\mathbf{x}}_{k|k}$ at time k given the measurements $\mathbf{y}_0, \ldots, \mathbf{y}_k$. The filter also computes an estimate $P_{k|k}$ of the covariance of $\hat{\mathbf{x}}_{k|k}$ given those measurements. In these expressions, the hat means that the quantity is an estimate. Also, the first k in the subscript refers to which variable is being estimated, the second to which measurements are being used for the estimate. Thus, in general, $\hat{\mathbf{x}}_{i|j}$ is the estimate of the value that $\mathbf{x}$ assumes at time i given the first $j+1$ measurements $\mathbf{y}_0, \ldots, \mathbf{y}_j$.

7.3.1 Update

The covariance matrix $P_{k|k}$ must be computed in order to keep the Kalman filter running, in the following sense. At time k, just before the new measurement $\mathbf{y}_k$ comes in, we have an estimate $\hat{\mathbf{x}}_{k|k-1}$ of the state vector $\mathbf{x}_k$ based on the previous measurements $\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}$. Now we face the problem of incorporating the new measurement $\mathbf{y}_k$ into our estimate, that is, of transforming $\hat{\mathbf{x}}_{k|k-1}$ into $\hat{\mathbf{x}}_{k|k}$. If $\hat{\mathbf{x}}_{k|k-1}$ were exact, we could *compute* the new measurement $\mathbf{y}_k$ without even looking at it, through the measurement equation (7.11). Even if $\hat{\mathbf{x}}_{k|k-1}$ is not exact, the estimate

$$\hat{\mathbf{y}}_{k|k-1} = H_k \hat{\mathbf{x}}_{k|k-1}$$

is still our best bet. Now $\mathbf{y}_k$ becomes available, and we can consider the *residue*

$$\mathbf{r}_k = \mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1} = \mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1} \ .$$

If this residue is nonzero, we probably need to correct our estimate of the state $\mathbf{x}_k$, so that the new prediction

$$\hat{\mathbf{y}}_{k|k} = H_k \hat{\mathbf{x}}_{k|k}$$

of the measurement value is closer to the old prediction

$$\hat{\mathbf{y}}_{k|k-1} = H_k \hat{\mathbf{x}}_{k|k-1}$$

we made just before the new measurement $\mathbf{y}_k$ was available.

The question however is, by how much should we correct our estimate of the state? We do not want to make $\hat{\mathbf{y}}_{k|k}$ *coincide* with $\mathbf{y}_k$. That would mean that we trust the new measurement completely, but that we do not trust our state estimate $\hat{\mathbf{x}}_{k|k-1}$ at all, even if the latter was obtained through a large number of previous measurements. Thus, we need some criterion for comparing the quality of the new measurement $\mathbf{y}_k$ with that of our old estimate $\hat{\mathbf{x}}_{k|k-1}$ of the state. The uncertainty about the former is R_k, the covariance of the observation error. The uncertainty about the state just before the new measurement $\mathbf{y}_k$ becomes available is $P_{k|k-1}$. The *update* stage of the Kalman filter uses R_k and $P_{k|k-1}$ to weigh past evidence ($\hat{\mathbf{x}}_{k|k-1}$) and new observations ($\mathbf{y}_k$). This stage is represented graphically in the middle of figure 7.2. At the same time, also the uncertainty measure $P_{k|k-1}$ must be updated, so that it becomes available for the next step. Because a new measurement has been read, this uncertainty becomes usually smaller: $P_{k|k} < P_{k|k-1}$.

The idea is that as time goes by the uncertainty on the state decreases, while that about the measurements may remain the same. Then, measurements count less and less as the estimate approaches its true value.

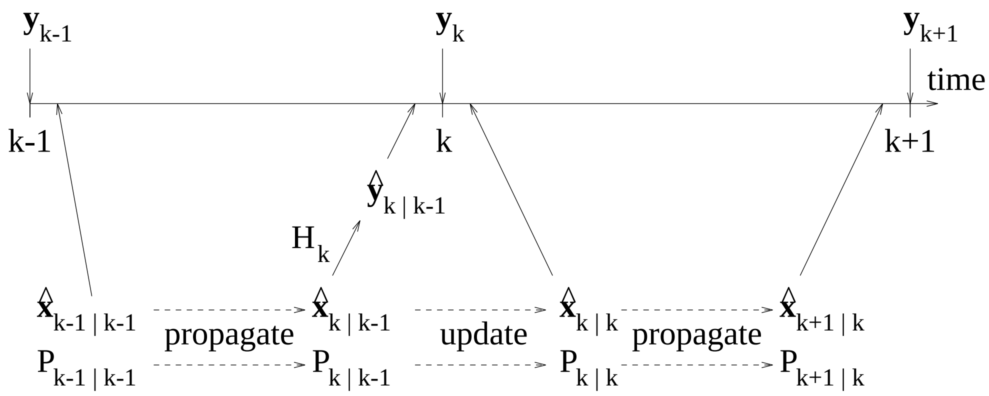

Figure 7.2: The update stage of the Kalman filter changes the estimate of the current system state $\mathbf{x}_k$ to make the prediction of the measurement closer to the actual measurement $\mathbf{y}_k$. Propagation then accounts for the evolution of the system state, as well as the consequent growing uncertainty.

7.3.2 Propagation

Just after arrival of the measurement $\mathbf{y}_k$, both state estimate and state covariance matrix have been updated as described above. But between time k and time $k+1$ both state and covariance may change. The state changes according to the system equation (7.10), so our estimate $\hat{\mathbf{x}}_{k+1|k}$ of $\mathbf{x}_{k+1}$ given $\mathbf{y}_0, \ldots, \mathbf{y}_k$ should reflect this change as well. Similarly, because of the system noise η_k, our uncertainty about this estimate may be somewhat greater than one time epoch ago. The system equation (7.10) essentially "dead reckons" the new state from the old, and inaccuracies in our model of how this happens lead to greater uncertainty. This increase in uncertainty depends on the system noise covariance Q_k. Thus, both state estimate and covariance must be *propagated* to the new time $k+1$ to yield the new state estimate $\hat{\mathbf{x}}_{k+1|k}$ and the new covariance $P_{k+1|k}$. Both these changes are shown on the right in figure 7.2.

In summary, just as the state vector $\mathbf{x}_k$ represents all the information necessary to describe the evolution of a deterministic system, the covariance matrix $P_{k|k}$ contains all the necessary information about the probabilistic part of the system, that is, about how both the system noise η_k and the measurement noise ξ_k corrupt the quality of the state estimate $\hat{\mathbf{x}}_{k|k}$.

Hopefully, this intuitive introduction to Kalman filtering gives you an idea of *what* the filter does, and what information it needs to keep working. To turn these concepts into a quantitative algorithm we need some preliminaries on optimal estimation, which are discussed in the next section. The Kalman filter itself is derived in section 7.5.

7.4 BLUE Estimators

In what sense does the Kalman filter use covariance information to produce better estimates of the state? As we will se later, the Kalman filter computes the *Best Linear Unbiased Estimate* (*BLUE*) of the state. In this section, we see what this means, starting with the definition of a linear estimation problem, and then considering the attributes "best" and "unbiased" in turn.

7.4.1 Linear Estimation

Given a quantity $\mathbf{y}$ (the *observation*) that is a known function of another (deterministic but unknown) quantity $\mathbf{x}$ (the *state*) plus some amount of noise,

$$\mathbf{y} = h(\mathbf{x}) + \mathbf{n} , \tag{7.12}$$

the estimation problem amounts to finding a function

$$\hat{\mathbf{x}} = \mathcal{L}(\mathbf{y})$$

such that $\hat{\mathbf{x}}$ is as close as possible to $\mathbf{x}$. The function $\mathcal{L}$ is called an *estimator*, and its value $\hat{\mathbf{x}}$ given the observations $\mathbf{y}$ is called an *estimate*. Inverting a function is an example of estimation. If the function h is invertible and the noise term $\mathbf{n}$ is zero, then $\mathcal{L}$ is the inverse of h, no matter how the phrase "as close as possible" is interpreted. In fact, in that case $\hat{\mathbf{x}}$ is equal to $\mathbf{x}$, and any distance between $\hat{\mathbf{x}}$ and $\mathbf{x}$ must be zero. In particular, solving a square, nonsingular system

$$\mathbf{y} = H\mathbf{x} \tag{7.13}$$

is, in this somewhat trivial sense, a problem of estimation. The optimal estimator is then represented by the matrix

$$L = H^{-1}$$

and the optimal estimate is

$$\hat{\mathbf{x}} = L\mathbf{y} \; .$$

A less trivial example occurs, for a linear observation function, when the matrix H has more rows than columns, so that the system (7.13) is overconstrained. In this case, there is usually no inverse to H, and again one must say in what sense $\hat{\mathbf{x}}$ is required to be "as close as possible" to $\mathbf{x}$. For linear systems, we have so far considered the criterion that prefers a particular $\hat{\mathbf{x}}$ if it makes the Euclidean norm of the vector $\mathbf{y} - H\mathbf{x}$ as small as possible. This is the *(unweighted) least squares* criterion. In section 7.4.2, we will see that in a very precise sense ordinary least squares solve a particular type of estimation problem, namely, the estimation problem for the observation equation (7.12) with h a linear function and $\mathbf{n}$ Gaussian zero-mean noise with the indentity matrix for covariance.

An estimator is said to be *linear* if the function $\mathcal{L}$ is linear. Notice that the observation function h can still be nonlinear. If $\mathcal{L}$ is required to be linear but h is not, we will probably have an estimator that produces a worse estimate than a nonlinear one. However, it still makes sense to look for the best possible linear estimator. The best estimator for a linear observation function happens to be a linear estimator.

7.4.2 Best

In order to define what is meant by a "best" estimator, one needs to define a measure of goodness of an estimate. In the least squares approach to solving a linear system like (7.13), this distance is defined as the Euclidean norm of the residue vector

$$\mathbf{y} - H\hat{\mathbf{x}}$$

between the left and the right-hand sides of equation (7.13), evaluated at the solution $\hat{\mathbf{x}}$. Replacing (7.13) by a "noisy equation",

$$\mathbf{y} = H\mathbf{x} + \mathbf{n} \tag{7.14}$$

does not change the nature of the problem. Even equation (7.13) has no exact solution when there are more independent equations than unknowns, so requiring equality is hopeless. What the least squares approach is really saying is that even at the solution $\hat{\mathbf{x}}$ there is some residue

$$\mathbf{n} = \mathbf{y} - H\hat{\mathbf{x}} \tag{7.15}$$

and we would like to make that residue as small as possible in the sense of the Euclidean norm. Thus, an overconstrained system of the form (7.13) and its "noisy" version (7.14) are really the same problem. In fact, (7.14) is the correct version, if the equality sign is to be taken literally.

The noise term, however, can be used to generalize the problem. In fact, the Euclidean norm of the residue (7.15) treats all components (all equations in (7.14)) equally. In other words, each equation counts the same when computing the norm of the residue. However, different equations can have noise terms of different variance. This amounts to saying that we have reasons to prefer the quality of some equations over others or, alternatively, that we want to enforce different equations to different degrees. From the point of view of least squares, this can be enforced by some scaling of the entries of $\mathbf{n}$ or, even, by some linear transformation of them:

$$\mathbf{n} \rightarrow W\mathbf{n}$$

so instead of minimizing $\|\mathbf{n}\|^2 = \mathbf{n}^T\mathbf{n}$ (the square is of course irrelevant when it comes to minimization), we now minimize

$$\|W\mathbf{n}\|^2 = \mathbf{n}^T R^{-1}\mathbf{n}$$

where

$$R^{-1} = W^T W$$

is a symmetric, nonnegative-definite matrix. This minimization problem, called *weighted least squares*, is only slightly different from its unweighted version. In fact, we have

$$W\mathbf{n} = W(\mathbf{y} - H\mathbf{x}) = W\mathbf{y} - WH\mathbf{x}$$

so we are simply solving the system

$$W\mathbf{y} = WH\mathbf{x}$$

in the traditional, "unweighted" sense. We know the solution from normal equations:

$$\hat{\mathbf{x}} = ((WH)^T WH)^{-1}(WH)^T W\mathbf{y} = (H^T R^{-1} H)^{-1} H^T R^{-1}\mathbf{y} \; .$$

Interestingly, this same solution is obtained from a completely different criterion of goodness of a solution $\hat{\mathbf{x}}$. This criterion is a probabilistic one. We consider this different approach because it will let us show that the Kalman filter is optimal in a very useful sense.

The new criterion is the so-called *minimum-covariance* criterion. The estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ is some function of the measurements $\mathbf{y}$, which in turn are corrupted by noise. Thus, $\hat{\mathbf{x}}$ is a function of a random vector (noise), and is therefore a random vector itself. Intuitively, if we estimate the same quantity many times, from measurements corrupted by different noise samples from the same distribution, we obtain different estimates. In this sense, the estimates are random.

It makes therefore sense to measure the quality of an estimator by requiring that its variance be as small as possible: the fluctuations of the estimate $\hat{\mathbf{x}}$ with respect to the true (unknown) value $\mathbf{x}$ from one estimation experiment to the next should be as small as possible. Formally, we want to choose a linear estimator L such that the estimates $\hat{\mathbf{x}} = L\mathbf{y}$ it produces minimize the following *covariance* matrix:

$$P = E[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T] \; .$$

Minimizing a matrix, however, requires a notion of "size" for matrices: how large is P? Fortunately, most interesting matrix norms are equivalent, in the sense that given two different definitions $\|P\|_1$ and $\|P\|_2$ of matrix norm there exist two positive scalars α, β such that

$$\alpha\|P\|_1 < \|P\|_2 < \beta\|P\|_1 \; .$$

Thus, we can pick any norm we like. In fact, in the derivations that follow, we only use properties shared by all norms, so which norm we actually use is irrelevant. Some matrix norms were mentioned in section 3.2.

7.4.3 Unbiased

In additionto requiring our estimator to be linear and with minimum covariance, we also want it to be *unbiased*, in the sense that if repeat the same estimation experiment many times we neither consistently overestimate nor consistently underestimate $\mathbf{x}$. Mathematically, this translates into the following requirement:

$$E[\mathbf{x} - \hat{\mathbf{x}}] = 0 \quad \text{and} \quad E[\hat{\mathbf{x}}] = E[\mathbf{x}] \; .$$

7.4.4 The BLUE

We now address the problem of finding the Best Linear Unbiased Estimator (BLUE)

$$\hat{\mathbf{x}} = L\mathbf{y}$$

of $\mathbf{x}$ given that $\mathbf{y}$ depends on $\mathbf{x}$ according to the model (7.13), which is repeated here for convenience:

$$\mathbf{y} = H\mathbf{x} + \mathbf{n} \; . \tag{7.16}$$

First, we give a necessary and sufficient condition for L to be unbiased.

Lemma 7.4.1 *Let $\mathbf{n}$ in equation (7.16) be zero mean. Then the linear estimator L is unbiased if an only if*

$$LH = I \; ,$$

the identity matrix.

Proof.

$$\begin{aligned} E[\mathbf{x} - \hat{\mathbf{x}}] &= E[\mathbf{x} - L\mathbf{y}] = E[\mathbf{x} - L(H\mathbf{x} + \mathbf{n})] \\ &= E[(I - LH)\mathbf{x}] - E[L\mathbf{n}] = (I - HL)E[\mathbf{x}] \end{aligned}$$

since $E[L\mathbf{n}] = L\,E[\mathbf{n}]$ and $E[\mathbf{n}] = 0$. For this to hold for all $\mathbf{x}$ we need $I - LH = 0$. Δ

And now the main result.

Theorem 7.4.2 *The Best Linear Unbiased Estimator (BLUE)*

$$\hat{\mathbf{x}} = L\mathbf{y}$$

for the measurement model

$$\mathbf{y} = H\mathbf{x} + \mathbf{n}$$

where the noise vector $\mathbf{n}$ has zero mean and covariance R is given by

$$L = (H^T R^{-1} H)^{-1} H^T R^{-1}$$

and the covariance of the estimate $\hat{\mathbf{x}}$ is

$$P = E[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T] = (H^T R^{-1} H)^{-1} \; . \tag{7.17}$$

Proof. We can write

$$\begin{aligned} P &= E[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T] = E[(\mathbf{x} - L\mathbf{y})(\mathbf{x} - L\mathbf{y})^T] \\ &= E[(\mathbf{x} - LH\mathbf{x} - L\mathbf{n})(\mathbf{x} - LH\mathbf{x} - L\mathbf{n})^T] = E[((I - LH)\mathbf{x} - L\mathbf{n})((I - LH)\mathbf{x} - L\mathbf{n})^T] \\ &= E[L\mathbf{n}\mathbf{n}^T L^T] = L\,E[\mathbf{n}\mathbf{n}^T]\,L^T = LRL^T \end{aligned}$$

because L is unbiased, so that $LH = I$.

To show that

$$L_0 = (H^T R^{-1} H)^{-1} H^T R^{-1} \tag{7.18}$$

is the best choice, let L be any (other) linear unbiased estimator. We can trivially write

$$L = L_0 + (L - L_0)$$

and

$$\begin{aligned} P &= LRL^T = [L_0 + (L - L_0)]R[L_0 + (L - L_0)]^T \\ &= L_0RL_0^T + (L - L_0)RL_0^T + L_0R(L - L_0)^T + (L - L_0)R(L - L_0)^T \; . \end{aligned}$$

From (7.18) we obtain

$$RL_0^T = RR^{-1}H(H^TR^{-1}H)^{-1} = H(H^TR^{-1}H)^{-1}$$

so that

$$(L - L_0)RL_0^T = (L - L_0)H(H^TR^{-1}H)^{-1} = (LH - L_0H)(H^TR^{-1}H)^{-1} \; .$$

But L and L_0 are unbiased, so $LH = L_0H = I$, and

$$(L - L_0)RL_0^T = 0 \; .$$

The term $L_0R(L - L_0)^T$ is the transpose of this, so it is zero as well. In conclusion,

$$P = L_0RL_0^T + (L - L_0)R(L - L_0)^T \; ,$$

the sum of two positive definite or at least semidefinite matrices. For such matrices, the norm of the sum is greater or equal to either norm, so this expression is minimized when the second term vanishes, that is, when $L = L_0$.

This proves that the estimator given by (7.18) is the best, that is, that it has minimum covariance. To prove that the covariance P of $\hat{\mathbf{x}}$ is given by equation (7.17), we simply substitute L_0 for L in $P = LRL^T$:

$$\begin{aligned} P &= L_0RL_0^T = (H^TR^{-1}H)^{-1}H^TR^{-1}RR^{-1}H(H^TR^{-1}H)^{-1} \\ &= (H^TR^{-1}H)^{-1}H^TR^{-1}H(H^TR^{-1}H)^{-1} = (H^TR^{-1}H)^{-1} \end{aligned}$$

as promised. Δ

7.5 The Kalman Filter: Derivation

We now have all the components necessary to write the equations for the Kalman filter. To summarize, given a linear measurement equation

$$\mathbf{y} = H\mathbf{x} + \mathbf{n}$$

where $\mathbf{n}$ is a Gaussian random vector with zero mean and covariance matrix R,

$$\mathbf{n} \sim \mathcal{N}(0, R) \; ,$$

the best linear unbiased estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ is

$$\hat{\mathbf{x}} = PH^TR^{-1}\mathbf{y}$$

where the matrix

$$P \stackrel{\Delta}{=} E[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T] = (H^TR^{-1}H)^{-1}$$

is the covariance of the estimation error.

Given a dynamic system with system and measurement equations

$$\begin{aligned} \mathbf{x}_{k+1} &= F_k\mathbf{x}_k + G_k\mathbf{u}_k + \eta_k \qquad (7.19) \\ \mathbf{y}_k &= H_k\mathbf{x}_k + \xi_k \end{aligned}$$

where the system noise η_k and the measurement noise ξ_k are Gaussian random vectors,

$$\begin{aligned} \eta_k &\sim \mathcal{N}(0, Q_k) \\ \xi_k &\sim \mathcal{N}(0, R_k) \,, \end{aligned}$$

as well as the best, linear, unbiased estimate $\hat{\mathbf{x}}_0$ of the initial state with an error covariance matrix P_0, the Kalman filter computes the best, linear, unbiased estimate $\hat{\mathbf{x}}_{k|k}$ at time k given the measurements $\mathbf{y}_0, \ldots, \mathbf{y}_k$. The filter also computes the covariance $P_{k|k}$ of the error $\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k$ given those measurements. Computation occurs according to the phases of update and propagation illustrated in figure 7.2. We now apply the results from optimal estimation to the problem of updating and propagating the state estimates and their error covariances.

7.5.1 Update

At time k, two pieces of data are available. One is the estimate $\hat{\mathbf{x}}_{k|k-1}$ of the state $\mathbf{x}_k$ given measurements up to but not including $\mathbf{y}_k$. This estimate comes with its covariance matrix $P_{k|k-1}$. Another way of saying this is that the estimate $\hat{\mathbf{x}}_{k|k-1}$ differs from the true state $\mathbf{x}_k$ by an error term $\mathbf{e}_k$ whose covariance is $P_{k|k-1}$:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{x}_k + \mathbf{e}_k \tag{7.20}$$

with

$$E[\mathbf{e}_k \mathbf{e}_k^T] = P_{k|k-1} \,.$$

The other piece of data is the new measurement $\mathbf{y}_k$ itself, which is related to the state $\mathbf{x}_k$ by the equation

$$\mathbf{y}_k = H_k \mathbf{x}_k + \xi_k \tag{7.21}$$

with error covariance

$$E[\xi_k \xi_k^T] = R_k \,.$$

We can summarize this available information by grouping equations 7.20 and 7.21 into one, and packaging the error covariances into a single, block-diagonal matrix. Thus, we have

$$\mathbf{y} = H\mathbf{x}_k + \mathbf{n}$$

where

$$\mathbf{y} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{y}_k \end{bmatrix} , \qquad H = \begin{bmatrix} I \\ H_k \end{bmatrix} , \qquad \mathbf{n} = \begin{bmatrix} \mathbf{e}_k \\ \mathbf{n}_k \end{bmatrix} ,$$

and where $\mathbf{n}$ has covariance

$$R = \begin{bmatrix} P_{k|k-1} & 0 \\ 0 & R_k \end{bmatrix} .$$

As we know, the solution to this classical estimation problem is

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= P_{k|k} H^T R^{-1} \mathbf{y} \\ P_{k|k} &= (H^T R^{-1} H)^{-1} \,. \end{aligned}$$

This pair of equations represents the update stage of the Kalman filter. These expressions are somewhat wasteful, because the matrices H and R contain many zeros. For this reason, these two update equations are now rewritten in a more efficient and more familiar form. We have

$$\begin{aligned} P_{k|k}^{-1} &= H^T R^{-1} H \\ &= \begin{bmatrix} I & H_k^T \end{bmatrix} \begin{bmatrix} P_{k|k-1}^{-1} & 0 \\ 0 & R_k^{-1} \end{bmatrix} \begin{bmatrix} I \\ H_k \end{bmatrix} \\ &= P_{k|k-1}^{-1} + H_k^T R_k^{-1} H_k \end{aligned}$$

and

$$
\begin{aligned}
\hat{\mathbf{x}}_{k|k} &= P_{k|k} H^T R^{-1} \mathbf{y} \\
&= P_{k|k} \left[\begin{array}{cc} P_{k|k-1}^{-1} & H_k^T R_k^{-1} \end{array} \right] \left[\begin{array}{c} \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{y}_k \end{array} \right] \\
&= P_{k|k} (P_{k|k-1}^{-1} \hat{\mathbf{x}}_{k|k-1} + H_k^T R_k^{-1} \mathbf{y}_k) \\
&= P_{k|k} ((P_{k|k}^{-1} - H_k^T R_k^{-1} H_k) \hat{\mathbf{x}}_{k|k-1} + H_k^T R_k^{-1} \mathbf{y}_k) \\
&= \hat{\mathbf{x}}_{k|k-1} + P_{k|k} H_k^T R_k^{-1} (\mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1}) \; .
\end{aligned}
$$

In the last line, the difference

$$\mathbf{r}_k \stackrel{\Delta}{=} \mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1}$$

is the *residue* between the actual measurement $\mathbf{y}_k$ and its best estimate based on $\hat{\mathbf{x}}_{k|k-1}$, and the matrix

$$K_k \stackrel{\Delta}{=} P_{k|k} H_k^T R_k^{-1}$$

is usually referred to as the *Kalman gain* matrix, because it specifies the amount by which the residue must be multiplied (or amplified) to obtain the correction term that transforms the old estimate $\hat{\mathbf{x}}_{k|k-1}$ of the state $\mathbf{x}_k$ into its new estimate $\hat{\mathbf{x}}_{k|k}$.

7.5.2 Propagation

Propagation is even simpler. Since the new state is related to the old through the system equation 7.19, and the noise term η_k is zero mean, unbiasedness requires

$$\hat{\mathbf{x}}_{k+1|k} = F_k \hat{\mathbf{x}}_{k|k} + G_k \mathbf{u}_k \; ,$$

which is the state estimate propagation equation of the Kalman filter. The error covariance matrix is easily propagated thanks to the linearity of the expectation operator:

$$
\begin{aligned}
P_{k+1|k} &= E[(\hat{\mathbf{x}}_{k+1|k} - \mathbf{x}_{k+1})(\hat{\mathbf{x}}_{k+1|k} - \mathbf{x}_{k+1})^T] \\
&= E[(F_k(\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k) - \eta_k)(F_k(\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k) - \eta_k)^T] \\
&= F_k E[(\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k)(\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k)^T] F_k^T + E[\eta_k \eta_k^T] \\
&= F_k P_{k|k} F_k^T + Q_k
\end{aligned}
$$

where the system noise η_k and the previous estimation error $\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k$ were assumed to be uncorrelated.

7.5.3 Kalman Filter Equations

In summary, the Kalman filter evolves an initial estimate and an initial error covariance matrix,

$$\hat{\mathbf{x}}_{0|-1} \stackrel{\Delta}{=} \hat{\mathbf{x}}_0 \quad \text{and } P_{0|-1} \stackrel{\Delta}{=} P_0 \; ,$$

both assumed to be given, by the update equations

$$
\begin{aligned}
\hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{y}_k - H_k \hat{\mathbf{x}}_{k|k-1}) \\
P_{k|k}^{-1} &= P_{k|k-1}^{-1} + H_k^T R_k^{-1} H_k
\end{aligned}
$$

where the Kalman gain is defined as

$$K_k = P_{k|k} H_k^T R_k^{-1}$$

and by the propagation equations

$$
\begin{aligned}
\hat{\mathbf{x}}_{k+1|k} &= F_k \hat{\mathbf{x}}_{k|k} + G_k \mathbf{u}_k \\
P_{k+1|k} &= F_k P_{k|k} F_k^T + Q_k \; .
\end{aligned}
$$

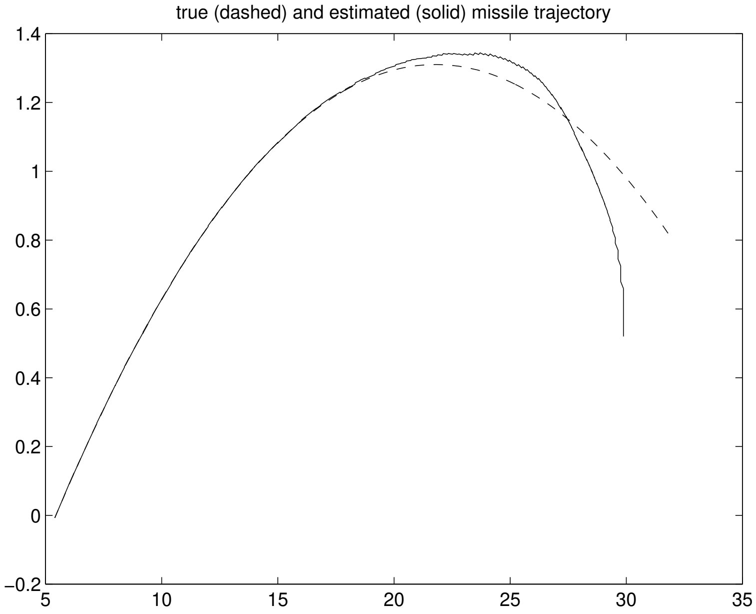

Figure 7.3: The true and estimated trajectories get closer to one another. Trajectories start on the right.

7.6 Results of the Mortar Shell Experiment

In section 7.2, the dynamic system equations for a mortar shell were set up. Matlab routines available through the class Web page implement a Kalman filter (with naive numerics) to estimate the state of that system from simulated observations. Figure 7.3 shows the true and estimated trajectories. Notice that coincidence of the trajectories does not imply that the state estimate is up-to-date. For this it is also necessary that any given point of the trajectory is reached by the estimate at the same time instant. Figure 7.4 shows that the distance between estimated and true target position does indeed converge to zero, and this occurs in time for the shell to be shot down. Figure 7.5 shows the 2-norm of the covariance matrix over time. Notice that the covariance goes to zero only asymptotically.

7.7 Linear Systems and the Kalman Filter

In order to connect the theory of state estimation with what we have learned so far about linear systems, we now show that estimating the initial state $\mathbf{x}_0$ from the first $k+1$ measurements, that is, obtaining $\hat{\mathbf{x}}_{0|k}$, amounts to solving a linear system of equations with suitable weights for its rows.

The basic recurrence equations (7.10) and (7.11) can be expanded as follows:

$$\begin{aligned}
\mathbf{y}_k &= H_k\mathbf{x}_k + \xi_k = H_k(F_{k-1}\mathbf{x}_{k-1} + G_{k-1}\mathbf{u}_{k-1} + \eta_{k-1}) + \xi_k \\
&= H_kF_{k-1}\mathbf{x}_{k-1} + H_kG_{k-1}\mathbf{u}_{k-1} + H_k\eta_{k-1} + \xi_k \\
&= H_kF_{k-1}(F_{k-2}\mathbf{x}_{k-2} + G_{k-2}\mathbf{u}_{k-2} + \eta_{k-2}) + H_kG_{k-1}\mathbf{u}_{k-1} + H_k\eta_{k-1} + \xi_k \\
&= H_kF_{k-1}F_{k-2}\mathbf{x}_{k-2} + H_k(F_{k-1}G_{k-2}\mathbf{u}_{k-2} + G_{k-1}\mathbf{u}_{k-1}) + \\
&\quad\; H_k(F_{k-1}\eta_{k-2} + \eta_{k-1}) + \xi_k \\
&\;\;\vdots \\
&= H_kF_{k-1}\ldots F_0\mathbf{x}_0 + H_k(F_{k-1}\ldots F_1G_0\mathbf{u}_0 + \ldots + G_{k-1}\mathbf{u}_{k-1}) + \\
&\quad\; H_k(F_{k-1}\ldots F_1\eta_0 + \ldots + \eta_{k-1}) + \xi_k
\end{aligned}$$

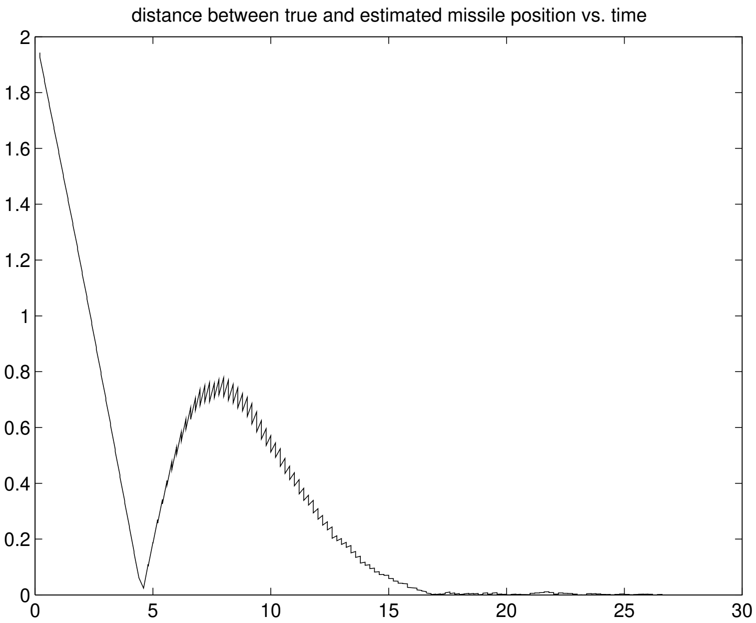

Figure 7.4: The estimate actually closes in towards the target.

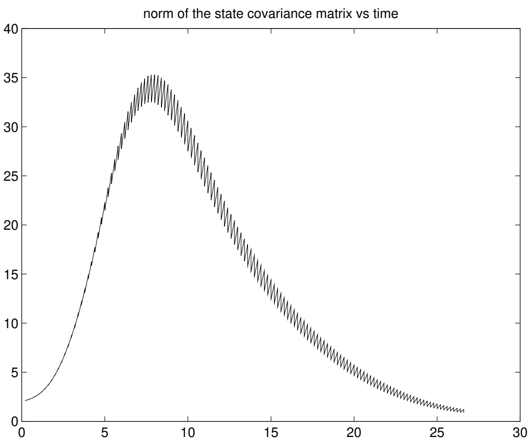

Figure 7.5: After an initial increase in uncertainty, the norm of the state covariance matrix converges to zero. Upwards segments correspond to state propagation, downwards ones to state update.

or in a more compact form,

$$\mathbf{y}_k = H_k \Phi(k-1,0)\mathbf{x}_0 + H_k \sum_{j=1}^{k} \Phi(k-1,j) G_{j-1}\mathbf{u}_{j-1} + \nu_k \tag{7.22}$$

where

$$\Phi(l,j) = \begin{cases} F_l \dots F_j & \text{for } l \geq j \\ 1 & \text{for } l < j \end{cases}$$

and the term

$$\nu_k = H_k \sum_{j=1}^{k} \Phi(k-1,j)\eta_{j-1} + \xi_k$$

is noise.

The key thing to notice about this somewhat intimidating expression is that for any k it is a linear system in $\mathbf{x}_0$, the initial state of the system. We can write one system like the one in equation (7.22) for every value of $k = 0, \dots, K$, where K is the last time instant considered, and we obtain a large system of the form

$$\mathbf{z}_K = \Psi_K \mathbf{x}_0 + \mathbf{g}_K + \mathbf{n}_K \tag{7.23}$$

where

$$\begin{aligned}
\mathbf{z}_K &= \begin{bmatrix} \mathbf{y}_0 \\ \vdots \\ \mathbf{y}_K \end{bmatrix} \\
\Psi_K &= \begin{bmatrix} 0 \\ H_1 F_0 \mathbf{u}_0 \\ \vdots \\ H_K \Phi(K-1,0) \end{bmatrix} \\
\mathbf{g}_K &= \begin{bmatrix} H_0 \\ H_1 F_0 G_0 \\ \vdots \\ H_K(\Phi(k-1,1)\mathbf{u}_0 + \dots + G_{K-1}\mathbf{u}_{K-1}) \end{bmatrix} \\
\mathbf{n}_K &= \begin{bmatrix} \nu_0 \\ \vdots \\ \nu_K \end{bmatrix} .
\end{aligned}$$

Without knowing anything about the statistics of the noise vector $\mathbf{n}_K$ in equation (7.23), the best we can do is to solve the system

$$\mathbf{z}_K = \Psi_K \mathbf{x}_0 + \mathbf{g}_K$$

in the sense of least squares, to obtain an estimate of $\mathbf{x}_0$ from the measurements $\mathbf{y}_0, \dots, \mathbf{y}_K$:

$$\hat{\mathbf{x}}_{0|K} = \Psi_K^{\dagger}(\mathbf{z}_K - \mathbf{g}_K)$$

where $\Psi_K^{\dagger}$ is the pseudoinverse of Ψ_K. We know that if Ψ_K has full rank, the result with the pseudoinverse is the same as we would obtain by solving the normal equations, so that

$$\Psi_K^{\dagger} = (\Psi_K^T \Psi_K)^{-1} \Psi_K^T .$$

The least square solution to system (7.23) minimizes the residue between the left and the right-hand side under the assumption that all equations are to be treated the same way. This is equivalent to assuming that all the noise terms in $\mathbf{n}_K$ are equally important. However, we know the covariance matrices of all these noise terms, so we ought to be able to do better, and weigh each equation to keep these covariances into account. Intuitively, a small covariance means that we believe in that measurement, and therefore in that equation, which should consequently be weighed more heavily than others. The quantitative embodiment of this intuitive idea is at the core of the Kalman filter.

In summary, the Kalman filter for a linear system has been shown to be equivalent to a linear equation solver, under the assumption that the noise that affects each of the equations has the same probability distribution, that is, that all the noise terms in $\mathbf{n}_K$ in equation 7.23 are equally important. However, the Kalman filter differs from a linear solver in the following important respects:

1. The noise terms in $\mathbf{n}_K$ in equation 7.23 are *not* equally important. Measurements come with covariance matrices, and the Kalman filter makes optimal use of this information for a proper weighting of each of the scalar equations in (7.23). Better information ought to yield more accurate results, and this is in fact the case.

2. The system (7.23) is not solved all at once. Rather, an initial solution is refined over time as new measurements become available. The final solution can be proven to be exactly equal to solving system (7.23) all at once. However, having better and better approximations to the solution as new data come in is much preferable in a dynamic setting, where one cannot in general wait for all the data to be collected. In some applications, data my never stop arriving.

3. A solution for the estimate $\hat{\mathbf{x}}_{k|k}$ of the current state is given, and not only for the estimate $\hat{\mathbf{x}}_{0|k}$ of the initial state. As time goes by, knowledge of the initial state may obsolesce and become less and less useful. The Kalman filter computes up-to-date information about the current state.