CS 296.4
Mathematical Modelling of Continuous Systems

Carlo Tomasi
Duke University

Fall 2002
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 most universities 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. 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.

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:

$$Ax = b$$

(2.1)

where $A$ is an $m \times n$ matrix, $x$ is an $n$-dimensional vector that collects all of the unknowns, and $b$ is a known vector of dimension $m$. In this chapter, we only consider the cases in which the entries of $A$, $b$, and $x$ 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 $f_1$ produces acceleration $a_1$ (both possibly vectors) and force $f_2$ produces acceleration $a_2$, then the combined force $f_1 + \alpha f_2$ produces acceleration $a_1 + \alpha a_2$. This is Newton’s second law of dynamics, although in a formulation less common than the equivalent $f = ma$. 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,

$$f(x, y) = 0$$
$$g(x, y) = 0,$$

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

$$F(x) = 0$$
$$y = h(x).$$

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 $a_1, \ldots, a_n$ and $n$ real numbers $x_1, \ldots, x_n$, the vector

\[ b = \sum_{j=1}^{n} x_j a_j \tag{2.2} \]

is said to be a linear combination of $a_1, \ldots, a_n$ with coefficients $x_1, \ldots, x_n$.

The vectors $a_1, \ldots, a_n$ 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

\[ \sum_{j=1}^{n} x_j a_j = 0. \tag{2.3} \]

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

\[ Ax = b \tag{2.4} \]

and equation (2.3) is the same as

\[ Ax = 0 \tag{2.5} \]

where

\[ A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}. \]

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 $a_1, \ldots, a_n$ are linearly dependent iff at least one of them is a linear combination of the others.

**Proof.** In one direction, dependency means that there is a nonzero vector $x$ such that

\[ \sum_{j=1}^{n} x_j a_j = 0. \]

Let $x_k$ be nonzero for some $k$. We have

\[ \sum_{j=1}^{n} x_j a_j = x_k a_k + \sum_{j=1, j \neq k}^{n} x_j a_j = 0 \]

so that

\[ a_k = - \sum_{j=1, j \neq k}^{n} \frac{x_j}{x_k} a_j \tag{2.6} \]

1 “iff” means “if and only if.”
as desired. The converse is proven similarly: if
\[ a_k = \sum_{j=1, j \neq k}^{n} x_j a_j \]
for some \( k \), then
\[ \sum_{j=1}^{n} x_j a_j = 0 \]
by letting \( x_k = -1 \) (so that \( x \) is nonzero). \( \triangle \)

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 \( a_1, \ldots, a_n \) 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
\[ a_k = \sum_{j<k}^{n} x_j a_j \]
as desired. \( \Delta \)

### 2.2 Basis

A set \( a_1, \ldots, a_n \) is said to be a *basis* for a set \( B \) of vectors if the \( a_j \) 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 \( b \) in the vector space \( B \) and a basis \( a_1, \ldots, a_n \) for \( B \), the coefficients \( x_1, \ldots, x_n \) such that
\[ b = \sum_{j=1}^{n} x_j a_j \]
are uniquely determined.

**Proof.** Let also
\[ b = \sum_{j=1}^{n} x'_j a_j . \]
Then,
\[ 0 = b - b = \sum_{j=1}^{n} x_j a_j - \sum_{j=1}^{n} x'_j a_j = \sum_{j=1}^{n} (x_j - x'_j) a_j \]
but because the \( a_j \) are linearly independent, this is possible only when \( x_j - x'_j = 0 \) for every \( j \). \( \Delta \)

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 $a_1, \ldots, a_n$ 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 $a_1, \ldots, a_n$ and $a'_1, \ldots, a'_n$ be two different bases for $B$. Then each $a'_j$ is in $B$ (why?), and can therefore be written as a linear combination of $a_1, \ldots, a_n$. Consequently, the vectors of the set

$$G = a'_1, a_1, \ldots, a_n$$

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 $a$ vectors from $G$ and replacing them with $a'$ vectors before we run out of $a'$ vectors, which proves that $n \geq n'$. We then switch the roles of $a$ and $a'$ 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 $a'_1$, since it has no other vectors preceding it. So it must be one of the $a_j$ vectors. Removing the latter keeps $G$ a generating set, since the removed vector depends on the others. Now we can add $a'_2$ to $G$, writing it right after $a'_1$:

$$G = a'_1, a'_2, \ldots$$

$G$ is still a generating set for $B$.

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

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

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

$$e_j = j\text{th column of the } m \times m \text{ identity matrix}$$

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

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

can be written as

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

and the $e_j$ are clearly independent. Since this elementary basis has $m$ vectors, theorem 2.2.2 implies that any other basis for $\mathbb{R}^m$ 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 $\mathbb{R}^m$ 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 $\theta$ 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 $a^2 = b^2 + c^2 - 2bc \cos \theta$.](image)

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

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

(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

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

(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

\[ b^T c = \|b\| \|c\| \cos \theta \]

where \( \theta \) is the angle between \( b \) and \( c \).

**Proof.** The law of cosines applied to the triangle with sides \( \|b\|, \|c\|, \) and \( \|b - c\| \) yields

\[ \|b - c\|^2 = \|b\|^2 + \|c\|^2 - 2\|b\| \|c\| \cos \theta \]

and from equation (2.8) we obtain

\[ b^T b + c^T c - 2b^T c = b^T b + c^T c - 2\|b\| \|c\| \cos \theta . \]

Canceling equal terms and dividing by -2 yields the desired result. \( \Delta \)

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

**Proof.** When \( \theta = \pm \pi/2 \), the previous theorem yields \( b^T c = 0 \). \( \Delta \)

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

![Figure 2.2: The vector from the origin to point \( p \) is the projection of \( b \) onto \( c \). The line from the endpoint of \( b \) to \( p \) is orthogonal to \( c \).](image)

**Theorem 2.3.3** The projection of \( b \) onto \( c \) is the vector 

\[ p = P_c b \]

where \( P_c \) is the following square matrix:

\[ P_c = \frac{cc^T}{c^T c} . \]

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

\[ c^T (b - ac) = 0 \]
which yields

\[ a = \frac{c^T b}{c^T c} \]

so that

\[ p = ac = c a = \frac{c c^T}{c^T c} b \]

as advertised. \( \triangle \)

## 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 \( \mathbb{R}^m \) for some \( m \), then the orthogonal complement of \( A \) is the set of all vectors in \( \mathbb{R}^m \) 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 \( \mathbb{R}^3 \) is all of \( \mathbb{R}^3 \) except the \( xy \) plane, while the orthogonal complement of the \( xy \) plane is the \( z \) axis.

Theorem 2.4.1 Any basis \( a_1, \ldots, a_n \) for a subspace \( A \) of \( \mathbb{R}^m \) can be extended into a basis for \( \mathbb{R}^m \) by adding \( m - n \) vectors \( a_{n+1}, \ldots, a_m \).

Proof. If \( n = m \) we are done. If \( n < m \), the given basis cannot generate all of \( \mathbb{R}^m \), so there must be a vector, call it \( a_{n+1} \), that is linearly independent of \( a_1, \ldots, a_n \). This argument can be repeated until the basis spans all of \( \mathbb{R}^m \), that is, until \( m = n \). \( \triangle \)

Theorem 2.4.2 (Gram-Schmidt) Given \( n \) vectors \( a_1, \ldots, a_n \), the following construction

\[
\begin{align*}
r &= 0 \\
&\text{for } j = 1 \text{ to } n \\
&\quad a'_j = a_j - \sum_{l=1}^{r} (q'_l a_j) q_l \\
&\quad \text{if } \|a'_j\| \neq 0 \\
&\quad r = r + 1 \\
&\quad q_r = \frac{a'_r}{\|a'_r\|} \\
&\end{align*}
\]

yields a set of orthonormal \( ^3 \) vectors \( q_1, \ldots, q_r \) that span the same space as \( a_1, \ldots, a_n \).

\( ^3 \)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

\[
\mathbf{q}_i^T \mathbf{a}_j = \mathbf{q}_i^T \mathbf{a}_j - \sum_{l=1}^{r} (\mathbf{q}_i^T \mathbf{a}_j) \mathbf{q}_l^T \mathbf{q}_l = 0
\]

because the term \( \mathbf{q}_i^T \mathbf{a}_j \) cancels the \( i \)-th term \( (\mathbf{q}_i^T \mathbf{a}_j) \mathbf{q}_l^T \mathbf{q}_l \) 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 \( \mathbf{q}_r = \mathbf{a}_r^\prime / \| \mathbf{a}_r^\prime \| \), 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 \( \mathbf{a}_1, \ldots, \mathbf{a}_n \).

\[\Delta\]

**Theorem 2.4.3** If \( A \) is a subspace of \( \mathbb{R}^m \) and \( A^\perp \) is the orthogonal complement of \( A \) in \( \mathbb{R}^m \), 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 \( \mathbb{R}^m \) (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 \( \mathbb{R}^m \). Thus, the dimension of the orthogonal space \( A^\perp \) is exactly \( m - n \), as promised.

\[\Delta\]

We can now start to talk about matrices in terms of the subspaces associated with them. The null space \( \text{null}(A) \) 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 \( A\mathbf{x} = \mathbf{0} \), and \( \mathbf{b} \in \text{range}(A) \) iff \( A\mathbf{x} = \mathbf{b} \) for some \( \mathbf{x} \).

From theorem 2.4.3, if \( \text{null}(A) \) 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 \( \text{range}(A) \).

Let \( \mathbf{v}_1, \ldots, \mathbf{v}_h \) be a basis for \( \text{null}(A) \), and extend this basis (theorem 2.4.1) into a basis \( \mathbf{v}_1, \ldots, \mathbf{v}_n \) for \( \mathbb{R}^n \). 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 \( Ax = b \). The \( n \)-tuple \( x \) itself, being an element of \( \mathbb{R}^n \), must be some linear combination of \( v_1, \ldots, v_n \), our basis for \( \mathbb{R}^n \):

\[
x = \sum_{j=1}^{n} c_j v_j .
\]

Thus,

\[
b = Ax = A \sum_{j=1}^{n} c_j v_j = \sum_{j=1}^{n} c_j Av_j = \sum_{j=h+1}^{n} c_j Av_j
\]

since \( v_1, \ldots, v_h \) span null(\( A \)), so that \( Av_j = 0 \) for \( j = 1, \ldots, h \). This proves that the \( n-h \) vectors \( Av_{h+1}, \ldots, Av_n \) generate range(\( A \)).

Second, we prove that the \( n-h \) vectors \( Av_{h+1}, \ldots, Av_n \) 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

\[
\sum_{j=h+1}^{n} x_j Av_j = 0
\]

so that

\[
A \sum_{j=h+1}^{n} x_j v_j = 0 .
\]

But then the vector \( \sum_{j=h+1}^{n} x_j v_j \) is in the null space of \( A \). Since the vectors \( v_1, \ldots, v_h \) are a basis for null(\( A \)), there must exist coefficients \( x_1, \ldots, x_h \) such that

\[
\sum_{j=h+1}^{n} x_j v_j = \sum_{j=1}^{h} x_j v_j ,
\]

in conflict with the assumption that the vectors \( v_1, \ldots, v_n \) are linearly independent.

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 \( h \):

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

The space \( \text{range}(A)^\perp \) is called the left nullspace of the matrix, and \( \text{null}(A)^\perp \) 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

\[
\text{null}(A)^\perp = \text{range}(A^T) \\
\text{range}(A)^\perp = \text{null}(A^T)
\]

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 $x$ in its null space into the zero vector, and an arbitrary vector $x$ into a vector in $\text{range}(A)$.

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

$$Ax = b$$

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,

$$b \notin \text{range}(A) \Rightarrow \text{no solutions}$$

$$b \in \text{range}(A) \Rightarrow \infty^{n-r} \text{ solutions}$$

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

In the first case above, there can be no linear combination of the columns (no $x$ vector) that gives $b$, 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 $x$ that satisfies the system, since the columns of $A$ span all of $\mathbb{R}^n$. Notice that invertibility depends only on $A$, not on $b$.

- **When $r = n$ and $m > n$, the system is *redundant***. There are more equations than unknowns, but since $b$ is in the range of $A$ there is a linear combination of the columns (a vector $x$) that produces $b$. 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 $x$ such that $Ax = 0$. Since $b$ is assumed to be in the range of $A$, there are solutions $x$ to $Ax = b$, but then for any $y \in \text{null}(A)$ also $x + y$ is a solution:

$$Ax = b, \ A y = 0 \Rightarrow A(x + y) = b$$

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 $A, b$. 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 $b \in \text{range}(A)$, 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.”
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 ??.

Consider the $m \times n$ system

$$Ax = b \quad (2.9)$$

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 $b$, which is transformed to a new vector $c$, 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 $x$ by backsubstitution, that is, by solving the transformed system

$$Ux = c.$$

### 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 $c^{(1)} = b$. The $k$-th step is applied to rows $k, \ldots, m$ of $U^{(k)}$ and $c^{(k)}$ and produces $U^{(k+1)}$ and $c^{(k+1)}$. The last step produces $U^{(m)} = U$ and $c^{(m)} = c$. 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 $i$. If $l \neq k$,

exchange rows $l$ and $k$ of $U^{(k)}$ and of $c^{(k)}$.

---

5“Stop” means that the entire algorithm is finished.

6Different 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 \( c^{(k)} \) multiplied by \( u_{ip}/u_{kp} \) from entry \( i \) of \( c^{(k)} \). 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

\[
U x = c \tag{2.10}
\]

in echelon form is easily solved for \( x \). 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:

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

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 \( x \) 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:

\[
\text{for } i = r \text{ downto 1} \quad x_{j_i} = \frac{1}{u_{ij_i}} \left( c_i - \sum_{l=j_i+1}^{n} u_{il} x_l \right) 
\]

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
\[ x = v_0 + x_{j_1}v_1 + \ldots + x_{j_{n-r}}v_{n-r} \]
(2.11)
where the \( x_{j_i} \) are the free variables. The vector \( v_0 \) 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 \( v_i \) for \( i = 1, \ldots, n - r \) can be obtained by solving the homogeneous system
\[ Ux = 0 \]

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).

2.6.3 An Example
An example will clarify both the reduction to echelon form and backsubstitution. Consider the system
\[ Ax = b \]

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

Reduction to echelon form transforms \( A \) and \( b \) 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 \( u^{(1)}_{11} = a_{11} = 1 \) is the pivot. In other words, no row exchange is necessary. 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 \( c^{(1)} \) this yields
\[
U^{(2)} = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 2 \end{bmatrix}, \quad c^{(2)} = \begin{bmatrix} 1 \\ 3 \\ 6 \end{bmatrix}.
\]

Notice that now \((k = 2)\) the entries \( u^{(2)}_{ip} \) 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 \( u^{(2)}_{33} \) 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 \( c^{(2)} \) to yield
\[
U = U^{(3)} = \begin{bmatrix} 1 & 3 & 2 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}, \quad c = c^{(3)} = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}.
\]

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
\[
\begin{bmatrix} 1 & 3 & 3 & 2 \\ 0 & 0 & 3 & 1 \end{bmatrix} x = \begin{bmatrix} 1 \\ 3 \end{bmatrix}
\]
(2.12)

\[^8\text{Selecting the largest entry in the column at or below row } k \text{ is a frequent choice, and this would have caused rows 1 and 2 to be switched.}\]
CHAPTER 2. ALGEBRAIC LINEAR SYSTEMS

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:

$$
x_3 = \frac{1}{u_{23}}(c_2 - u_{24}x_4) = \frac{1}{3}(3 - x_4) = 1 - \frac{1}{3}x_4$$

$$
x_1 = \frac{1}{u_{11}}(c_1 - u_{12}x_2 - u_{13}x_3 - u_{14}x_4) = \frac{1}{1}(1 - 3x_2 - 3x_3 - 2x_4) = 1 - 3x_2 - (3 - x_4) - 2x_4 = -2 - 3x_2 - x_4$$

so the general solution is

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

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

$$
x_3 = \frac{1}{3}(3 - 1 \cdot 0) = 1$$

$$
x_1 = \frac{1}{1}(1 - 3 \cdot 0 - 3 \cdot 1 - 2 \cdot 0) = -2$$

so that

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

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

$$
x_3 = \frac{1}{3}(-1 \cdot 0) = 0$$

$$
x_1 = \frac{1}{1}(-3 \cdot 1 - 3 \cdot 0 - 2 \cdot 0) = -3$$

so that

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

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

$$
x_3 = \frac{1}{3}(-1 \cdot 1) = -\frac{1}{3}$$

$$
x_1 = \frac{1}{1}(-3 \cdot 0 - 3 \cdot \left(-\frac{1}{3}\right) - 2 \cdot 1) = -1$$

so that

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

\[ x = v_0 + x_2 v_1 + x_4 v_2 = \begin{bmatrix} -2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} -3 \\ 1 \\ 0 \\ 0 \end{bmatrix} + x_4 \begin{bmatrix} -1 \\ 0 \\ -\frac{1}{3} \\ 1 \end{bmatrix} \]

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 \( 5 \times 5 \) 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.