CHAPTER 1

LINEAR COMBINATIONS

VECTORS AND MATRICES

1. Quantities

1.1. Numbers

Most scientific disciplines introduce an idea of the amount of some entity or property of interest. Furthermore, the amount is usually combined with the concept of a *number*, an abstraction of the observation that the two sets $A = \{\text{Mary, Jane, Tom}\}$ and $B = \{\text{apple, plum, cherry}\}$ seem quite different, but we can match one distinct person to one distinct fruit as in $\{\text{Mary}\longrightarrow\text{plum, Jane}\longrightarrow\text{apple, Tom}\longrightarrow\text{cherry}\}$. In contrast we cannot do the same matching of distinct persons to a distinct color from the set $\{\text{red, green}\}$, and one of the colors must be shared between two persons. Formal definition of the concept of a number from the above observations is surprisingly difficult since it would be self-referential due to the apperance of the numbers "one" and "two". Leaving this aside, the key concept is that of *quantity* of some property of interest that is expressed through a number. Several types of numbers have been introduced in mathematics to express different types of quantities, and the following will be used throughout this text:

- $\mathbb{N}$. The set of natural numbers, $\mathbb{N} = \{0, 1, 2, 3, \dots\}$, infinite and countable, $\mathbb{N}_+ = \{1, 2, 3, \dots\}$;
- $\mathbb{Z}$. The set of integers, $\mathbb{Z} = \{0, \pm 1, \pm 2, \pm 3, \dots\}$, infinite and countable;
- $\mathbf{Q}$. The set of rational numbers $\mathbb{Q} = \{p/q, p \in \mathbb{Z}, q \in \mathbb{N}_+\}$, infinite and countable;
- $\mathbb{R}$. The set of real numbers, infinite and not countable;
- $\mathbb{C}$. The set of complex numbers, $\mathbb{C} = \{x + iy, x, y \in \mathbb{R}\}$.

A computer has a finite amount of memory, hence cannot represent all numbers, but rather a subset of the above sets. Furthermore, computers internally use binary numbers composed of binary digits, or bits. Many computer number types are defined for specific purposes, and are often encountered in applications such as image representation or digital data acquisition. Here are the main types.

Subsets of $\mathbb{N}$. The number types `uint8`, `uint16`, `uint32`, `uint64` represent subsets of the natural numbers (unsigned integers) using 8,16,32, 64 bits respectively. An unsigned integer with b bits can store a natural number in the range from 0 to $2^b - 1$. Two arbitrary natural numbers, written as $\forall i, j \in \mathbb{N}$ can be added and will give another natural number, $k = i + j \in \mathbb{N}$. In contrast, addition of computer unsigned integers is only defined within the specific range 0 to $2^b - 1$.

```
octave] i=uint8(15); j=uint8(10); k=i+j
```

```
k = 25
```

```
octave] i=uint8(150); j=uint8(200); k=i+j
k = 255
octave] k=i-j
k = 0
octave]
```

Subsets of $\mathbb{Z}$. The number types `int8`, `int16`, `int32`, `int64` represent subsets of the integers. One bit is used to store the sign of the number, so the subset of $\mathbb{Z}$ that can be represented is from $1-2^{b-1}$ to $2^{b-1}-1$

```
octave] i=int8(100); j=int8(101); k=i+j
k = 127
octave] k=i-j
k = -1
octave]
```

Subsets of $\mathbb{Q}, \mathbb{R}, \mathbb{C}$. Computers approximate the real numbers through the set $\mathbb{F}$ of *floating point numbers*. Floating point numbers that use $b=32$ bits are known as *single precision*, while those that use $b=64$ are *double precision*. A floating point number $x \in \mathbb{F}$ is stored internally as $x = \pm.B_1B_2\ldots B_m \times 2^{\pm b_1 b_2 \ldots b_e}$ where B_i, $i=1,\ldots,m$ are bits within the *matissa* of length m, and b_j, $j=1,\ldots,e$ are bits within the *exponent*, along with signs $\pm$ for each. The default number type is usually double precision, more concisely referred to as simply `double`. Common constants such as e, π are predefined as `double`, can be truncated to `single`, and the number of displayed decimal digits is controlled by `format`.

```
octave] format long; disp([e pi])
   2.718281828459045   3.141592653589793
octave] disp([single(e) single(pi)])
   2.7182817   3.1415927
octave]
```

The approximation of the reals $\mathbb{R}$ by the floats $\mathbb{F}$ is characterized by: `realmax`, the largest float, `realmin` the smallest float in absolute value, and `eps` known as *machine epsilon*. Machine epsilon highlights the differences between floating point and real numbers since it is defined as the largest number $\epsilon \in \mathbb{F}$ that satisfies $1+\epsilon=1$. If $\varepsilon \in \mathbb{R}$ of course $1+\varepsilon=1$ implies $\varepsilon=0$, but floating points exhibit "granularity", in the sense that over a unit interval there are small steps that are indistinguishable from zero due to the finite number of bits available for a float. Machine epsilon is small, and floating point errors can usually be kept under control. Keep in mind that perfect accuracy is a mathematical abstraction, not encountered in nature. In fields as sociology or psychology 3 digits of accuracy are excellent, in mechanical engineering this might increase to 6 digits, or in electronic engineering to 8 digits. The most precisely known physical constant is the Rydberg constant known to 12 digits. The granularity of double precision expressed by machine epsilon is sufficient to represent natural phenomena.

```
octave] format short; disp([realmin realmax eps 1+eps])
   2.2251e-308   1.7977e+308   2.2204e-16   1.0000e+00
octave]
```

Within the reals certain operations are undefined such as $1/0$. Special float constants are defined to handle such situations: `Inf` is a float meant to represent infinity, and `NaN` ("not a number") is meant to represent an undefinable result of an arithmetic operation.

```
octave] warning("off"); disp([Inf 1/0 2*realmax NaN Inf-Inf Inf/Inf])
   Inf   Inf   Inf   NaN   NaN   NaN
octave]
```

Complex numbers $z \in \mathbb{C}$ are specified by two reals, in Cartesian form as $z = x + iy$, $x, y \in \mathbb{R}$ or in polar form as $z = \rho e^{i\theta}$, $\rho, \theta \in \mathbb{R}$, $\rho \geq 0$. The computer type complex is similarly defined from two floats and the additional constant I is defined to represent $\sqrt{-1} = i = e^{i\pi/2}$. Functions are available to obtain the real and imaginary parts within the Cartesian form, or the absolute value and argument of the polar form.

```
octave] z1=complex(1,1); z2=complex(1,-1); disp([z1+z2 z1/z2])
   2 + 0i   0 + 1i
octave] disp([real(z1) real(z2) real(z1+z2) real(z1/z2)])
   1   1   2   0
octave] disp([imag(z1) imag(z2) imag(z1+z2) imag(z1/z2)])
   1  -1   0   1
octave] disp([abs(z1) abs(z2) abs(z1+z2) abs(z1/z2)])
   1.4142   1.4142   2.0000   1.0000
octave] disp([arg(z1) arg(z2) arg(z1+z2) arg(z1/z2)])
```

Care should be exercised about the cummulative effect of many floating point errors. For instance, in an "irrational" numerical investigation of Zeno's paradox, one might want to compare the distance traversed S_N by step sizes that are scaled by $1/\pi$ starting from one to T_N, that traversed by step sizes that are scaled by π starting from π^{-N}

$$S_N = 1 + \frac{1}{\pi} + \frac{1}{\pi^2} + \cdots + \frac{1}{\pi^N}, \; T_N = \frac{1}{\pi^N} + \frac{1}{\pi^{N-1}} + \cdots + 1.$$

In the reals the above two expressions are equal $S_N = T_N$, but this is not verfied for all N when using floating point numbers.

```
octave] N=10; S=pi.^(0:-1:-N); T=pi.^(-N:1:0); sum(S)==sum(T)
 ans = 1
octave] N=30; S=pi.^(0:-1:-N); T=pi.^(-N:1:0); sum(S)==sum(T)
 ans = 0
octave]
```

In the above numerical experiment a==b expresses an equality relationship which might evaluate as true denoted by 1, or false denoted by 0.

```
octave] disp([1==1 1==2])
   1   0
octave]
```

The above was called an "irrational" investigation since in Zeno's original paradox the scaling factor was 2 rather than π, and given the binary representation used by floats equality always holds.

```
octave] N=30; S=2.^(0:-1:-N); T=2.^(-N:1:0); sum(S)==sum(T)
 ans = 1
```

```
octave]
```

1.2. Quantities described by a single number

The above numbers and their computer approximations are sufficient to describe many quantities encountered in applications. Typical examples include:

- the position $x \in \mathbb{R}$ of a point on the unit line segment $[0,1]$, approximated by the floating point number $\tilde{x} \in \mathbb{F}$, to within machine epsilon precision, $|x - \tilde{x}| \leq \epsilon$;
- the measure of resistance to change of the rate of motion known as *mass*, $m \in \mathbb{R}$, $m > 0$;
- the population of a large community expressed as a float $p \in \mathbb{F}$, even though for a community of individuals the population is a natural number, as in "the population of the United States is $p = 328.2\text{E}6$, i.e., 328.2 million".

In most disciplines, there is a particular interest in comparison of two quantities, and to facilitate such comparison a common reference is used known as a *standard unit*. For measurement of a length L, the meter $\ell = 1\,\text{m}$ is a standard unit, as in the statement $L = 10\,\text{m}$, that states that L is obtained by taking the standard unit ten times, $L = 10\ell$. The rules for carrying out such comparisons are part of the definition of real and rational numbers. These rules are formalized in the mathematical definition of a *field* $(F, +, \times)$ presented in the next chapter. Quantities that obey such rules, i.e., belong to a field, allow for changes of scale and are called *scalars*. Not all numbers are scalars in this sense. For instance, the integers would not allow a scaling of 1:2 (halving the scale) even though 1,2 are integers.

1.3. Quantities described by multiple numbers

Other quantities require more than a single number. The distribution of population in the year 2000 among the alphabetically-ordered South American countries (Argentina, Bolivia,..,Venezuela) requires 12 numbers. These are placed together in a list known in mathematics as a *tuple*, in this case a 12-tuple $P = (p_1, p_2, \ldots, p_{12})$, with p_1 the population of Argentina, p_2 that of Bolivia, and so on. An analogous 12-tuple can be formed from the South American populations in the year 2020, say $Q = (q_1, q_2, ..., q_{12})$. Note that it is difficult to ascribe meaning to apparently plausible expressions such as $P + Q$ since, for instance, some people in the 2000 population are also in the 2020 population, and would be counted twice.

2. Vectors

In contrast to the population 12-tuple example above, combining multiple numbers is well defined in operations such as specifying a position within a three-dimensional Cartesian grid, or determining the resultant of two forces in space. Both of these lead to the consideration of 3-tuples or *triples* such as the force (f_1, f_2, f_3). When combined with another force (g_1, g_2, g_3) the resultant is $(f_1 + g_1, f_2 + g_2, f_3 + g_3)$. If the force (f_1, f_2, f_3) is amplified by the scalar α and the force (g_1, g_2, g_3) is similarly scaled by β, the resultant becomes

$$\alpha(f_1, f_2, f_3) + \beta(g_1, g_2, g_3) = (\alpha f_1, \alpha f_2, \alpha f_3) + (\beta g_1, \beta g_2, \beta g_3) = (\alpha f_1 + \beta g_1, \alpha f_2 + \beta g_2, \alpha f_3 + \beta g_3).$$

It is useful to distinguish tuples for which scaling and addition is well defined from simple lists of numbers. In fact, since the essential difference is the behavior with respect to scaling and addition, the focus should be on these operations rather than the elements of the tuple.

The above observations underlie the definition of a *vector space* as a set V whose elements satisfy certain scaling and addition properties, denoted all together by the 4-tuple $(V, S, +, \cdot)$. The first element of the 4-tuple is a set whose elements are called *vectors*. The second element is a set of scalars, and the third is the vector addition operation. The last is the scaling operation, seen as multiplication of a vector by a scalar. The vector addition and scaling operations must satisfy rules suggested by positions or forces in three-dimensional space, which are listed in Table ?. In particular, a vector space requires definition of two distinguished elements: the zero vector $\mathbf{0} \in V$, and the identity scalar element $1 \in S$.

Addition rules for	$\forall \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in V$
$\boldsymbol{a} + \boldsymbol{b} \in V$	Closure
$\boldsymbol{a} + (\boldsymbol{b} + \boldsymbol{c}) = (\boldsymbol{a} + \boldsymbol{b}) + \boldsymbol{c}$	Associativity
$\boldsymbol{a} + \boldsymbol{b} = \boldsymbol{b} + \boldsymbol{a}$	Commutativity
$\mathbf{0} + \boldsymbol{a} = \boldsymbol{a}$	Zero vector
$\boldsymbol{a} + (-\boldsymbol{a}) = \mathbf{0}$	Additive inverse
Scaling rules for	$\forall \boldsymbol{a}, \boldsymbol{b} \in V, \forall x, y \in S$
$x\boldsymbol{a} \in V$	Closure
$x(\boldsymbol{a} + \boldsymbol{b}) = x\boldsymbol{a} + x\boldsymbol{b}$	Distributivity
$(x + y)\boldsymbol{a} = x\boldsymbol{a} + y\boldsymbol{a}$	Distributivity
$x(y\boldsymbol{a}) = (xy)\boldsymbol{a}$	Composition
$1\boldsymbol{a} = \boldsymbol{a}$	Scalar identity

Table 1.1. Properties of vector space $(V, S, +, \cdot)$ for arbitrary $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in V$

The definition of a vector space reflects everyday experience with vectors in Euclidean geometry, and it is common to refer to such vectors by descriptions in a Cartesian coordinate system. For example, a position vector $\boldsymbol{r}$ within the plane can be referred through the pair of coordinates (x, y). This intuitive understanding can be made precise through the definition of a vector space $E_2 = (\mathbb{R}^2, \mathbb{R}, +, \cdot)$, called the *Euclidean 2-space*. Vectors within E_2 are elements of $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R} = \{(x, y) \mid x, y \in \mathbb{R}\}$, meaning that a vector is specified through two real numbers, $\boldsymbol{r} \leftrightarrow (x, y)$. Addition of two vectors, $\boldsymbol{q} \leftrightarrow (s, t)$, $\boldsymbol{r} \leftrightarrow (x, y)$ is defined by addition of coordinates $\boldsymbol{q} + \boldsymbol{r} = (s + x, t + v)$. Scaling $\boldsymbol{r} \leftrightarrow (x, y)$ by scalar a is defined by $a\boldsymbol{r} \leftrightarrow (ax, ay)$. Similarly, consideration of position vectors in three-dimensional space leads to the definition of the *Euclidean 3-space* $E_3 = (\mathbb{R}^3, \mathbb{R}, +, \cdot)$, or more generally an Euclidean m-space $E_m = (\mathbb{R}^m, \mathbb{R}, +, \cdot)$, $m \in \mathbb{N}$, $m > 0$.

Note however that there is no mention of coordinates in the definition of a vector space as can be seen from the list of properties in Table 1.1. The intent of such a definition is to highlight that besides Euclidean vectors, many other mathematical objects follow the same rules. As an example, consider the set of all continuous functions $C(\mathbb{R}) = \{f \mid f : \mathbb{R} \longrightarrow \mathbb{R}\}$, with function addition defined by the sum at each argument x, $(f + g)(x) = f(x) + g(x)$, and scaling by $a \in \mathbb{R}$ defined as $(af)(x) = af(x)$. Read this as: "given two continuous functions f and g, the function $f + g$ is defined by stating that its value for argument x is the sum of the two real numbers $f(x)$ and $g(x)$". Similarly: "given a continuous function f, the function af is defined by stating that its value for argument x is the product of the real numbers a and $f(x)$". Under such definitions $(C(\mathbb{R}), \mathbb{R}, +, \cdot)$ is a vector space, but quite different from E_m. Nonetheless, the fact that both $C(\mathbb{R})$ and E_m are vector spaces can be used to obtain insight into the behavior of continuous functions from Euclidean vectors, and vice versa.

Since the Euclidean spaces $E_m = (\mathbb{R}^m, \mathbb{R}, +, \cdot)$ play such an important role in themselves and as a guide to other vector spaces, familiarity with vector operations in E_m is necessary to fully appreciate the utility of linear algebra to a wide range of applications. Note that E_1 corresponds to Following the usage in geometry and physics, the m real numbers that specify a vector $\boldsymbol{u} \in E_m$ are called the *components* of $\boldsymbol{u}$. The one-to-one correspondence between a vector and its components $\boldsymbol{u} \leftrightarrow (u_1, \ldots, u_m)$, is by convention taken to define an equality relationship,

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix},$$

with the components arranged vertically and enclosed in square brackets. To aid in visual recognition of vectors, the following notation conventions are introduced:

- vectors are denoted by lower-case bold

In Octave, successive components placed vertically are separated by a semicolon.

```
octave] [1; 0; -1; 2]
ans =

   1
   0
  -1
   2

octave]
```

The equal sign in mathematics signifies a particular equivalence relationship. In computer systems such as Octave the equal sign has the different meaning of *assignment*, that is defining the label on the left side of the equal sign to be the expression on the right side. Subsequent invocation of the label returns the assigned object.

```
octave] u=[1; 0; -1; 2]; u
u =

   1
   0
  -1
   2

octave]
```

Instead of the vertical placement or components into one *column*, the components of could have been placed horizontally in one *row* $[\ u_1\ \ldots\ u_m\ ]$, that contains the same data, differently organized. The transpose operation denoted by a T superscript is introduced to relate the two representations

$$\boldsymbol{u}^T = [\ u_1\ \ldots\ u_m\ ].$$

In Octave, horizontal placement of successive components is denoted by a space.

```
octave] uT=transpose(u)
uT =

   1   0  -1   2

octave] [1 0 -1 2]
ans =

   1   0  -1   2

octave]
```

By conventi

Though the same components might be present in both $\boldsymbol{u}$ and $\boldsymbol{u}^T$, the different organization means that the two vectors are not equal for $m > 1$ components, $\boldsymbol{u} \in E_m, m > 1 \Longrightarrow \boldsymbol{u} \neq \boldsymbol{u}^T$.

```
octave] u.+uT
ans =

   2   1   0   3
   1   0  -1   2
   0  -1  -2   1
   3   2   1   4

octave]
```

3. Matrices

MATRIX OPERATIONS