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 $A =$ {Mary, Jane, Tom} and $B =$ {apple, plum, cherry} seem quite different, but we can match one distinct person to one distinct fruit as in {Mary→plum, Jane→apple, Tom→cherry}. In contrast we cannot do the same matching of distinct persons to a distinct color from the set {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;
- $\mathbb{Q}$. 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, not countable, can be ordered;
- $\mathbb{C}$. The set of complex numbers, $\mathbb{C} = \{x + iy, x, y \in \mathbb{R}\}$, infinite, not countable, cannot be ordered.

A computer has a finite amount of memory, hence cannot represent all numbers, but rather subsets 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 *mantissa* 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 `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`. The function `disp(x)` displays its argument `x`.

```
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 positive float, 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 $z = \rho e^{i\theta}$, $\rho, \theta \in \mathbb{R}$, $\rho \geqslant 0$. 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)])
   0.78540  -0.78540   0.00000   1.57080
octave] I-sqrt(-1)
ans = 0
octave]
```

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 S_N traversed by step sizes that are scaled by $1/\pi$ starting from one to T_N, traversed by step sizes 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. Lists of the values π^j, for the two orderings $j = 0, \ldots, N$, and $j = N, \ldots, 0$, can be generated and summed.

```
octave] N=10; S=pi.^(0:-1:-N); T=pi.^(-N:1:0); sum(S)==sum(T)
ans = 1
octave] N=15; 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 due to 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, $|x - \tilde{x}| \leqslant \epsilon$;
- 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, can be used in 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 *tupl*e, 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 $Q = (q_1, q_2, \ldots, q_{12})$. 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

2.1. Vector spaces

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* $\mathscr{V}$ by a set V whose elements satisfy certain scaling and addition properties, denoted all together by the 4-tuple $\mathscr{V} = (V, S, +, \cdot)$. 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 1. 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. Vector space $\mathscr{V} = (V, S, +, \cdot)$ properties 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 $\mathscr{R}_2 = (\mathbb{R}^2, \mathbb{R}, +, \cdot)$, called the real 2-space. Vectors within $\mathscr{R}_2$ are elements of $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R} = \{(x, y) | x, y \in \mathbb{R}\}$, 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 $\mathscr{R}_3 = (\mathbb{R}^3, \mathbb{R}, +, \cdot)$, or more generally a real m-space $\mathscr{R}_m = (\mathbb{R}^m, \mathbb{R}, +, \cdot)$, $m \in \mathbb{N}$, $m > 0$.

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. The intent of such a definition is to highlight that besides position vectors, many other mathematical objects follow the same rules. As an example, consider the set of all continuous functions $C(\mathbb{R}) = \{f \mid f\colon \mathbb{R} \to \mathbb{R}\}$, with function addition defined by the sum at each argument t, $(f+g)(t) = f(t) + g(t)$, and scaling by $a \in \mathbb{R}$ defined as $(af)(t) = af(t)$. 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 $f(t)$ and $g(t)$". Similarly: "given a continuous function f, the function af is defined by stating that its value for argument t is the product of the real numbers a and $f(t)$". Under such definitions $\mathscr{C}^0 = (C(\mathbb{R}), \mathbb{R}, +, \cdot)$ is a vector space, but quite different from $\mathscr{R}_m$. Nonetheless, the fact that both $\mathscr{C}^0$ and $\mathscr{R}_m$ are vector spaces can be used to obtain insight into the behavior of continuous functions from Euclidean vectors, and vice versa.

2.2. Real vector space $\mathscr{R}_m$

Column vectors. Since the real spaces $\mathscr{R}_m = (\mathbb{R}^m, \mathbb{R}, +, \cdot)$ play such an important role in themselves and as a guide to other vector spaces, familiarity with vector operations in $\mathscr{R}_m$ is necessary to fully appreciate the utility of linear algebra to a wide range of applications. Following the usage in geometry and physics, the m real numbers that specify a vector $\boldsymbol{u} \in \mathbb{R}^m$ 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,

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix}, \tag{1}$$

with the components arranged vertically and enclosed in square brackets. Given two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^m$, and a scalar $a \in \mathbb{R}$, vector addition and scaling are defined in $\mathscr{R}_m$ by real number addition and multiplication of components

$$\boldsymbol{u} + \boldsymbol{v} = \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} u_1 + v_1 \\ \vdots \\ u_m + v_m \end{bmatrix}, a\boldsymbol{u} = a \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} = \begin{bmatrix} au_1 \\ \vdots \\ au_m \end{bmatrix}. \tag{2}$$

The vector space $\mathscr{R}_m$ is defined using the real numbers as the set of scalars, and constructing vectors by grouping together m scalars, but this approach can be extended to any set of scalars S, leading to the definition of the vector spaces $\mathscr{S}_n = (S^n, S, +, \cdot)$. These will often be referred to as *n-vector space of scalars*, signifying that the set of vectors is $V = S^n$.

To aid in visual recognition of vectors, the following notation conventions are introduced:

- vectors are denoted by lower-case bold Latin letters: $\boldsymbol{u}, \boldsymbol{v}$;
- scalars are denoted by normal face Latin or Greek letters: a, b, α, β;
- the components of a vector are denoted by the corresponding normal face with subscripts as in equation (1);
- related sets of vectors are denoted by indexed bold Latin letters: $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n$.

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

```
octave] [1; 2; -1; 2]
```

```
ans =

   1
   2
  -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. Components of a vector are obtained by enclosing the index in parantheses.

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

   1
   2
  -1
   2

octave] u(3)
ans = -1
octave]
```

Row vectors. 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. By convention vertical placement of vector components is the preferred organization, and $\boldsymbol{u}$ shall denote a *column vector* henceforth. A transpose operation denoted by a T superscript is introduced to relate the two representations

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

and $\boldsymbol{u}^T$ is the notation used to denote a *row vector*. In Octave, horizontal placement of successive components in a row is denoted by a space.

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

   1   2  -1   2

octave] [1 2 -1 2]
ans =

   1   2  -1   2

octave] uT(4)
ans =  2
octave]
```

Compatible vectors. Addition of real vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^m$ defines another vector $\boldsymbol{w} = \boldsymbol{u} + \boldsymbol{v} \in \mathbb{R}^m$. The components of $\boldsymbol{w}$ are the sums of the corresponding components of $\boldsymbol{u}$ and $\boldsymbol{v}$, $w_i = u_i + v_i$, for $i = 1, 2, \ldots, m$. Addition of vectors with different number of components is not defined, and attempting to add such vectors produces an error. Such vectors with different number of components are called *incompatible*, while vectors with the same number of components are said to be *compatible*. Scaling of $\boldsymbol{u}$ by a defines a vector $\boldsymbol{z} = a\boldsymbol{u}$, whose components are $z_i = a u_i$, for $i = 1, 2, \ldots, m$. Vector addition and scaling in Octave are defined using the + and $*$ operators.

```
octave] uT=[1 0 1 2]; vT=[2 1 3 -1]; wT=uT+vT; disp(wT)
   3   1   4   1
octave] rT=[1 2]; uT+rT
operator +: nonconformant arguments (op1 is 1x4, op2 is 1x2)
octave] a=3; zT=a*uT; disp(zT)
   3   6  -3   6
octave]
```

2.3. Working with vectors

Ranges. The vectors used in applications usually have a large number of components, $m \gg 1$, and it is important to become proficient in their manipulation. Previous examples defined vectors by explicit listing of their m components. This is impractical for large m, and support is provided for automated generation for often-encountered situations. First, observe that Table 1 mentions one distinguished vector, the zero element that is a member of any vector space $\mathbf{0} \in V$. The zero vector of a real vector space $\mathscr{R}_m$ is a column vector with m components, all of which are zero, and a mathematical convention for specifying this vector is $\mathbf{0}^T = [\ 0\ \ 0\ \ \ldots\ \ 0\ ] \in \mathbb{R}^m$. This notation specifies that transpose of the zero vector is the row vector with m zero components, also written through explicit indexing of each component as $\mathbf{0}_i = 0$, for $i = 1, \ldots, m$. Keep in mind that the zero vector $\mathbf{0}$ and the zero scalar 0 are different mathematical objects. The ellipsis symbol in the mathematical notation is transcribed in Octave by the notion of a range, with `1:m` denoting all the integers starting from 1 to m, organized as a row vector. The notation is extended to allow for strides different from one, and the mathematical ellipsis $i = m, m-1, \ldots, 1$ is denoted as `m:-1:1`. In general `r:s:t` denotes the set of numbers $\{r, r+s, \ldots, r+ns\}$ with $r+ns \leqslant t$, and r, s, t real numbers and n a natural number, $r, s, t \in \mathbb{R}$, $n \in \mathbb{N}$. If there is no natural number n such that $r+ns \leqslant t$, an empty vector with no components is returned.

```
octave] m=4; disp(1:m)
   1   2   3   4
octave] disp(m:-1:2)
   4   3   2
octave] r=0; s=0.2; t=1; disp(r:s:t)
    0.00000    0.20000    0.40000    0.60000    0.80000    1.00000
octave] r=0; s=0.3; t=1; disp(r:s:t)
    0.00000    0.30000    0.60000    0.90000
octave] r=0; s= -0.2; t=1; disp(r:s:t)
[](1x0)
octave]
```

An efficient, expressive feature of many software systems including Octave is to use ranges as indices to a vector, as shown below for the definition of $\mathbf{0} \in \mathbb{R}^4$. Note that the index range `i` is organized as a row, and a transpose operation must be applied to obtain `z` as a column vector.

```
octave] m=4; i=1:m; z(i)=i.^i; z=transpose(z); disp(z)
     1
     4
    27
   256
```

```
octave] i
i =

   1   2   3   4

octave] disp(transpose(z))
   0   0   0   0
octave]
```

Visualization. A component-by-component display of a vector becomes increasingly unwieldy as the number of components m becomes large. For example, the numbers below seem an inefficient way to describe the sine function.

```
octave]  t=0:0.1:1.5; disp(sin(t))
  Columns 1 through 8:

   0.00000   0.09983   0.19867   0.29552   0.38942   0.47943   0.56464   0.64422

  Columns 9 through 16:

   0.71736   0.78333   0.84147   0.89121   0.93204   0.96356   0.98545   0.99749
octave]
```

Indeed, such a piece-by-piece approach is not the way humans organize large amounts of information, preferring to conceptualize the data as some other entity: an image, a sound excerpt, a smell, a taste, a touch, a sense of balance, or relative position. All seven of these human senses will be shown to allow representation by linear algebra concepts, including representation by vectors.

As a first example consider *visualization*, the process of transforming data into a sight perception. A familiar example is constructing a plot of the graph of a function. Recall that in mathematics the *graph of a function* $f: X \to Y$ relating elements of the *domain* X to those in *codomain* Y is a set of ordered pairs $G_f = \{(x,y) | y = f(x), x \in X\}$. For a commonly encountered function such as sin: $[0, 2\pi) \to [-1, 1]$, the graph $G_{\sin} = \{(x, \sin(x)) | x \in [0, 2\pi)\}$ contains an uncountably infinite number of elements, and obviously cannot be explicitly listed. The sine function is continuous, meaning that no matter how small an open interval (c,d) within the function codomain $[-1,1]$ one considers, there exists an interval (a,b) in the function domain $[0, 2\pi]$ whose image by the sine function is contained in (c,d). In mathematical "$\delta - \varepsilon$" notation this is stated as: $\forall \varepsilon > 0, \exists \delta_\varepsilon, |x_1 - x_0| < \delta_\varepsilon \Rightarrow |\sin(x_1) - \sin(x_0)| < \varepsilon$. This mathematical notation is concise and precise, but perceptive mainly to the professional mathematician. A more intuitive visualization of continuity is obtained by approximating the graph of a function $f: X \to Y$ by a finite set of samples, $G_f^m = \{(x_i, y_i) | x_i \in X, y_i = f(x_i), i = 1, \ldots, m, m \in \mathbb{N}\}$. Strictly speaking, the sampled graph G_f^m would indicate jumps interpretable as discontinuities, but when plotting the points human sight perception conveys a sense of continuity for large sample sizes, $m \gg 1$. For the sine function example, consider sampling the domain $[0, 2\pi)$ with a step size $h = 2\pi / m$, $m \gg 1$. To obtain a visual representation of the sampled sine function the Octave `plot` function can be used to produce a figure that will appear in another window, interactively investigated, and subsequently closed. For large m one cannot visually distinguish the points in the graph sample, though this is apparent for smaller sample sizes. This is shown below by displaying a subrange of the sampled points with stride s. This example also shows the procedure to save a permanent copy of the displayed figure through the Octave `print -deps` command that places the currently displayed plot into an Encapsulated Postscript file. The generated figure file can be linked to a document as shown here in Figure 1, in which both plots render samples of the graph of the sine function, but the one with large m is perceived as being continuous.

```
octave] m=1000; h=2*pi/m; x=(0:m-1)*h;
octave] y=sin(x); plot(x,y);
octave] close;
octave] s=50; i=1:s:m; xs=x(i); ys=y(i);
octave] plot(x,y,'b',xs,ys,'bo');
octave] print -depsc L01Fig01.eps;
octave] close;
octave]
```

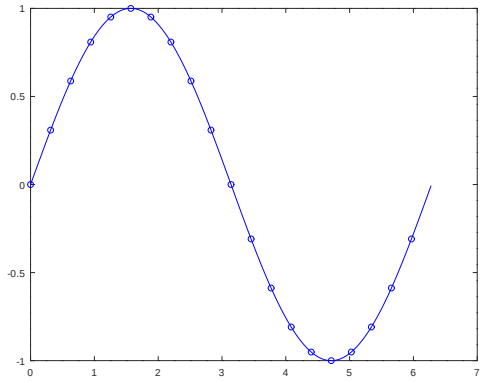

Figure 1. Visualization of vectors of sampled function graphs.

3. Matrices

3.1. Forming matrices

The real numbers themselves form the vector space $\mathscr{R}_1 = (\mathbb{R}, \mathbb{R}, +, \cdot)$, as does any field of scalars, $\mathscr{S}_1 = (S, S, +, \cdot)$. Juxtaposition of m real numbers has been seen to define the new vector space $\mathscr{R}_m$. This process of juxtaposition can be continued to form additional mathematical objects. A *matrix* is defined as a juxtaposition of compatible vectors. As an example, consider n vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n \in V$ within some vector space $\mathscr{V} = (V, S, +, \cdot)$. Form a matrix by placing the vectors into a row,

$$\boldsymbol{A} = [\ \boldsymbol{a}_1\ \ \boldsymbol{a}_2\ \ \ldots\ \ \boldsymbol{a}_n\ ]. \tag{3}$$

To aid in visual recognition of a matrix, upper-case bold Latin letters will be used to denote matrices. The columns of a matrix will be denoted by the corresponding lower-case bold letter with a subscripted index as in equation (3). Note that the number of columns in a matrix can be different from the number of components in each column, as would be the case for matrix $\boldsymbol{A}$ from equation (3) when choosing vectors from, say, the real space $\mathscr{R}_m$, $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n \in \mathbb{R}^m$.

Vectors were seen to be useful juxtapositions of scalars that could describe quantities a single scalar could not: a position in space, a force in physics, or a sampled function graph. The crucial utility of matrices is their central role in providing a description of new vectors other then their column vectors, and is suggested by experience with Euclidean spaces.

3.2. Identity matrix

Consider first $\mathscr{R}_1$, the vector space of real numbers. A position vector $\boldsymbol{r} \in \mathscr{R}_1$ on the real axis is specified by a single scalar component, $\boldsymbol{r} = [x]$, $x \in \mathbb{R}$. Read this to mean that the position $\boldsymbol{r}$ is obtained by traveling x units from the origin at position vector $\boldsymbol{0} = [0]$. Look closely at what is meant by "unit" in this context. Since x is a scalar, the mathematical expression $\boldsymbol{r} = \boldsymbol{0} + x$ has no meaning, as addition of a vector to a scalar has not been defined. Recall that scalars were introduced to capture the concept of scaling of a vector, so in the context of vector spaces they always appear as multiplying some vector. The correct mathematical description is $\boldsymbol{r} = \boldsymbol{0} + x\boldsymbol{e}$, where $\boldsymbol{e}$ is the unit vector $\boldsymbol{e} = [1]$. Taking the components leads to $r_1 = 0_1 + xe_1$, where $r_1, 0_1, e_1$ are the first (and in this case only) components of the $\boldsymbol{r}, \boldsymbol{0}, \boldsymbol{e}$ vectors. Since $r_1 = x$, $0_1 = 0$, $e_1 = 1$, one obtains the identity $x = 0 + x \cdot 1$.

Now consider $\mathscr{R}_2$, the vector space of positions in the plane. Repeating the above train of thought leads to the identification of two direction vectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$

$$\boldsymbol{r} = \begin{bmatrix} x \\ y \end{bmatrix} = x \begin{bmatrix} 1 \\ 0 \end{bmatrix} + y \begin{bmatrix} 0 \\ 1 \end{bmatrix} = x\boldsymbol{e}_1 + y\boldsymbol{e}_2, \boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \boldsymbol{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

```
octave] x=2; y=4; e1=[1; 0]; e2=[0; 1]; r=x*e1+y*e2
r =

   2
   4

octave]
```

Continuing the procees to $\mathscr{R}_m$ gives

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = x_1\boldsymbol{e}_1 + x_2\boldsymbol{e}_2 + \cdots + x_m\boldsymbol{e}_m, \boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{e}_2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \ldots, \boldsymbol{e}_m = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

For arbitrary m, the components are now $x_1, x_2, \ldots, x_m$ rather than the alphabetically ordered letters common for $m = 2$ or $m = 3$. It is then consistent with the adopted notation convention to use $\boldsymbol{x} \in \mathscr{R}_m$ to denote the position vector whose components are $(x_1, \ldots, x_m)$. The basic idea is the same as in the previous cases: to obtain a position vector scale direction $\boldsymbol{e}_1$ by x_1, $\boldsymbol{e}_2$ by $x_2, \ldots,$ $\boldsymbol{e}_m$ by x_m, and add the resulting vectors.

Juxtaposition of the vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_m$ leads to the formation of a matrix of special utility known as the *identity matrix*

$$\boldsymbol{I} = [\ \boldsymbol{e}_1\ \ \boldsymbol{e}_2\ \ \ldots\ \ \boldsymbol{e}_m\ ].$$

The identity matrix is an example of a matrix in which the number of column vectors n is equal to the number of components in each column vector $m = n$. Such matrices with equal number of columns and rows are said to be *square*. Due to entrenched practice an exception to the notation convention is made and the identity matrix is denoted by $\boldsymbol{I}$, but its columns are denoted the indexed bold-face of a different lower-case letter, $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m$. If it becomes necessary to explicitly state the number of columns in $\boldsymbol{I}$, the notation $\boldsymbol{I}_m$ is used to denote the identity matrix with m columns, each with m components.

4. Linear combinations

4.1. Linear combination as a matrix-vector product

The expression $\boldsymbol{x} = x_1\boldsymbol{e}_1 + x_2\boldsymbol{e}_2 + \cdots + x_m\boldsymbol{e}_m$ expresses the idea of scaling vectors within a set and subsequent addition to form a new vector $\boldsymbol{x}$. The matrix $\boldsymbol{I} = [\ \boldsymbol{e}_1\ \ \boldsymbol{e}_2\ \ \ldots\ \ \boldsymbol{e}_m\ ]$ groups these vectors together in a single entity, and the scaling factors are the components of the vector $\boldsymbol{x}$. To bring all these concepts together it is natural to consider the notation

$$\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x},$$

as a generalization of the scalar expression $x = 1 \cdot x$. It is clear what the operation $\boldsymbol{I}\boldsymbol{x}$ should signify: it should capture the vector scaling and subsequent vector addition $x_1\boldsymbol{e}_1 + x_2\boldsymbol{e}_2 + \cdots + x_m\boldsymbol{e}_m$. A specific meaning is now ascribed to $\boldsymbol{I}\boldsymbol{x}$ by identifying two definitions to one another.

Linear combination. Repeateadly stating "vector scaling and subsequent vector addition" is unwieldy, so a special term is introduced for some given set of vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$.

DEFINITION. (LINEAR COMBINATION) . *The linear combination of vectors* $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n \in V$ *with scalars* $x_1, x_2, \ldots, x_n \in S$ *in vector space* $(V, S, +, \cdot)$ *is the vector* $\boldsymbol{b} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + \ldots x_n\boldsymbol{a}_n$.

Matrix-vector product. Similar to the grouping of unit vectors $\boldsymbol{e}_1, ..., \boldsymbol{e}_m$ into the identity matrix $\boldsymbol{I}$, a more concise way of referring to arbitrary vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ from the same vector space is the matrix $\boldsymbol{A} = [\ \boldsymbol{a}_1\ \ \boldsymbol{a}_2\ \ \ldots\ \ \boldsymbol{a}_n\ ]$. Combining these observations leads to the definition of a matrix-vector product.

DEFINITION. (MATRIX-VECTOR PRODUCT). *In the vector space* $(V,S,+,\cdot)$, *the product of matrix* $\boldsymbol{A}=[\,\boldsymbol{a}_1\ \boldsymbol{a}_2\ \ldots\ \boldsymbol{a}_n\,]$ *composed of columns* $\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots,\boldsymbol{a}_n\in V$ *with the vector* $\boldsymbol{x}\in S_n$ *whose components are scalars* $x_1,x_2,\ldots,x_n\in S$ *is the linear combination* $\boldsymbol{b}=x_1\boldsymbol{a}_1+x_2\boldsymbol{a}_2+\ldots x_n\boldsymbol{a}_n=\boldsymbol{A}\boldsymbol{x}\in V$.

4.2. Linear algebra problem examples

Linear combinations in E_2. Consider a simple example that leads to a common linear algebra problem: decomposition of forces in the plane along two directions. Suppose a force is given in terms of components along the Cartesian x,y-axes, $\boldsymbol{b}=b_x\,\boldsymbol{e}_x+b_y\,\boldsymbol{e}_y$, as expressed by the matrix-vector multiplication $\boldsymbol{b}=\boldsymbol{I}\boldsymbol{b}$. Note that the same force could be obtained by linear combination of other vectors, for instance the normal and tangential components of the force applied on an inclined plane with angle θ, $\boldsymbol{b}=x_t\,\boldsymbol{e}_t+x_n\,\boldsymbol{e}_n$, as in Figure 2. This defines an alternate reference system for the problem. The unit vectors along these directions are

$$\boldsymbol{t}=\begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix},\boldsymbol{n}=\begin{bmatrix}-\sin\theta\\ \cos\theta\end{bmatrix},$$

and can be combined into a matrix $\boldsymbol{A}=[\,\boldsymbol{t}\ \ \boldsymbol{n}\,]$. The value of the components (x_t,x_n) are the scaling factors and can be combined into a vector $\boldsymbol{x}=[\,x_t\ \ x_n\,]^T$. The same force must result irrespective of whether its components are given along the Cartesian axes or the inclined plane directions leading to the equality

$$\boldsymbol{I}\boldsymbol{b}=\boldsymbol{b}=\boldsymbol{A}\boldsymbol{x}. \tag{4}$$

Interpret equation (4) to state that the vector $\boldsymbol{b}$ could be obtained either as a linear combination of $\boldsymbol{I}$, $\boldsymbol{b}=\boldsymbol{I}\boldsymbol{b}$, or as a linear combination of the columns of $\boldsymbol{A}$, $\boldsymbol{b}=\boldsymbol{A}\boldsymbol{x}$. Of course the simpler description seems to be $\boldsymbol{I}\boldsymbol{b}$ for which the components are already known. But this is only due to an arbitrary choice made by a human observer to define the force in terms of horizontal and vertical components. The problem itself suggests that the tangential and normal components are more relevant; for instance a friction force would be evaluated as a scaling of the normal force. The components in this more natural reference system are not known, but can be determined by solving the vector equality $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$, known as a *linear system of equations*. Procedures to carry this out will be studied in more detail later, but Octave provides an instruction for this common problem, the backslash operator, as in `x=A\b`.

```
octave] ex=[1; 0]; ey=[0; 1];
octave] b=[0.2; 0.4]; I=[ex ey]; I*b
ans =

   0.20000
   0.40000

octave] th=pi/6; c=cos(th); s=sin(th);
octave] tvec=[c; s]; nvec=[-s; c];
octave] A=[tvec nvec];
octave] x=A\b
x =

   0.37321
   0.24641

octave] [x(1)*tvec x(2)*nvec]
ans =

   0.32321  -0.12321
   0.18660   0.21340

octave]
```

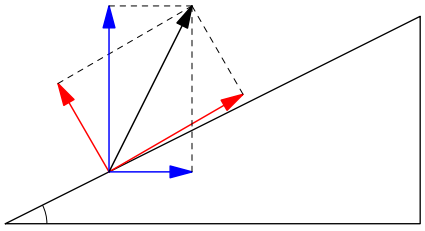

Figure 2. Alternative decompositions of force on inclined plane.

Linear combinations in $\mathscr{R}_m$ and $\mathscr{C}^0[0,2\pi)$. Linear combinations in a real space can suggest properties or approximations of more complex objects such as continuous functions. Let $\mathscr{C}^0[0,2\pi)=(C[0,2\pi),\mathbb{R},+,\cdot)$ denote the vector space of continuous functions that are periodic on the interval $[0,2\pi)$, $C[0,\pi)=\{f|f:\mathbb{R}\to\mathbb{R},f(t)=f(t+2\pi)\}$. Recall that vector addition is defined by $(f+g)(t)=f(t)+g(t)$, and scaling by $(af)(t)=af(t)$, for $f,g\in C[0,2\pi)$, $a\in\mathbb{R}$. Familiar functions within this vector space are $\sin(kt)$, $\cos(kt)$ with $k\in\mathbb{N}$, and these can be recognized to intrinsically represent periodicity on $[0,2\pi)$, a role analogous to the normal and tangential directions in the inclined plane example.

Define now another periodic function $b(t+2\pi)=b(t)$ by repeating the values $b(t)=t(\pi-t)(2\pi-t)$ from the interval $[0,2\pi)$ on all intervals $[2p\pi,2(p+1)\pi\,]$, for $p\in\mathbb{Z}$. The function b is not given in terms of the "naturally" periodic functions $\sin(kt)$, $\cos(kt)$, but could it thus be expressed? This can be stated as seeking a linear combination $b(t)=\sum_{k=1}^{\infty} x_k \sin(kt)$, as studied in Fourier analysis. The coefficients x_k could be determined from an analytical formula involving calculus operations $x_k=\frac{1}{\pi}\int_0^{2\pi} b(t)\sin(kt)\,\mathrm{d}t$, but we'll seek an approximation using a linear combination of n terms

$$b(t)\cong\sum_{k=1}^{n} x_k \sin(kt), A(t)=[\ \sin(t)\ \ \sin(2t)\ \ \ldots\ \ \sin(nt)\ ], A:\mathbb{R}\to\mathbb{R}^n.$$

Organize this as a matrix vector product $b(\boldsymbol{t})\cong A(\boldsymbol{t})\,\boldsymbol{x}$, with

$$A(\boldsymbol{t})=[\ \sin(\boldsymbol{t})\ \ \sin(2\boldsymbol{t})\ \ \ldots\ \ \sin(n\boldsymbol{t})\ ], \boldsymbol{x}=[\ x_1\ \ x_2\ \ \ldots\ \ x_n\ ]^T\in\mathbb{R}^n.$$

The idea is to sample the column vectors of $A(\boldsymbol{t})$ at the components of the vector $\boldsymbol{t}=[\ t_1\ \ t_2\ \ \ldots\ \ t_m\ ]^T\in\mathbb{R}^m$, $t_j=(j-1)h$, $j=1,2,\ldots,m$, $h=\pi/m$. Let $\boldsymbol{b}=b(\boldsymbol{t})$, and $\boldsymbol{A}=A(\boldsymbol{t})$, denote the so-sampled b,A functions leading to the definition of a vector $\boldsymbol{b}\in\mathbb{R}^m$ and a matrix $\boldsymbol{A}\in\mathbb{R}^{m\times n}$. There are n coefficients available to scale the column vectors of $\boldsymbol{A}$, and $\boldsymbol{b}$ has m components. For $m>n$ it is generally not possible to find $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x}$ would exactly equal $\boldsymbol{b}$, but as seen later the condition to be as close as possible to $\boldsymbol{b}$ leads to a well defined solution procedure. This is known as a least squares problem and is automatically applied in the Octave `x=A\b` instruction when the matrix `A` is not square. As seen in the following numerical experiment and Figure 3, the approximation is excellent even though the information conveyed by $m=1000$ samples of $b(t)$ is now much more efficiently stored in the form chosen for the columns of $\boldsymbol{A}$ and the $n=11$ scaling coefficients that are the components of $\boldsymbol{x}$.

```
octave] m=1000; h=2*pi/m; j=1:m;
octave] t(j)=(j-1)*h; t=transpose(t);
octave] n=5; A=[];
octave] for k=1:n
          A = [A sin(k*t)];
        end
octave] bt=t.*(pi-t).*(2*pi-t);
octave] x=A\bt;
octave] b=A*x;
octave] s=50; i=1:s:m;
        ts=t(i); bs=bt(i);
        plot(ts,bs,'ok',t,b,'r');
octave] print -depsc L01Fig02.eps
octave] close;
```

```
octave]
```

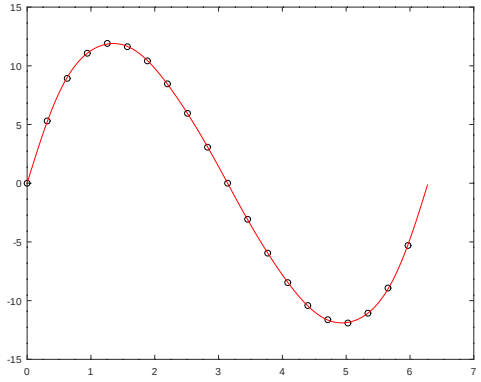

Figure 3. Comparison of least squares approximation (red line) with samples of exact function $b(t)$.

5. Vectors and matrice in data science

The above examples highlight some essential aspects of linear algebra in the context of data science applications.

- Vectors organize information that cannot be expressed as a single number and for which there exists a concept of scaling and addition.
- Matrices group together multiple vectors.
- The matrix-vector product expresses a linear combination of the column vectors of the matrix.
- Solving a linear system $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}=\boldsymbol{I}\boldsymbol{b}$, to find $\boldsymbol{x}\in\mathbb{R}^m$ for given $\boldsymbol{b}\in\mathbb{R}^m$, re-expresses the linear combination

$$\boldsymbol{b}=b_1\boldsymbol{e}_1+\cdots+b_m\boldsymbol{e}_m, \boldsymbol{I}=[\ \boldsymbol{e}_1\ \ \boldsymbol{e}_2\ \ \ldots\ \ \boldsymbol{e}_m\ ],$$

as another linear combination

$$\boldsymbol{b} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + \ldots x_n\boldsymbol{a}_n, \boldsymbol{A} = [\ \boldsymbol{a}_1\ \ \boldsymbol{a}_2\ \ \ldots\ \ \boldsymbol{a}_n\ ].$$

For certain problems the linear combination $\boldsymbol{Ax}$ might be more insightful, but the above transformation is information-preserving, with $\boldsymbol{b}, \boldsymbol{x}$ both having the same number of components.

- Finding the best approximation of some given $\boldsymbol{b} \in \mathbb{R}^m$ by a linear combination $\boldsymbol{Ax}$ of the n column vectors of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is known as a least squares problem and transforms the information from the m components of $\boldsymbol{b}$ into n components of $\boldsymbol{x}$, and knowledge of the form of the column vectors. If $m > n$ and the form of the columns of $\boldsymbol{A}$ can be succintly stated, the transformation compresses information.

Data science seeks to extract regularity directly from available data, not necessarily invoking any additional hypotheses. The typical scenario is that immense amounts of data are available, with limited capability of human analysis. In this context it is apparent that the least squares problem is of greater interest than solving a linear system with a square matrix. It should also be clear that while computation by hand of small examples is useful to solidify theroretical concepts, it is essential to become proficient in the use of software that can deal with large data sets, such as Octave.