1 Basic Concepts and Tools

Scientific Computing relies on tools from many branches of Mathematics, as well as from Computer Science and Engineering. Calculus and Linear Algebra, in particular, stand out by providing concepts that underlie almost all problems in Scientific Computing. We will begin by reviewing a few key linear algebra notions, such as vectors, matrices, inner product, and norm. These notions are simple but important - they will be used in throughout this course.

These lectures are about computation. We take an experimental approach and illustrate many concepts and ideas with numerical examples using either MATLAB or GNU Octave computer programs. If Linear Algebra is the language of Scientific Computing, then MATLAB and GNU Octave are expert translators. Both program environments allow us to compute at the linear algebra level, without resorting to lower level programming. MATLAB and GNU Octave also provide visualization tools and a powerful and easy to use scripting language for writing new functions and simple programs. MATLAB and GNU Octave are similar enough that almost all examples presented in this book will work on either system. We henceforth will refer to GNU Octave simply as Octave.

MATLAB is developed and sold by the company The MathWorks with headquarter in Natick, Massachusetts. Octave is a free and open-source programming environment developed by John W. Eaton at the University of Wisconsin in Madison, Wisconsin. An installation guide can be found in the appendix. Scilab is another similar programming environment which, like Octave, is available for free. The development of Scilab was begun in France.

MATLAB and Octave can be used in an interactive fashion. We can compute simple expressions directly:

```
>> 2 + 2
ans =  4
>> quit
```

The command `quit` is used to exit MATLAB and Octave.

Many of the examples in this course can be computed by working with MATLAB or Octave interactively at the command-line. However, more sophisticated examples require the use of scripts and functions that can be edited as text files. We will illustrate this below.

1.1 Linear Algebra Concepts and MATLAB and Octave

The aim of this lecture is to review some useful concepts in linear algebra and to introduce a reader to MATLAB and Octave. The examples below are done in MATLAB, but can be carried out in Octave in the same manner. We will point out some minor differences in these computing environments. MATLAB and Octave have a help command, which is valuable for refreshing details of the use of certain MATLAB commands. When we in the text below refer to MATLAB, generally Octave can be used in the same manner. We will comment on situations when this is not the case.

[0]Version September 16, 2013

1.1.1 Vectors, matrices, inner product, and norms

Let

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

be a vector with m real-valued entries b_j. We refer to $\mathbf{b}$ as an m-vector. The entries of $\mathbf{b}$ are called scalars. All vectors in this course are column vectors, unless explicitly specified otherwise. Throughout this text vectors are represented with lower case boldface letters. Scalars are represented with regular lower case letters and sometimes with Greek letters $(\alpha, \beta, \gamma, \dots)$.

We can create and manipulate vectors in MATLAB and Octave directly:

```
>> b = [1; 2; 3; 4]
b =

   1
   2
   3
   4

>>
```

The above example creates a 4-vector. Vector entries are enclosed by square brackets. The semicolon is used inside brackets to separate rows; commas or blanks separate columns. For example, a row-vector can be created with

```
>> r = [1  2  3  4]
r =

   1   2   3   4

>>
```

The *transpose operator*, written as $\mathbf{b}^T$, transforms a column vector into a row vector and vice-versa. The MATLAB command apostrophe `'` indicates transposition. In Octave the command is dot-apostrophe `.'`. This command also can be used in MATLAB.

Let $\mathbf{b}$ be the vector defined above. Then

```
>> btranspose=b'
btranspose =

   1   2   3   4

>>
```

Thus, `btranspose` is a row vector. The columns vector **b** defined above, can instead be defined as

```
>> b = [1 2 3 4]'
b =

   1
   2
   3
   4

>>
```

If we do not assign variable to store the result, the result is put in the variable ans:

```
>> b'
ans =

   1   2   3   4

>>
```

Two vectors are added by adding their corresponding entries:

```
>> d = [0 -1 2 -3]'
>> c = b + d
c =

    1
    1
    5
    1

>>
```

Individual entries of a vector may be accessed by using their subscripts. For example, the third entry of the vector **c** defined above can be accessed with

```
>> c(3)
ans =

    5

>>
```

The first three entries of the vector **c** defined above can be picked out with

```
>> c(1:3)
ans =
     1
     1
     5

>>
```

The vector with all entries zero is written **0**. In MATLAB the zero vector with three entries is generated with the command

```
>> z = zeros(3,1)
ans =
     0
     0
     0

>>
```

The first index indicates that the vector **z** has three columns and the last index that this vector is a matrix with one column. We will consider matrices below. Similarly, to determine a 3-vector with all entries one, we use the command

```
>> e = ones(3,1)
ans =
     1
     1
     1

>>
```

Multiplication of a scalar and a vector is defined by multiplying each entry of the vector by the scalar; for example:

```
>> 5 * c(1:3)
ans =

     5
     5
    25

>>
```

The set of all m-vectors with real-valued entries is denoted by $\mathbb{R}^m$. Let

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}$$

be vectors in $\mathbb{R}^m$. We define the *inner product* between $\mathbf{u}$ and $\mathbf{v}$ by:

$$(\mathbf{u}, \mathbf{v}) = \sum_{j=1}^{m} u_j v_j. \tag{1}$$

For any $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^m$ and $\alpha \in \mathbb{R}$, the inner product satisfies

$$\begin{aligned} (\mathbf{u}+\mathbf{v}, \mathbf{w}) &= (\mathbf{u}, \mathbf{w}) + (\mathbf{v}, \mathbf{w}), & (2) \\ (\alpha\mathbf{u}, \mathbf{v}) &= \alpha(\mathbf{u}, \mathbf{v}), & (3) \\ (\mathbf{u}, \mathbf{v}) &= (\mathbf{v}, \mathbf{u}), & (4) \\ (\mathbf{u}, \mathbf{u}) &\geq 0 \ \text{ with equality if and only if } \ \mathbf{u} = \mathbf{0}. & (5) \end{aligned}$$

Exercise 1.1

Show that the inner product (1) satisfies (2). □

Exercise 1.2

Show that the inner product (1) satisfies (3). □

Exercise 1.3

Show that the inner product (1) is *bilinear*, i.e., that it is linear in each argument $\mathbf{u}$ and $\mathbf{v}$ separately:

$$\begin{aligned} (\alpha\mathbf{u} + \beta\mathbf{v}, \mathbf{w}) &= \alpha(\mathbf{u}, \mathbf{w}) + \beta(\mathbf{v}, \mathbf{w}), \\ (\mathbf{u}, \alpha\mathbf{v} + \beta\mathbf{w}) &= \alpha(\mathbf{u}, \mathbf{v}) + \beta(\mathbf{u}, \mathbf{w}). \end{aligned}$$

Thus, show that the above relations follow from the properties (2)-(5). □

We can compute the inner product of two (column) m-vectors in MATLAB and Octave with the operation apostrophe-star (`'*`). The inner product of the vectors $\mathbf{b}$ and $\mathbf{c}$ defined above can be computed as:

```
>> innerprod=b'*c
innerprod = 22

>>
```

The right-hand side of the above formula signifies that MATLAB evaluates the inner product by evaluating the matrix-matrix product of the matrix **b**' with one row and four columns and the matrix $\mathbf{c}$ with four rows and one column. We will discuss matrix-matrix products further below.

MATLAB and Octave produce a lot of output, which quickly can clutter the computer screen. Appending a semicolon after a command suppresses the output. Several commands can be written on the same line. The above definitions of $\mathbf{b}$ and $\mathbf{c}$, and the computation of the inner product, give with the use of semicolons after the definitions the output:

```
>> b = [1 2 3 4]'; c = [1 1 5 1]';

>> innerprod=b'*c

innerprod = 22

>>
```

The *Euclidean length* of an m-vector $\mathbf{u}$ is defined as

$$||\mathbf{u}|| = \sqrt{(\mathbf{u}, \mathbf{u})} = \sqrt{\sum_{j=1}^{m} u_j^2}. \tag{6}$$

It can be computed in MATLAB and Octave with the `norm` function. When applied to the vector defined above, we obtain

```
>> norm(b)

ans =  5.4772

>>
```

Using transposition, the Euclidean length can be written as

$$\|\mathbf{u}\| = \sqrt{\mathbf{u}^T \mathbf{u}}. \tag{7}$$

The Euclidean length is an example of a *vector norm.* Vector norms are used to measure the size of a vector. They are real-valued and nonnegative functions of the vector entries. Any function $\| \cdot \| : \mathbb{R}^m \to \mathbb{R}$ such that, for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, and $\alpha \in \mathbb{R}$,

$$\begin{aligned} ||\mathbf{u}|| &> 0 \quad \text{for all} \quad \mathbf{u} \neq \mathbf{0} \quad \text{(positivity)}, & (8)\\ \|\mathbf{u}+\mathbf{v}\| &\leq \|\mathbf{u}\| + \|\mathbf{v}\| \quad \text{(triangle inequality)}, & (9)\\ \|\alpha\mathbf{u}\| &= |\alpha|\, \|\mathbf{u}\| \quad \text{(homogeneity)}, & (10) \end{aligned}$$

is referred to as a vector norm.

In order to show that the Euclidean length is a vector norm, we have to establish that it satisfies the properties (8)-(10). Positivity and homogeneity are straightforward consequences of the definition of the Euclidean length. The triangle inequality follows from the important Cauchy inequality,

$$|(\mathbf{u}, \mathbf{v})| \leq \|\mathbf{u}\|\, \|\mathbf{v}\|, \qquad \mathbf{u}, \mathbf{v} \in \mathbb{R}^m, \tag{11}$$

which provides a relation between the Euclidean length and the inner product. A hint for how it can be shown is provided in Exercise 1.8. We will often refer to the Euclidean length as the Euclidean vector norm, or just as the norm. However, we note that there are other vector norms that sometimes are convenient to use. This includes the uniform vector norm

$$\|\mathbf{u}\|_\infty = \max_{1 \leq j \leq m} |u_j| \tag{12}$$

and the 1-vector norm

$$\|\mathbf{u}\|_1 = \sum_{j=1}^{m} |u_j| \tag{13}$$

The latter norm is frequently used in statistics. In MATLAB and Octave these norms can be computed as

```
>> norm(b,inf)

ans =  4

>> norm(b,1)=

ans =  10

>>
```

The quotient between the inner product $(\mathbf{u},\mathbf{v})$ and product of the vector norms $\|\mathbf{u}\|\,\|\mathbf{v}\|$ has a geometric interpretation. It is the cosine of the angle θ between the vectors $\mathbf{u}$ and $\mathbf{v}$, i.e.,

$$\cos(\theta) = \frac{(\mathbf{u},\mathbf{v})}{\|\mathbf{u}\|\,\|\mathbf{v}\|}. \tag{14}$$

This formula can be used to determine the angle between two vectors.

When $|(\mathbf{u},\mathbf{v})| = \|\mathbf{u}\|\,\|\mathbf{v}\|$, then $|\cos(\theta)| = 1$. It follows that θ is an integer multiple of π. When θ is an even multiple of π, the vectors $\mathbf{u}$ and $\mathbf{v}$ point in the same direction. If, instead, θ is an odd multiple of π, the vectors $\mathbf{u}$ and $\mathbf{v}$ point in opposite direction. A situation that will receive more attention later in this course is when the vectors $\mathbf{u}$ and $\mathbf{v}$ are nonvanishing and $\theta = \frac{\pi}{2}$ or $\theta = -\frac{\pi}{2}$. Then $(\mathbf{u},\mathbf{v}) = 0$. Vectors with this property are said to be *orthogonal*.

Exercise 1.4

Show that the Euclidean norm satisfies (10). $\Box$

Exercise 1.5

Let

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Draw a picture of the set $\{\mathbf{u} : \|\mathbf{u}\| \leq 1\}$. $\Box$

Exercise 1.6

Let $\mathbf{u} = [u_1, u_2, \ldots, u_m]^T$ and define the function

$$||\mathbf{u}|| = \sqrt{\sum_{j=1}^{m-1} |u_j|^2}.$$

Differently from the definition of the Euclidean norm, the above sum involves the first $m-1$ components of $\mathbf{u}$, only. Is the function $\|\cdot\|$ so defined a norm? Either show that $\|\cdot\|$ satisfies the properties (8)-(10), or show that one of these properties is not satisfied for some vector $\mathbf{u}$. $\Box$

Exercise 1.7

Use the Cauchy inequality to show that (6) is a norm. Hint: Express $\|\mathbf{u}+\mathbf{v}\|^2 = (\mathbf{u}+\mathbf{v}, \mathbf{u}+\mathbf{v})$ in terms of $\|\mathbf{u}\|^2 = (\mathbf{u}, \mathbf{u})$ and $\|\mathbf{v}\|^2 = (\mathbf{v}, \mathbf{v})$ and use the bilinearity of the inner product to show the triangle inequality. $\Box$

Exercise 1.8

Show the Cauchy inequality. Hint: If one of the vectors $\mathbf{u}$ and $\mathbf{v}$ vanishes, then the Cauchy inequality holds. Therefore, assume that they are nonvanishing and consider the function

$$f(t) = \|\mathbf{u} + t\mathbf{v}\|^2, \qquad t \in \mathbb{R}.$$

This function is nonnegative for any value of t. By using that $\|\mathbf{u}+t\mathbf{v}\|^2 = (\mathbf{u}+t\mathbf{v}, \mathbf{u}+t\mathbf{v})$ and the properties of inner products, we can express the right-hand side in terms of $\|\mathbf{u}\|^2$, $t^2\|\mathbf{v}\|^2$, and $2t(\mathbf{u}, \mathbf{v})$. This representation shows that $f(t)$ is a quadratic polynomial. Determine the t-value for which f achieves its minimum, e.g., by determining the zero of derivative of $f(t)$. Denote this value by t_0. The Cauchy inequality follows from the fact that $f(t_0) \geq 0$. $\Box$

Exercise 1.9

Let $\mathbf{u} = [\cos(\alpha), \sin(\alpha)]^T$ and $\mathbf{v} = 3[\cos(\beta), \sin(\beta)]^T$. Thus, $\mathbf{u}$ and $\mathbf{v}$ are vectors in the plane at angles α and β with the positive horizontal axis, respectively. Verify formula (14) for these vectors. $\Box$

Exercise 1.10

Let

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

Draw pictures of the sets $\{\mathbf{u} : \|\mathbf{u}\|_\infty \leq 1\}$ and $\{\mathbf{u} : \|\mathbf{u}\|_1 \leq 1\}$. $\Box$

Exercise 1.11

Show that $\|\cdot\|_\infty$ is a vector norm, i.e., show that the function $\|\cdot\|_\infty$ satisfies the conditions (8)-(10). $\Box$

1.1.2 Matrices

Matrices are arrays of scalars. We denote matrices with upper case letters. For instance,

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix} \tag{15}$$

denotes an $m \times n$-matrix and we will write $A \in \mathbb{R}^{m \times n}$. When $n = 1$, the matrix simplifies to an m-vector. If instead $m = 1$, then the matrix becomes a row vector with n entries. If both $m = 1$ and $n = 1$, then the matrix is a scalar. We can enter and manipulate matrices in MATLAB and Octave as doubly-indexed vectors:

```
>> A=[1 2; 3 4]
A =

   1   2
   3   4

>> A(2,1)

ans = 3

>>
```

The first index indicates row number and the second index column number. It is often meaningful to think of a matrix as set of column vectors. Many of the properties of a matrix can be inferred by studying its columns. We write

$$A = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n],$$

where

$$\mathbf{a}_j = \begin{bmatrix} a_{1,j} \\ a_{2,j} \\ \vdots \\ a_{m,j} \end{bmatrix}.$$

Let $A \in \mathbb{R}^{m \times n}$ and

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n.$$

The *matrix-vector product* $A\mathbf{x}$ is a linear combination of the columns of A,

$$A\mathbf{x} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{j=1}^{n} x_j \mathbf{a}_j.$$

Matrix-vector products are easily computed in MATLAB and Octave:

```
>> A=[1 2; 3 4]
A =

   1   2
   3   4

>> x = [-1 1]'
x =

  -1
   1

>> A*x
ans =

   1
   1
```

In the above example, we multiply the first column of A by -1 and add that to the 2nd column of A. Matrix-vector products are linear combinations of the columns of the matrix - the matrix and vector must have conformable dimensions for the product to make sense.

The mapping $\mathbf{x} \to A\mathbf{x}$ is *linear*, which means that for any vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and any scalar α, we have

$$\begin{aligned} A(\mathbf{x}+\mathbf{y}) &= A\mathbf{x} + A\mathbf{y}, \\ A(\alpha\mathbf{x}) &= \alpha\,(A\mathbf{x}). \end{aligned}$$

Any linear mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$ can be expressed with an $m \times n$-matrix.

The *range* of a matrix $A \in \mathbb{R}^{m \times n}$, written range$(A)$, is the set of vectors that can be expressed as a linear combination of the columns of A. Thus,

$$\text{range}(A) = \{A\mathbf{x} : \mathbf{x} \in \mathbb{R}^n\}.$$

The *rank* of A is the dimension of the range. Hence, the rank is the number of linearly independent columns of A. A matrix is said to be of *full rank* if its rank is $\min\{m, n\}$. If a matrix is not of full rank, then its columns are linearly dependent. The MATLAB and Octave command `rank` estimates the rank of its matrix argument. We will see throughout this course that, generally, the columns of a matrix provide much more useful information about the matrix than the rows.

A square matrix with linearly dependent columns is said to be *singular*; its determinant vanishes. A square matrix with linearly independent columns is said to be *nonsingular*; it has a nonvanishing determinant. The MATLAB/Octave command for evaluating the determinant of a matrix is `det`. It is better to use the command `cond` to test for singularity. This command is discussed in Lecture 7.

Example 1.1

Let
$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}.$$
Then $\text{rank}(A) = 2$ and $\det(A) = -2$. $\square$

Example 1.2

Let
$$M = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}.$$
Then $\text{rank}(M) = 1$. Thus, the matrix is not of full rank. It is easy to see that its columns are linearly dependent. The determinant of the matrix is 0. $\square$

The *span* of a set of vectors $\{\mathbf{v}_j\}_{j=1}^k$, written as
$$\text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\},$$
is the set of all linear combinations the vectors $\mathbf{v}_j$.

Example 1.3

Let the matrix A have columns $\mathbf{a}_j$, i.e.,
$$A = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \in \mathbb{R}^{m\times n}.$$
Then
$$\text{range}(A) = \text{span}\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\},$$
$\square$

Example 1.4

Consider the diagonal matrix
$$A = \text{diag}[0.1, 0.1, \dots, 0.1] \in \mathbb{R}^{10,10}.$$
Then $\det(A) = 10^{-10}$. The determinant is tiny; however, the matrix is not close to a singular matrix. This illustrates that the size of the determinant is a poor indication of how close a matrix is to a singular matrix. We will in Lecture 7 return to how to measure this distance. $\square$

Magic squares are matrices, whose row and column sums, as well of the sums of the diagonal and anti-diagonal entries, are equal. One can generate magic squares in MATLAB using the function `magic`. For instance,

```
>> A=magic(3)
A =
     8   1   6
     3   5   7
     4   9   2

>>
```

The command `sum` determines the sum of the entries of a vector when applied to a vector, and a vector whose entries are the column sums when applied to a matrix. When `sum` is applied to the above matrix, we obtain

```
>> sum(A)
ans =

  15  15  15

>>
```

The command `diag`, when applied to a matrix, gives a vector made up of the diagonal entries of the matrix. We obtain

```
>> v=diag(A)
v =

   8
   5
   2

>> sum(v)
ans =

     15

>>
```

It remains to check the row and antidiagonal sums. We will return to this when we have introduced the transpose of a matrix. Here we would like to consider the rank of magic squares of increasing order. The following loop determines the rank of magic squares of orders one through 30:

```
>>for j=1:30
r(j)=rank(magic(j));
end

>> r

r =
  Columns 1 through 13

     1     2     3     3     5     5     7     3     9     7    11     3    13

  Columns 14 through 26

     9    15     3    17    11    19     3    21    13    23     3    25    15

  Columns 27 through 34

    27     3    29    17    31     3    33    19

>>
```

The above commands generate a row vector r, whose jth entry contains the rank of the magic square of order j. It is instructive to plot the vector with the command `bar(r)`, which yields the bar graph shown in Figure 1.

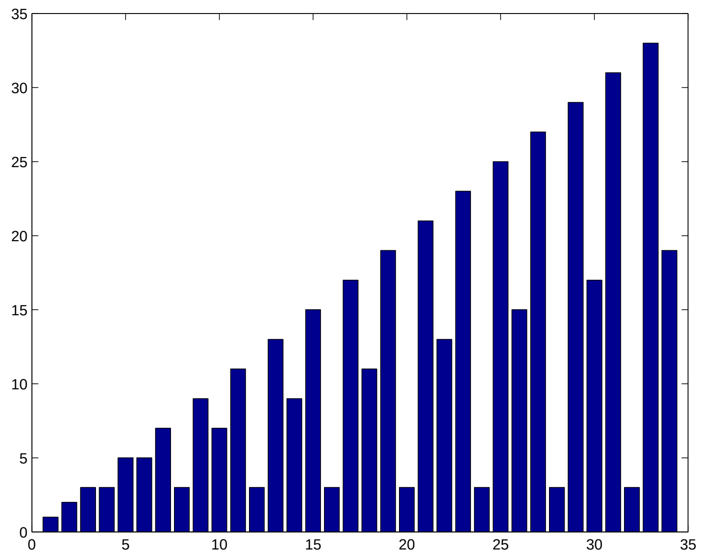

Figure 1: Rank of magic squares versus their order.

To gain some more insight into the rank distribution, we also plot r with the command `hist(r,35)`, which yields the histogram with 35 bins shown in Figure 2. The histogram shows that there are no magic squares of even rank larger than 2 and odd ranks larger or equal to 5 appear only once or twice. Clearly, magic squares have many curious properties.

A matrix that is not of full rank, such as many magic squares, have a null space of dimension larger than zero. The *null space* of a matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ is denoted by null(A). It is the set of solutions to $A\mathbf{x} = \mathbf{0}$, i.e.,

$$\text{null}(A) = \{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}.$$

Example 1.5

Let M be the matrix of Example 1.2. Then

$$\text{null}(M) = \text{span}\left\{ \begin{bmatrix} 2 \\ -1 \end{bmatrix} \right\}.$$

Thus, $\dim(\text{null}(M)) = 1$. $\Box$

Example 1.6

Let A be the matrix of Example 1.1. Then $\text{null}(A) = \{\mathbf{0}\}$. This space is said to have dimension zero. More generally, let the matrix A have linearly independent columns. Then $\text{null}(A) = \{\mathbf{0}\}$. $\Box$

We turn to matrix-matrix products. Introduce the matrices

$$\begin{aligned} A &= [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n] \in \mathbb{R}^{m\times n}, \qquad C = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_\ell] \in \mathbb{R}^{m\times \ell} \\ B &= [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_\ell] = \begin{bmatrix} b_{1,1} & b_{1,2} & \ldots & b_{1,\ell} \\ b_{2,1} & b_{2,2} & \ldots & b_{2,\ell} \\ \vdots & \vdots & & \vdots \\ b_{n,1} & b_{n,2} & \ldots & b_{n,\ell} \end{bmatrix} \in \mathbb{R}^{n\times \ell}. \end{aligned}$$

We express $C = AB$ as

$$\mathbf{c}_j = A\mathbf{b}_j = \sum_{k=1}^{n} b_{k,j}\mathbf{a}_k, \qquad j = 1, 2, \ldots, \ell.$$

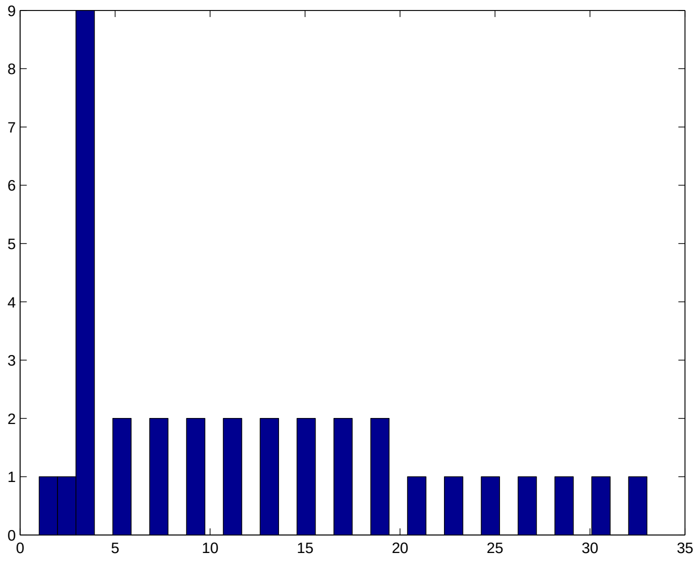

Figure 2: Histogram of ranks of magic squares of orders 1 through 34.

Each column of C is a matrix-vector product between A and the corresponding column of B. Therefore, each column of C is a linear combination of columns of A.

Matrix-vector products are easy to evaluate in MATLAB and Octave. Let A and M be the matrices of Examples 1.1 and 1.2, respectively. Then

```
>> C=A*M
C =

    5  10
   11  22

>> D=M*A
D =
    7 10
   14 20

>>
```

Thus, $AM \neq MA$.

Let $A \in \mathbb{R}^{n\times n}$ be of full rank. Then there is a unique matrix, denoted by A^{-1}, such that

$$A^{-1}A = A\,A^{-1} = I,$$

where

$$I = \mathrm{diag}[1, 1, \ldots, 1] \in \mathbb{R}^{n\times n}$$

denotes the *identity matrix*. The matrix A^{-1} is referred to as the *inverse of* A.

Consider the linear system of equations

$$A\mathbf{x} = \mathbf{b}. \tag{16}$$

Multiplying the left-hand side and the right-hand side by A^{-1} yields the solution

$$\mathbf{x} = A^{-1}\mathbf{b} \tag{17}$$

of the linear system. The vector $A^{-1}\mathbf{b}$ contains the coefficients of the expansion of $\mathbf{b}$ in terms of the columns $\mathbf{a}_j$ of A,

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

where x_j is the jth entry of the vector $\mathbf{x}$.

Define the *axis vectors* in $\mathbb{R}^n$,

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \cdots \quad \mathbf{e}_n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{18}$$

An expansion of $\mathbf{b}$ in terms of the axis vectors can be written as

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \sum_{j=1}^{n} b_j \mathbf{e}_j .$$

The solution of linear systems of equations provides a a change of basis. Instead of expressing $\mathbf{b}$ in terms of the axis vectors $\mathbf{e}_j$, the solution $\mathbf{x}$ consists of the coefficients in an expansion of $\mathbf{b}$ in terms of the columns $\mathbf{a}_j$ of A.

The solution $\mathbf{x}$, defined by (17), of the linear system of equations (16) can be computed efficiently in MATLAB and Octave with the *backslash operator*:

```
>> A=[1, 2; 3, 4]
A =

   1   2
   3   4

>> b = [1; 1]
b =

   1
   1

>> x = A\b
x =

  -1
   1

>>
```

The backslash operator applies a variant of Gaussian elimination to A to solve the linear system (16). Gaussian elimination will be discussed in Chapter 3. Here we just would like to point out that the use of the backslash operator requires fewer arithmetic operations than first computing the inverse A^{-1} and then evaluating the matrix-vector product implied in (17). We will discuss the backslash operator more in subsequent lectures.

We already have discussed transposition of a vector. The transpose of an $m \times n$- matrix A, denoted by A^T, is an $n \times m$-matrix, whose jth row contains the entries of the jth column of A for all rows. Let A be given by (15). Then

$$A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \dots & a_{m,1} \\ a_{1,2} & a_{2,2} & \dots & a_{m,2} \\ \vdots & \vdots & & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{m,n} \end{bmatrix} .$$

Example 1.7

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}, \qquad A^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix}.$$

□

Exercise 1.12

We return to the magic square of order 3. How can one determine the row sums in MATLAB or Octave using the commands sum and transposition? □

Exercise 1.13

The MATLAB or Octave command flipud reverses the order of the rows a matrix or vector. For instance, we obtain for the magic square of order three that

```
>> A=flipud(magic(3))
A =

   4   9   2
   3   5   7
   8   1   6

>>
```

How can the flipud command be used to determine the sum of the antidiagonal of A? □

A matrix is said to be *symmetric* if it is equal to its transpose, i.e., if $A = A^T$. In particular, symmetric matrices have to be square. Moreover, the entries $a_{i,j}$ of a symmetric matrix A satisfy $a_{i,j} = a_{j,i}$ for all indices i, j.

Let the matrices A and B be of commensurate sizes so that their product AB exists. Then it can be verified by straightforward (but tedious) computation that

$$(AB)^T = B^T A^T. \tag{19}$$

Exercise 1.14

Determine the rank of the matrix

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 4 & 3 \end{bmatrix}.$$

□

Exercise 1.15

Determine the transpose of the matrix in Exercise 1.14. □

Exercise 1.16

Give an example of a symmetric 2×2 matrix. □

Matrix norms are used to measure the "size" of a matrix. Let $\|\cdot\|$ denote one of the vector norms introduced above and let A be an $m \times n$ matrix. The matrix norm induced by this vector norm is defined as

$$\|A\| = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|. \tag{20}$$

A matrix norm measures how large the norm of the image $A\mathbf{x}$ can be when $\mathbf{x}$ is a vector of unit length. When the vector norm used for $A\mathbf{x}$ and $\mathbf{x}$ in (20) is the Euclidean vector norm, the function $\mathbf{x} \mapsto A\mathbf{x}$ maps the unit sphere in $\mathbb{R}^n$ to an ellipsoid in $\mathbb{R}^m$, i.e., the set of vectors $\{A\mathbf{x} : \|\mathbf{x}\| = 1\}$ is an ellipsoid in $\mathbb{R}^m$. Exercises 1.19 and 1.20 illustrate this property. When the vector norms used in (20) are the norms $\|\cdot\|_\infty$ or $\|\cdot\|_1$, defined by (12) and (13), respectively, the "ellipsoid" has edges.

Example 1.8

Let the vector norms in the right-hand side of (20) be any one of the vector norms introduced in earlier in this lecture and let I denote the identity matrix. Then, it follows from the definition (20) that

$$\|I\| = \max_{\|\mathbf{x}\|=1} \|I\mathbf{x}\| = 1.$$

□

Example 1.9

This example considers a very special matrix of order two. Let

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

A vector $\mathbf{x}$ in $\mathbb{R}^2$ of unit length can be written as

$$\mathbf{x}(t) = \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix}, \qquad t \in \mathbb{R}.$$

It follows that the vectors

$$A\mathbf{x}(t) = \begin{bmatrix} 2\cos(t) \\ \sin(t) \end{bmatrix}$$

describe an ellipse in $\mathbb{R}^2$ with semi-axes of length 2 and 1 as t traverses the interval $[0, 2\pi]$. □

Matrix norms satisfy properties similar to the properties (8)-(10) satisfied by vector norms. Specifically, a matrix norm is a function $\|\cdot\| : \mathbb{R}^{m\times n} \to \mathbb{R}$ that assigns a real-valued nonnegative number to every matrix in $\mathbb{R}^{m\times n}$, such that, for all $A, B \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$,

$$\begin{aligned} \|A\| &> 0 \quad \text{for all} \quad A \neq O \quad \text{(positivity)}, & (21)\\ \|A + B\| &\leq \|A\| + \|B\| \quad \text{(triangle inequality)}, & (22)\\ \|\alpha A\| &= |\alpha|\,\|A\| \quad \text{(homogeneity)}. & (23) \end{aligned}$$

Matrix norms that are induced by a vector norm (cf. (20) satisfy the above properties as well as

$$\|A\mathbf{x}\| \leq \|A\|\,\|\mathbf{x}\| \quad \text{(compatibility)}. \tag{24}$$

This property will be used in various contexts in this course and can be shown as follows. Let $\mathbf{x}$ be a vector of unit length. Then

$$\|A\mathbf{x}\| \leq \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| = \|A\| = \|A\|\,\|\mathbf{x}\|.$$

We may multiply this inequality by any nonnegative scalar α and obtain

$$\alpha\|A\mathbf{x}\| \leq \|A\|\,\alpha\|\mathbf{x}\|,$$

which in view of property (10) of a vector norm is equivalent to

$$\|A(\alpha\mathbf{x})\| \leq \|A\|\,\|\alpha\mathbf{x}\|. \tag{25}$$

Now $\mathbf{y} = \alpha\mathbf{x}$ is an arbitrary vector. Therefore (25) shows that (24) holds for any vector.

Exercise 1.17

Let $A = [a_{i,j}] \in \mathbb{R}^{m\times n}$. Consider the function

$$\|A\| = \max_{i,j} |a_{i,j}|, \tag{26}$$

where the maximum is taken over all indices i, j. Is $\|\cdot\|$ a matrix norm, i.e., does $\|\cdot\|$ satisfy the conditions (21)-(23)? Justify your answer. Find an example that shows that the function (26) does not satisfy (24). □

Exercise 1.18

Let $A = [a_{i,j}] \in \mathbb{R}^{m\times n}$. Consider the function

$$\|A\| = \sqrt{\sum_{i,j} a_{i,j}^2},$$

where summation is over all indices i, j. What is $\|I\|$? Compare with Example 1.8. Is $\|\cdot\|$ a matrix norm, i.e., does $\|\cdot\|$ satisfy the conditions (21)-(23)? Justify your answer. □

Exercise 1.19

Generate the matrix T with the MATLAB/Octave command

$$T = \textit{toeplitz}(\mathbf{v});$$

where $\mathbf{v} = [1, 1/2, 1/3, \ldots, 1/10]^T$. How does the matrix look? Matrices with this kind of structure are called *Toeplitz matrices*.

Let A be the leading 3×3 submatrix of the Toeplitz matrix T. Determine the image of A applied to the unit sphere in 3D. Specifically, generate a plot in 3D of many vectors $A\mathbf{v}$, where $\mathbf{v}$ lives on the unit sphere. This can be done with the MATLAB command plot3 as described below. Text following % is considered a comment by MATLAB.

```
V=randn(3,1000);  % generate 1000 vectors with normally distributed random entries
for k=1:1000
  V(:,k)=V(:,k)/norm(V(:,k));  % normalize columns to have unit length
end
W=T*V;
figure; hold;  % the command hold secures that previously generated graphs are kept
for k=1:1000
  plot3(W(1,k),W(2,k),W(3,k));  % plot image of vectors V(:,k) under T
end
```

You can click on the "round arrow" above the plot generated by the above MATLAB code and look at the plot from different angles. □

Exercise 1.20

The Hilbert matrix of order 10 can be generated with the MATLAB/Octave command

$$H = hilb(10);$$

Hilbert matrices are examples of Hankel matrices, which are characterized by having constant entries along every skew diagonal.

Let A be the leading 3×3 submatrix of the Hilbert matrix H. Determine the image of A applied to the unit sphere in 3D similarly as in Exercise 1.19. □

1.2 Script files for MATLAB and Octave

We illustrated above how MATLAB and Octave can be used in an interactive manner. However, sometimes it is inconvenient to have to enter data and commands several times when experiments are to be repeated. Script files provide a remedy.

1.2.1 The Collatz conjecture

Let x_1 be a positive integer and define the sequence $x_2, x_3, \ldots$ by

$$x_{k+1} := \begin{cases} 3x_k + 1 & \text{if } x_k \text{ is odd,} \\ x_k/2 & \text{if } x_k \text{ is even.} \end{cases}$$

Collatz[1] conjectured that for any initial integer x_1 larger than two, we obtain $x_n = 1$ for some finite n. The size of n depends on x_0. A Google search for "Collatz conjecture" yields further information. We would like to experimentally investigate this conjecture and whether there are other sequences with the same property.

Since we expect to have to determine elements x_{k+1} for large values of k, it is convenient to write a script in MATLAB. Scripts are stored in files with extension .m and are referred to as m-files. In the following m-file, the variable `x` is for the integer x_k. The `while`-loop is executed as long as the condition following `while` is true. The symbol `=` stands for "not equal to". The MATLAB function call `rem(k, j)` yields k modulo j. The MATLAB command

[1]L. Collatz, 1910-1990, was professor in Applied Mathematics at the University of Hamburg in Hamburg, Germany. He formulated the conjecture in 1937.

```
help rem
```

provides more information about the function `rem`. Text following the symbol % on the same line is considered a comment. The operator `==` is logical equal. The function `plot` provides graphical output.

Store the following script in a file named `collatz.m`.

```
xsave=[];  % initialize vector as the empty vector.
x=input('Enter integer larger than two: ');
while x~= 1
  if rem(x,2)==1  % True if x is odd. Then the next line will be executed.
    x=3*x+1;
  else
    x=x/2;
  end
  xsave=[xsave x];  % store sequence by prepending x to the available sequence.
end
plot(xsave,'*-')
number_of_steps=length(xsave)
```

The command `xsave=[]` initializes `xsave` to the empty vector. The command `xsave=[xsave x]` appends `x` to the end of the row vector. The above script can be executed by typing `collatz` on the MATLAB command line:

```
>> collatz
Enter integer larger than two: 4321

number_of_steps =

   170
```

The variables in the script are accessible after the script has been executed. For instance, we can look at the entries in the vector `xsave` by typing the name of this vector on the command line.

Exercise 1.21

Run the above code for several initial values x_1. Can one say anything about how the number of steps relates to the size of x_1? Change the plotted graph from a piece-wise straight line with starts at the break points to a piece-wise straight line with circles at the break points. Use the MATLAB command `help plot` to find out how. The MATLAB command `print` is helpful for storing a graph, for instance, in PostScript. Type `help plot` to find out how this can be done. □

In order to investigate other similar recursion formulas, we extend the above script to a function.

```
function [steps,xsave] = generalized_collatz(x,a,b)
% Input: x is the initial integer; a, b integer coefficients larger than 1.
% if x is not divisible by a, then we multiply x by b and add the
% remainder rem(x,a); otherwise we divide by a.
xsave=[];
if nargin==1
  a=2; b=3; % defaults gives Collatz sequence.
elseif nargin==2
  b=3;
end
while x>=a & length(xsave)<=1000
  remainder=rem(x,a);
  if remainder>0
    x=b*x+remainder;
  else
    x=x/a;
  end
  xsave=[xsave x];
end
semilogy(xsave,'r*-')
steps=length(xsave);
```

The MATLAB command `semilogy` plots the vector `xsave` using a logarithmic scale. The above code has input parameters `x`, `a`, and `b`. The output parameters are `steps` and `xsave`. Variables used inside a function are not accessible after execution of the function unless they are output parameters.

Exercise 1.22

Run the above code for several input parameter values. For which kind of pairs $\{a, b\}$ does the sequence seem to terminate? What is the purpose of the parameter `nargin`?. Key in the MATLAB command `help nargin` to find out. One also can use an analogous parameter `nargout`. What are its values and how can it be used? □

Exercise 1.23

Modify the above code so that it generates the sequence

$$x_{k+1} := \begin{cases} (3x_k - 1)/2 & \text{if } x_k \text{ is odd,} \\ x_k/2 & \text{if } x_k \text{ is even} \end{cases}$$

with x_0 a positive integer. Which initial values smaller than 1000 yield 1, 5, and 17 steps? □

The recent book "The Ultimate Challenge: The $3x+1$ Problem" by J. C. Lagarias, Amer. Math. Society, Providence, 2010, provides much insight about the Collatz conjecture and related problems.

1.3 Appendix: Installing Octave

Octave is available as a ready to run package for all major GNU Linux distributions. Windows and Mac OS X users can find installers for Octave at the web site:

http://sourceforge.net/project/showfiles.php?group_id=2888.

Follow the instructions found on the SoruceForge web site for installing Octave on your system.

Once installed, GNU Octave can be started by either clicking on its program icon or by opening a terminal window and typing "octave." Octave runs in a terminal window and presents the user with a welcome message and a command prompt:

```
GNU Octave, version 3.0.0
Copyright (C) 2007 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTIBILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type ‘warranty’.

Octave was configured for "x86_64-unknown-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type ‘news’.
```