Linear Algebra and Differential Equations using MATLAB

© Martin Golubitsky[1] and Michael Dellnitz[2]

August 5, 1998

[1]Department of Mathematics, University of Houston, Houston, TX 77204-3476, USA
[2]Department of Mathematics and Computer Science, University of Paderborn, D-33098 Paderborn, Germany

Contents

Preface

These notes provide an integrated approach to linear algebra and ordinary differential equations based on computers — in this case the software package MATLAB® [1]. We believe that computers can improve the conceptual understanding of mathematics — not just enable the completion of complicated calculations. We use computers in two ways: in linear algebra computers reduce the drudgery of calculations and enable students to focus on concepts and methods, while in differential equations computers display phase portraits graphically and enable students to focus on the qualitative information embodied in solutions rather than just on developing formulas for solutions.

We develop methods for solving both systems of linear equations and systems of (constant coefficient) linear ordinary differential equations. It is generally accepted that linear algebra methods aid in finding closed form solutions to systems of linear differential equations. The fact that the graphical solution of systems of differential equations can motivate concepts (both geometric and algebraic) in linear algebra is less often discussed. These notes begin by solving linear systems of equations (through standard Gaussian elimination theory) and discussing elementary matrix theory. We then introduce simple differential equations — both single equations and planar systems — to motivate the notions of eigenvectors and eigenvalues. In subsequent chapters linear algebra and ODE theory are often mixed.

Regarding differential equations, our purpose is to introduce at the sophomore – junior level ideas from dynamical systems theory. We focus on phase portraits (and time series) rather than on techniques for finding closed form solutions. We assume that now and in the future practicing scientists and mathematicians will use ODE solving computer programs more frequently than they will use techniques of integration. For this reason we have focused on the information that is embedded in the computer graphical approach. We discuss both typical phase portraits (Morse-Smale systems) and typical one parameter bifurcations (both local and global). Our goal is to provide the mathematical background that is needed when interpreting the results of computer simulation.

[1] MATLAB is a registered trademark of The MathWorks Inc. Natick, MA

The integration of computers: Our approach assumes that students have an easier time learning with computers if the computer segments are fully integrated with the course material. So we have interleaved the instructions on how to use MATLAB with the examples and theory in the text. With ease of use in mind, we have also provided a number of preloaded matrices and differential equations with the notes. Any equation label in this text that is followed by an asterisk can be loaded into MATLAB just by typing the formula number. For the successful use of this text, it is important that students have access to computers with MATLAB and the computer files associated with these notes.

John Polking has developed an excellent graphical user interface for solving planar systems of autonomous differential equations called pplane5. Until Chapter **??** we use pplane5 (and a companion code dfield5) instead of using the MATLAB native commands for solving ODEs. In these notes we also provide an introduction to pplane5 and the other associated software routines.

For the most part we treat the computer as a black box. We have not attempted to explain how the computer, or more precisely MATLAB, performs computations. Linear algebra structures are developed (typically) with proofs, while differential equations theorems are presented (typically) without proof and are instead motivated by computer experimentation.

There are two types of exercises included with most sections — those that should be completed using pencil and paper (called Hand Exercises) and those that should be completed with the assistance of computers (called Computer Exercises).

Ways to use the text: We envision this course as a one-year sequence replacing the standard one semester linear algebra and ODE courses. There is a natural one semester *Linear Systems* course that can be taught using the material in this book. In this course students will learn both the basics of linear algebra and the basics of linear systems of differential equations. This one semester course covers the material in the first eight chapters. The *Linear Systems* course stresses eigenvalues and a baby Jordan normal form theory for 2×2 matrices and culminates in a classification of phase portraits for planar constant coefficient linear systems of differential equations. Time permitting additional linear algebra topics from Chapters 9 and 10 may be included. Such material includes changes of coordinates for linear mappings, and

orthogonality including Gram-Schmidt orthonormalization and least squares fitting of data.

We believe that by being exposed to ODE theory a student taking just the first semester of this sequence will gain a better appreciation of linear algebra than will a student who takes a standard one semester introduction to linear algebra. However, a more traditional *Linear Algebra* course can be taught by omitting Chapter 7 and de-emphasizing some of the material in Chapter 6. Then there will be time in a one semester course to cover a selection of the linear algebra topics mentioned at the end of the previous paragraph.

A schematic diagram of the dependency relations between chapters is given in Figure **??**. Note that if two arrows in that figure point towards the same chapter, then material in both of the preceding chapters will be needed to complete the given chapter. Of particular importance is the observation that the material in Chapters **??**, **??**, and **??** may be omitted without repercussion.

file=figures/depdiag.eps

Figure 0.1: Dependency relations between chapters.

Comments on Individual Chapters

Chapters ??–?? We consider the first two chapters to be introductory material and we attempt to cover this material as quickly as we can. Chapter **??** introduces MATLAB along with elementary remarks on vectors and matrices. In our course we ask the students to read the material in Chapter **??** and to use the computer instructions in that chapter as an entry into MATLAB. In class we cover only the material on dot product. Chapter **??** explains how to solve systems of linear equations and is required for a first course on linear algebra. The proof of the uniqueness of reduced echelon form matrices is not very illuminating for students and can be omitted in classroom discussion. Sections whose material we feel can be omitted are noted by asterisks in the Table of Contents and Section **??** is the first example of such a section.

In Chapter **??** we introduce matrix multiplication as a notation that simplifies the presentation of systems of linear equations. We then show how matrix multiplication

leads to linear mappings and how linearity leads to the principle of superposition. Multiplication of matrices is introduced as composition of linear mappings, which makes transparent the observation that multiplication of matrices is associative. The chapter ends with a discussion of inverse matrices and the role that inverses play in solving systems of linear equations. The determinant of a 2×2 matrix is introduced and its role in determining matrix inverses is emphasized.

Chapter ?? This chapter provides a nonstandard introduction to differential equations. We begin by emphasizing that solutions to differential equations are functions (or pairs of functions for planar systems). We explain in detail the two ways that we may graph solutions to differential equations (time series and phase space) and how to go back and forth between these two graphical representations. The use of the computer is mandatory in this chapter. Chapter **??** dwells on the qualitative theory of solutions to autonomous ordinary differential equations. In one dimension we discuss the importance of knowing equilibria and their stability so that we can understand the fate of all solutions. In two dimensions we emphasize constant coefficient linear systems and the existence (numerical) of invariant directions (eigendirections). In this way we motivate the introduction of eigenvalues and eigenvectors, which are discussed in detail for 2×2 matrices. Once we know how to compute eigenvalues and eigendirections, we then show how this information coupled with superposition leads to closed form solution to initial value problems, at least when the eigenvalues are real and distinct.

We are not trying to give a thorough grounding in techniques for solving differential equations in Chapter **??**; rather we are trying to give an introduction to the ways that modern computer programs will represent graphically solutions to differential equations. We have included, however, a section on separation of variables for those who wish to introduce techniques for finding closed form solutions to single differential equations at this time. Our preference is to omit this section in the *Linear Systems* course as well as to omit the applications in Section **??** of the linear growth model in one dimension to interest rates and population dynamics.

Chapter ?? In this chapter we introduce vector space theory: vector spaces, subspaces, spanning sets, linear independence, bases, dimensions and the other basic

notions in linear algebra. Since solutions to differential equations naturally reside in function spaces, we are able to illustrate that vector spaces other than $\mathbf{R}^n$ arise naturally. We have found that, depending on time, the proof of the main theorem, which appears in Section **??**, may be omitted in a first course. The material in these chapters is mandatory in any first course on linear algebra.

Chapters ??, ??, ??, and ?? At this juncture the text divides into two tracks: one concerned with the qualitative theory of solutions to linear and nonlinear planar systems of differential equations and one mainly concerned with the development of higher dimensional linear algebra. We begin with a description of the differential equations chapters.

Chapter **??** describes closed form solutions to planar systems of constant coefficient linear differential equations in three different ways: a direct method based on eigenvalues and eigenvectors, a method based on matrix exponentials, and a related method based on similarity of matrices. Each of these methods has its virtues and vices. Note that the Jordan normal form theorem for 2×2 matrices is proved when discussing how to solve linear planar systems using similarity of matrices.

The qualitative description of phase portraits (saddles, sinks, sources, stability, centers, etc.) for planar linear systems is presented in Chapter **??**. This description depends crucially on the linear algebra of 2×2 matrices developed in Chapters **??** and **??**.

In Chapter **??** we discuss the fact that nonlinear equations behave like their linear counterparts in a neighborhood of a hyperbolic equilibrium and we verify this fact through computer experimentation. Periodic solutions to planar systems are introduced through the use of phase–amplitude equations. The chapter ends with a description of Morse-Smale nonlinear autonomous differential equations.

The discussion of planar systems of differential equations ends in Chapter **??** with a discussion of the typical bifurcations that one may expect with the variation of one parameter. These bifurcations include saddle-node bifurcations, which create a pair of equilibria, Hopf bifurcations, which create limit cycles, and global bifurcations. The discussion is framed around the breakdown of the linearization theorem at nonhyperbolic equilibria and the use of computer experimentation.

It is unusual to include the material in Chapters **??** and **??** in courses at this

level. We contend, however, that the material is accessible to students who feel comfortable with computer experimentation and provides an alternative to learning only techniques for finding closed form solutions in a first course on differential equations. Both approaches are important; but we believe that the geometric approach based on computer simulation will be more the norm in the future than will be the determination of closed form solutions. Accordingly, we have arranged the material with the geometric theory appearing first and the techniques for finding closed form solutions appearing later.

Chapters ??, ??, ??, and ?? Material from these four chapters on linear algebra can be covered directly after Chapter **??**. Chapter **??** discusses determinants, characteristic polynomials, and eigenvalues for $n \times n$ matrices. Chapter **??** presents more advanced material on linear mappings including row rank equals column rank and the matrix representation of mappings in different coordinate systems. The material in Sections **??** and **??** could be presented directly after Chapter **??**, while the material in Section **??** explains the geometric meaning of similarity and requires the discussion of similar matrices in Chapter **??**.

Orthogonal bases and orthogonal matrices, least squares and Gram-Schmidt orthonormalization, and symmetric matrices are presented in Chapter **??**. This material is very important, but is not required later in the text, and may be omitted.

The Jordan normal form theorem for $n \times n$ matrices is presented in Chapter **??**. Diagonalization of matrices with distinct real and complex eigenvalues is presented in the first two sections. The appendices, including the proof of the complete Jordan normal form theorem, are included for completeness and should be omitted in classroom presentations.

Chapter ?? In this chapter we present elements of the qualitative theory of autonomous systems of differential equations in higher dimensions. A comprehensive discussion of this topic is far beyond a course at this level. We emphasize the n-dimensional version of linearization near an equilibrium (including asymptotic stability which uses the Jordan normal form theorem of Chapter **??**). We also discuss some of the complicated dynamics that appear in the solutions of systems of three and four differential equations, including quasiperiodic motion (in linear and nonlin-

ear systems) and chaos (as represented by the Lorenz attractor). In order to explore these topics we need to introduce the MATLAB native routine `ode45` for solving systems of ordinary differential equations.

Chapters ??, ??, ??, and ?? The material in the remaining chapters is material that is more standard for a junior level course in ordinary differential equations. Chapter **??** discusses techniques for finding solutions of linear systems and higher order linear equations. This discussion requires generalized eigenvectors and hence aspects of Jordan normal form. We find solutions to higher order equations by reducing them to first order systems. Undetermined coefficients and resonance are also presented. The popular method of Laplace transforms is discussed in Chapter **??** and the importance of this method for solving forced linear equations (including discontinuous forcing) is emphasized. The material on Laplace transforms could be presented after linearity has been discussed (after Chapter **??** or even after Chapter **??**), but we do not recommend this approach.

Chapter **??** discusses linear equations with nonconstant coefficients (variation of parameters and reduction of order) and several techniques (some elementary) for solving certain nonlinear equations. These include substitution, exact differential equations, and planar Hamiltonian systems.

The text ends with a discussion of numerical techniques in Chapter **??**. Euler and Runge-Kutta methods are presented. This material could, in principle, be introduced as early as after Chapter **??**. However, this material has been included more as an appendix than as an integrated part of the course. These topics are appropriate as assignments for independent study by interested students.

The Classroom Use of Computers At the University of Houston we use a classroom with an IBM compatible PC and an overhead display. Lectures are presented three hours a week using a combination of blackboard and computer display. We find it inadvisable to use the computer for more than five minutes at a time; we tend to go back and forth between standard lecture style and computer presentations. (The preloaded matrices and differential equations are important to the smooth use of the computer in class.)

We ask students to enroll in a one hour computer lab where they can practice using

the material in the text on a computer, do their homework and additional projects, and ask questions of TA's. Our computer lab happens to have 15 power macs. In addition, we ensure that MATLAB and the laode files are available on student use computers around the campus (which is not always easy). The laode files are on the enclosed CDROM; they may also be downloaded by using a web browser or by anonymous ftp. See page ??? for instructions.

Acknowledgements: This course was first taught on a pilot basis during the 1995-96 academic year at the University of Houston. We thank the Mathematics Department and the College of Natural Sciences and Mathematics of the University of Houston for providing the resources needed to bring a course such as this to fruition. We gratefully acknowledge John Polking's help in adapting his software for our use and for allowing us access to his code so that we could write companion software for use in linear algebra.

We thank Denny Brown for his advice and his careful readings of the many drafts of this manuscript. We thank Gerhard Dangelmayr, Michael Field, Michael Friedberg, Steven Fuchs, Kimber Gross, Barbara Keyfitz, Charles Peters and David Wagner for their advice on the presentation of the material. We also thank Elizabeth Golubitsky, who has written the companion *Solutions Manual*, for her help in keeping the material accessible and in a proper order. Finally, we thank the students who stayed with this course on an experimental basis and by doing so helped to shape its form.

Houston and Bayreuth — Martin Golubitsky
May, 1998 — Michael Dellnitz

Chapter 1

Preliminaries

The subjects of linear algebra and differential equations involve manipulating vector equations. In this chapter we introduce our notation for vectors and matrices — and we introduce MATLAB, a computer program that is designed to perform vector manipulations in a natural way.

We begin, in Section **??**, by defining vectors and matrices, and by explaining how to add and scalar multiply vectors and matrices. In Section **??** we explain how to enter vectors and matrices into MATLAB, and how to perform the operations of addition and scalar multiplication in MATLAB. There are many special types of matrices; these types are introduced in Section **??**. In the concluding section, we introduce the geometric interpretations of vector addition and scalar multiplication; in addition we discuss the angle between vectors through the use of the dot product of two vectors.

1.1 Vectors and Matrices

In their elementary form, matrices and vectors are just lists of real numbers in different formats. An n-vector is a list of n numbers $(x_1, x_2, \ldots, x_n)$. We may write this vector

as a *row* vector as we have just done — or as a *column* vector

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

The set of all (real-valued) n-vectors is denoted by $\mathbf{R}^n$; so points in $\mathbf{R}^n$ are called vectors. The sets $\mathbf{R}^n$ when n is small are very familiar sets. The set $\mathbf{R}^1 = \mathbf{R}$ is the real number line, and the set $\mathbf{R}^2$ is the Cartesian plane. The set $\mathbf{R}^3$ consists of points or vectors in three dimensional space.

An $m \times n$ *matrix* is a rectangular array of numbers with m rows and n columns. A general 2×3 matrix has the form

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}.$$

We use the convention that matrix entries a_{ij} are indexed so that the first subscript i refers to the *row* while the second subscript j refers to the *column*. So the entry a_{21} refers to the matrix entry in the 2^{nd} row, 1^{st} column.

An $n \times m$ matrix A and an $n' \times m'$ matrix B are equal precisely when the sizes of the matrices are equal ($n = n'$ and $m = m'$) and when each of the corresponding entries are equal ($a_{ij} = b_{ij}$).

There is some redundancy in the use of the terms "vector" and "matrix". For example, a row n-vector may be thought of as a $1 \times n$ matrix, and a column n-vector may be thought of as a $n \times 1$ matrix. There are situations where matrix notation is preferable to vector notation and vice-versa.

Addition and Scalar Multiplication of Vectors

There are two basic operations on vectors: addition and scalar multiplication. Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be n-vectors. Then

$$x + y = (x_1 + y_1, \ldots, x_n + y_n);$$

that is, *vector addition* is defined as componentwise addition.

Similarly, *scalar multiplication* is defined as componentwise multiplication. A *scalar* is just a number. Initially, we use the term scalar to refer to a real number — but later on we sometimes use the term scalar to refer to a *complex* number. Suppose r is a real number; then the multiplication of a vector by the scalar r is defined as

$$rx = (rx_1, \ldots, rx_n).$$

Subtraction of vectors is defined simply as

$$x - y = (x_1 - y_1, \ldots, x_n - y_n).$$

Formally, subtraction of vectors may also be defined as

$$x - y = x + (-1)y.$$

Division of a vector x by a scalar r is defined to be

$$\frac{1}{r}x.$$

The standard difficulties concerning division by zero still hold.

Addition and Scalar Multiplication of Matrices

Similarly, we add two $m \times n$ matrices by adding corresponding entries, and we multiply a scalar times a matrix by multiplying each entry of the matrix by that scalar. For example,

$$\begin{pmatrix} 0 & 2 \\ 4 & 6 \end{pmatrix} + \begin{pmatrix} 1 & -3 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ 5 & 10 \end{pmatrix}$$

and

$$4 \begin{pmatrix} 2 & -4 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 8 & -16 \\ 12 & 4 \end{pmatrix}.$$

The main restriction on adding two matrices is that the matrices must be of the same size. So you cannot add a 4×3 matrix to 6×2 matrix — even though they both have twelve entries.

Hand Exercises

In Exercises **??** – **??**, let $x = (2, 1, 3)$ and $y = (1, 1, -1)$ and compute the given expression.

1 $x + y$.

2 $2x - 3y$.

3 $4x$.

4 *Let A be the 3×4 matrix*

$$A = \begin{pmatrix} 2 & -1 & 0 & 1 \\ 3 & 4 & -7 & 10 \\ 6 & -3 & 4 & 2 \end{pmatrix}.$$

(a) For which n is a row of A a vector in $\mathbf{R}^n$?

(b) What is the 2^{nd} column of A?

(c) Let a_{ij} be the entry of A in the i^{th} row and the j^{th} column. What is $a_{23} - a_{31}$?

For each of the pairs of vectors or matrices in Exercises **??** – **??**, decide whether addition of the members of the pair is possible; and, if addition is possible, perform the addition.

5 $x = (2, 1)$ *and* $y = (3, -1)$.

6 $x = (1, 2, 2)$ *and* $y = (-2, 1, 4)$.

7 $x = (1, 2, 3)$ *and* $y = (-2, 1)$.

8 $A = \begin{pmatrix} 1 & 3 \\ 0 & 4 \end{pmatrix}$ *and* $B = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$.

9 $A = \begin{pmatrix} 2 & 1 & 0 \\ 4 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ *and* $B = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$.

In Exercises **??** – **??**, let $A = \begin{pmatrix} 2 & 1 \\ -1 & 4 \end{pmatrix}$ and $B = \begin{pmatrix} 0 & 2 \\ 3 & -1 \end{pmatrix}$ and compute the given expression.

10 $4A + B$.

11 $2A - 3B$.

1.2 MATLAB

We shall use MATLAB to compute addition and scalar multiplication of vectors in two and three dimensions. This will serve the purpose of introducing some basic MATLAB commands.

Entering Vectors and Vector Operations

Begin a MATLAB session. We now discuss how to enter a vector into MATLAB. The syntax is straightforward; to enter the row vector $x = (1, 2, 1)$ type[1]

```
x = [1 2 1]
```

and MATLAB responds with

```
x =
     1     2     1
```

Next we show how easy it is to perform addition and scalar multiplication in MATLAB. Enter the row vector $y = (2, -1, 1)$ by typing

```
y = [2 -1 1]
```

and MATLAB responds with

```
y =
     2    -1     1
```

To add the vectors x and y, type

```
x + y
```

and MATLAB responds with

```
ans =
     3     1     2
```

[1]MATLAB has several useful line editing features. We point out two here:

(a) Horizontal arrow keys ($\rightarrow, \leftarrow$) move the cursor one space without deleting a character.

(b) Vertical arrow keys ($\uparrow, \downarrow$) recall previous and next command lines.

This vector is easily checked to be the sum of the vectors x and y. Similarly, to perform a scalar multiplication, type

```
2*x
```

which yields

```
ans =
     2     4     2
```

MATLAB subtracts the vector y from the vector x in the natural way. Type

```
x - y
```

to obtain

```
ans =
    -1     3     0
```

We mention two points concerning the operations that we have just performed in MATLAB.

(a) When entering a vector or a number, MATLAB automatically echoes what has been entered. *This echoing can be suppressed by appending a semicolon to the line.* For example, type

```
z = [-1 2 3];
```

and MATLAB responds with a new line awaiting a new command. To see the contents of the vector z just type `z` and MATLAB responds with

```
z =
    -1     2     3
```

(b) MATLAB stores in a new vector the information obtained by algebraic manipulation. Type

```
a = 2*x - 3*y + 4*z;
```

Now type `a` to find

```
a =
    -8    15    11
```

We see that MATLAB has created a new row vector a with the correct number of entries.

Note: In order to use the result of a calculation later in a MATLAB session, we need to name the result of that calculation. To recall the calculation `2*x - 3*y + 4*z`, we needed to name that calculation, which we did by typing `a = 2*x - 3*y + 4*z`. Then we were able to recall the result just by typing `a`.

We have seen that we enter a row n vector into MATLAB by surrounding a list of n numbers separated by spaces with square brackets. For example, to enter the 5-vector $w = (1, 3, 5, 7, 9)$ just type

```
w = [1 3 5 7 9]
```

Note that the addition of two vectors is only defined when the vectors have the same number of entries. Trying to add the 3-vector x with the 5-vector w by typing `x + w` in MATLAB yields the warning:

```
??? Error using ==> +
Matrix dimensions must agree.
```

In MATLAB new rows are indicated by typing `;`. For example, to enter the column vector

$$z = \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix},$$

just type:

```
z = [-1; 2; 3]
```

and MATLAB responds with

```
z =
    -1
     2
     3
```

Note that MATLAB will not add a row vector and a column vector. Try typing `x + z`.

Individual entries of a vector can also be addressed. For instance, to display the first component of `z` type `z(1)`.

Entering Matrices

Matrices are entered into MATLAB row by row with rows separated either by semicolons or by line returns. To enter the 2×3 matrix

$$A = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 4 & 7 \end{pmatrix},$$

just type

```
A = [2 3 1; 1 4 7]
```

MATLAB has very sophisticated methods for addressing the entries of a matrix. You can directly address individual entries, individual rows, and individual columns. To display the entry in the 1^{st} row, 3^{rd} column of A, type `A(1,3)`. To display the 2^{nd} column of A, type `A(:,2)`; and to display the 1^{st} row of A, type `A(1,:)`. For example, to add the two rows of A and store them in the vector x, just type

```
x = A(1,:) + A(2,:)
```

MATLAB has many operations involving matrices — these will be introduced later, as needed.

Computer Exercises

1 *Enter the* 3×4 *matrix*

$$A = \begin{pmatrix} 1 & 2 & 5 & 7 \\ -1 & 2 & 1 & -2 \\ 4 & 6 & 8 & 0 \end{pmatrix}.$$

As usual, let a_{ij} *denote the entry of* A *in the* i^{th} *row and* j^{th} *column. Use* MATLAB *to compute the following:*

(a) $a_{13} + a_{32}$.

(b) Three times the 3^{rd} *column of* A.

(c) Twice the 2^{nd} *row of* A *minus the* 3^{rd} *row.*

(d) The sum of all of the columns of A.

2 *Verify that* MATLAB *adds vectors only if they are of the same type, by typing*

(a) `x = [1 2]`, `y = [2; 3]` *and* `x + y`.

(b) `x = [1 2]`, `y = [2 3 1]` *and* `x + y`.

In Exercises **??** – **??**, let $x = (1.2, 1.4, -2.45)$ and $y = (-2.6, 1.1, 0.65)$ and use MATLAB to compute the given expression.

3 $3.27x - 7.4y$.

4 $1.65x + 2.46y$.

In Exercises **??** – **??**, let

$$A = \begin{pmatrix} 1.2 & 2.3 & -0.5 \\ 0.7 & -1.4 & 2.3 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} -2.9 & 1.23 & 1.6 \\ -2.2 & 1.67 & 0 \end{pmatrix}$$

and use MATLAB to compute the given expression.

5 $-4.2A + 3.1B$.

6 $2.67A - 1.1B$.

1.3 Special Kinds of Matrices

There are many matrices that have special forms and hence have special names — which we now list.

- A *square* matrix is a matrix with the same number of rows and columns; that is, a square matrix is an $n \times n$ matrix.
- A *diagonal* matrix is a square matrix whose only nonzero entries are along the main diagonal; that is, $a_{ij} = 0$ if $i \neq j$. The following is a 3×3 diagonal matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

  There is a shorthand in MATLAB for entering diagonal matrices. To enter this 3×3 matrix, type `diag([1 2 3])`.

- The *identity* matrix is the diagonal matrix all of whose diagonal entries equal 1. The $n \times n$ identity matrix is denoted by I_n. This identity matrix is entered in MATLAB by typing `eye(n)`.

- A *zero* matrix is a matrix all of whose entries are 0. A zero matrix is denoted by 0. This notation is ambiguous since there is a zero $m \times n$ matrix for every m and n. Nevertheless, this ambiguity rarely causes any difficulty. In MATLAB, to define an $m \times n$ matrix A whose entries all equal 0, just type `A = zeros(m,n)`. To define an $n \times n$ zero matrix B, type `B = zeros(n)`.

- The *transpose* of an $m \times n$ matrix A is the $n \times m$ matrix obtained from A by interchanging rows and columns. Thus the transpose of the 4×2 matrix

$$\begin{pmatrix} 2 & 1 \\ -1 & 2 \\ 3 & -4 \\ 5 & 7 \end{pmatrix}$$

  is the 2×4 matrix

$$\begin{pmatrix} 2 & -1 & 3 & 5 \\ 1 & 2 & -4 & 7 \end{pmatrix}.$$

  Suppose that you enter this 4×2 matrix into MATLAB by typing

```
A = [2 1; -1 2; 3 -4; 5 7]
```

  The transpose of a matrix A is denoted by A^t. To compute the transpose of A in MATLAB, just type `A'`.

- A *symmetric* matrix is a square matrix whose entries are symmetric about the main diagonal; that is $a_{ij} = a_{ji}$. Note that a symmetric matrix is a square matrix A for which $A^t = A$.

- An *upper triangular* matrix is a square matrix all of whose entries below the main diagonal are 0; that is, $a_{ij} = 0$ if $i > j$. A *strictly upper triangular* matrix is an upper triangular matrix whose diagonal entries are also equal to 0. Similar definitions hold for *lower triangular* and *strictly lower triangular* matrices. The following four 3×3 matrices are examples of upper triangular, strictly upper triangular, lower triangular,

and strictly lower triangular matrices:

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 2 & 4 \\ 0 & 0 & 6 \end{pmatrix} \quad \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 4 \\ 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 7 & 0 & 0 \\ 5 & 2 & 0 \\ -4 & 1 & -3 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 5 & 0 & 0 \\ 10 & 1 & 0 \end{pmatrix}.$$

- A square matrix A is *block diagonal* if

$$A = \begin{pmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_k \end{pmatrix}$$

where each B_j is itself a square matrix. An example of a 5×5 block diagonal matrix with one 2×2 block and one 3×3 block is:

$$\begin{pmatrix} 2 & 3 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 3 & 2 & 4 \\ 0 & 0 & 1 & 1 & 5 \end{pmatrix}.$$

Hand Exercises

In Exercises **??** – **??** decide whether or not the given matrix is symmetric.

1 $\begin{pmatrix} 2 & 1 \\ 1 & 5 \end{pmatrix}$.

2 $\begin{pmatrix} 1 & 1 \\ 0 & -5 \end{pmatrix}$.

3 (3).

4 $\begin{pmatrix} 3 & 4 \\ 4 & 3 \\ 0 & 1 \end{pmatrix}$.

5 $\begin{pmatrix} 3 & 4 & -1 \\ 4 & 3 & 1 \\ -1 & 1 & 10 \end{pmatrix}$.

In Exercises **??** – **??** decide which of the given matrices are upper triangular and which are strictly upper triangular.

6 $\begin{pmatrix} 2 & 0 \\ -1 & -2 \end{pmatrix}$.

7 $\begin{pmatrix} 0 & 4 \\ 0 & 0 \end{pmatrix}$.

8 (2).

9 $\begin{pmatrix} 3 & 2 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$.

10 $\begin{pmatrix} 0 & 2 & -4 \\ 0 & 7 & -2 \\ 0 & 0 & 0 \end{pmatrix}$.

A general 2×2 diagonal matrix has the form $\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$. Thus the two unknown real numbers a and b are needed to specify each 2×2 diagonal matrix. In Exercises **??** – **??**, how many unknown real numbers are needed to specify each of the given matrices:

11 *An upper triangular* 2×2 *matrix?*

12 *A symmetric* 2×2 *matrix?*

13 *An* $m \times n$ *matrix?*

14 *A diagonal* $n \times n$ *matrix?*

15 *An upper triangular* $n \times n$ *matrix?* **Hint:** *Recall the summation formula:*

$$1 + 2 + \cdots + k = \frac{k(k+1)}{2}.$$

16 *A symmetric $n \times n$ matrix?*

In each of Exercises **??** – **??** determine whether the statement is *True* or *False*?

17 *Every symmetric, upper triangular matrix is diagonal.*

18 *Every diagonal matrix is a multiple of the identity matrix.*

19 *Every block diagonal matrix is symmetric.*

Computer Exercises

20 *Use* MATLAB *to compute A^t when*

$$A = \begin{pmatrix} 1 & 2 & 4 & 7 \\ 2 & 1 & 5 & 6 \\ 4 & 6 & 2 & 1 \end{pmatrix} \tag{1.3.1}$$

Use MATLAB *to verify that $(A^t)^t = A$ by setting* `B=A'`, `C=B'`, *and checking that $C = A$.*

21 *Use* MATLAB *to compute A^t when $A = (3)$ is a 1×1 matrix.*

1.4 The Geometry of Vector Operations

In this section we discuss the geometry of addition, scalar multiplication, and dot product of vectors. We also use MATLAB graphics to visualize these operations.

Geometry of Addition

MATLAB has an excellent graphics language that we shall use at various times to illustrate concepts in both two and three dimensions. In order to make the connections between ideas and graphics more transparent, we will sometimes use previously developed MATLAB programs. We begin with such an example — the illustration of the parallelogram law for vector addition.

Suppose that x and y are two planar vectors. Think of these vectors as line segments from the origin to the points x and y in $\mathbf{R}^2$. We use a program written by T.A. Bryan to visualize $x + y$. In MATLAB type[2]:

[2]Note that all MATLAB commands are case sensitive — upper and lower case must be correct

```
x = [1 2];
y = [-2 3];
addvec(x,y)
```

The vector x is displayed in blue, the vector y in green, and the vector $x + y$ in red. Note that $x + y$ is just the diagonal of the parallelogram spanned by x and y. A black and white version of this figure is given in Figure **??**.

file=figures/vec2.eps,width=2.5in

Figure 1.1: Addition of two planar vectors.

The parallelogram law (the diagonal of the parallelogram spanned by x and y is $x + y$) is equally valid in three dimensions. Use MATLAB to verify this statement by typing:

```
x = [1 0 2];
y = [-1 4 1];
addvec3(x,y)
```

The parallelogram spanned by x and y in $\mathbf{R}^3$ is shown in cyan; the diagonal $x + y$ is shown in blue. See Figure **??**. To test your geometric intuition, make several choices of vectors x and y. Note that one vertex of the parallelogram is always the origin.

file=figures/vec3.eps,width=3.0in

Figure 1.2: Addition of two vectors in three dimensions.

Geometry of Scalar Multiplication

In all dimensions scalar multiplication just scales the length of the vector. To discuss this point we need to define the length of a vector. View an n-vector $x = (x_1, \ldots, x_n)$ as a line segment from the origin to the point x. Using the Pythagorean theorem, it can be shown that the *length* or *norm* of this line segment is:

$$||x|| = \sqrt{x_1^2 + \cdots + x_n^2}.$$

MATLAB has the command `norm` for finding the length of a vector. Test this by entering the 3-vector

```
x = [1 4 2];
```

Then type

```
norm(x)
```

MATLAB responds with:

```
ans =
    4.5826
```

which is indeed approximately $\sqrt{1+4^2+2^2} = \sqrt{21}$.

Now suppose $r \in \mathbf{R}$ and $x \in \mathbf{R}^n$. A calculation shows that

$$||rx|| = |r|||x||. \tag{1.4.1}$$

See Exercise **??**. Note also that if r is positive, then the direction of rx is the same as that of x; while if r is negative, then the direction of rx is opposite to the direction of x. The lengths of the vectors $3x$ and $-3x$ are each three times the length of x — but these vectors point in opposite directions. Scalar multiplication by the scalar 0 produces the 0 vector, the vector whose entries are all zero.

Dot Product and Angles

The *dot product* of two n-vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ is an important operation on vectors. It is defined by:

$$x \cdot y = x_1 y_1 + \cdots + x_n y_n. \tag{1.4.2}$$

Note that $x \cdot x$ is just $||x||^2$, the length of x squared.

MATLAB also has a command for computing dot products of n-vectors. Type

```
x = [1 4 2];
y = [2 3 -1];
dot(x,y)
```

MATLAB responds with the dot product of x and y, namely,

```
ans =
    12
```