Linear algebra for MATH2601: Theory

László Erdős

August 12, 2000

Contents

1 Introduction

The purpose of this note is to review and extend basic concepts of linear algebra for students having a background on the level of MATH1502. Substantial part of this material has been covered in the MATH1502 syllabus, but the current presentation is coherent and deeper. Chapter 5 (vectorspaces, change of basis, orthogonality) and part of Chapter 7 (spectral theorem and singular value decomposition) is probably new.

The review will be concise with only a few examples, since we assume that the students have seen these concepts in one way or another. Further explanations can be found in Demko's book or in Hubbard and Hubbard. We do not aim at rigorous and exhaustive proof, these can be found in textbooks. We will refer to Demko's book by [D] and to Hubbard and Hubbard book by [HH]. However, the sketch of some proofs are included whenever they help to develop the concept. In addition, there are two Appendices. The first one was originally a class note titled "Handout on Gaussian Elimination". This is a self-contained exposition of Demko 1.2 and 1.3 (since it was quite messy), i.e. it discusses the elimination in details. The main text assumes that the students are familiar with this process. The second Appendix contains the definition of the determinant for general square matrices and the Cramer's rule.

Basic notational convention: Numbers are denoted by lower case letters like $a, b, \ldots, \alpha, \beta, \ldots$. Vectors are denoted by boldface lower case letters like $\mathbf{a}, \mathbf{b}, \ldots \mathbf{x}$, and they always refer to column vectors. Matrices are denoted by capital letters like $A, B, \ldots$.

We sometimes leave certain trivial steps to the reader (mainly checking trivial calculations or giving examples). This are denoted by (*). Never ignore them!!!!

1.1 List of crucial problems

There are a few problems and exercises worked out. Further practice problems can be found in [D], [HH]. I focused on a few crucial basic model problems which everybody should be able to solve. These are seperately numbered. Here is their list:

- Problem 4.1. Full solution of a general system $A\mathbf{x} = \mathbf{b}$.
- Problem 4.4 Column space membership problem.
- Problem 4.5 Constraints for $\mathbf{b} \in R(A)$.
- Problem 4.6. Basis in the column space.
- Problem 4.7. Basis in the nullspace.
- Problem 4.8. Linear (in)dependence of given vectors.
- Problem 4.9. Expressing a vector as a linear combination of given vectors (if possible).
- Problem 4.15. Compute the inverse matrix.
- Problem 5.6. and 5.7. Change of basis matrix between two bases in a vectorspace of polynomials
- Problem 5.12. Write a given vector in a given basis.
- Problem 5.14. Change the coordinates of a vector from one given basis to another one.
- Problem 5.22. Write a vector in a given orthonormal basis.
- Problem 5.25. Write up the matrix of a general spatial rotation.

- Find the Gram-Schmidt orthogonalization of a given set of vectors. Find the QR decomposition of a given matrix. (See: Problems on the webpages shown in Section 6)
- Problem 7.6. Diagonalize a square matrix (if possible).
- Problem 7.7. Diagonalize a square matrix even with complex eigenvalues.
- Problem 7.9. Compute a huge power of a square matrix
- Problem 7.12 and 7.13. Find the spectral decomposition of a given square matrix.
- Problem 7.15. Find the singular value decomposition of a given matrix.

1.2 Importance of linear algebra

Most functions appearing in applications are nonlinear. In most cases, however, they are too complicated to attack them directly. The basic concept is to *approximate* them by simpler functions which are computationally more feasible. In principle many classes of functions can be candidates (polynomials, trigonometric functions etc.), but in practice the linear functions are used most frequently. Due to their simplicity, they can be handled quite easily, explicitly and cheaply. It turns out that this advantage, in most cases, is more important than the disadvantage of less precise approximation. One can usually make up for the lack of precision by using iterations (especially when computers do the actual computations). The idea of replacing the original function by its "best linear approximation" is very widely used and is called *linearization*.

We list a few basic examples which lead to problems in linear algebra.

Example I. The first basic example is the Newton's method to find a solution to $f(x) = 0$, where $f : \mathbf{R} \to \mathbf{R}$ is a "nice" function. It consists of approximating f by a linear function $L_{x_0}(x)$ (its tangent line with a contact point x_0 near the root), and find the solution $x = x_1$

of $L_{x_0}(x) = 0$. The result is not the exact root of $f(x) = 0$, but it is usually closer to the root than x_0, hence one can easily iterate the procedure by repeating the argument with the tangent line $L_{x_1}(x)$ at the point x_1, etc. It turns out that the root of most "nice" function can be found with extremely good precision just by solving a few linear equation.

If we want to implement the same procedure for a general function $f : \mathbf{R}^n \to \mathbf{R}^m$, then we have to extend the notion of linear functions to several (n) variables and $\mathbf{R}^m$–valued (vector-valued) functions. Such a linear function $L : \mathbf{R}^n \to \mathbf{R}^m$, is represented by a matrix A. Hence $L(\mathbf{x}) = \mathbf{0}$ (or more generally $L(\mathbf{x}) = \mathbf{b}$) naturally leads to the standard linear equation $A\mathbf{x} = \mathbf{b}$.

Example II. Another crucial application of linear algebra is in the theory of differential equations. You have learned how to solve the simplest differential equation

$$x'(t) = mx(t) \tag{1.1}$$

(prime denotes derivative with respect to t), for example this equation occurs in the simplest population model ($x(t)$ is the size of the population at time t and m is the birth or death rate, depending on whether $m > 0$ or $m < 0$). The solution is just $x(t) = e^{mt}x(0)$ exponential function. Now suppose that there are two species (rabbits and wolves), and their growth rate depend on the other species as well (wolves eat rabbits). Let $x_r(t)$ and $x_w(t)$ denote the size of these two populations at time t and assume that their growth rate depend linearly on the actual sizes of the two population (this is already an essential simplification and is a new example of "linearizing" the problem). Hence we obtain a *system of differential equations*:

$$\begin{aligned} x_r'(t) &= ax_r(t) + bx_w(t) \\ x_w'(t) &= cx_r(t) + dx_w(t) \end{aligned} \tag{1.2}$$

where a, b, c, d are some constants. In short, we have

$$\mathbf{x}'(t) = M\mathbf{x}(t) \tag{1.3}$$

where

$$\mathbf{x}(t) := \begin{pmatrix} x_r(t) \\ x_w(t) \end{pmatrix} \qquad \text{and} \qquad M := \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

where $\mathbf{x}(t)$ is the vector consisting of the size of both populations. Equations (1.1) and (1.3) look very similar, it is tempting to write up the solution to (1.3) as

$$\mathbf{x}(t) = e^{Mt}\mathbf{x}(0) \tag{1.4}$$

just copying the one-variable solution. However Mt is a matrix and we have no idea how to define its exponential. If M were diagonal

$$M = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}$$

i.e b, c were zero, then you can try to define

$$e^{Mt} = \begin{pmatrix} e^{at} & 0 \\ 0 & e^{dt} \end{pmatrix}$$

and this is indeed the correct definition. However, this is the uninteresting case, since $b, c = 0$ means that the rabbit and wolf populations do not influence each other and in fact you could have solved the $x_r'(t) = ax_r(t)$ and $x_w'(t) = dx_w(t)$ equations independently.

In the general case, you might try to define e^{Mt} as the matrix

$$\begin{pmatrix} e^{at} & e^{bt} \\ e^{ct} & e^{dt} \end{pmatrix}$$

but this is WRONG (CHECK (*) that $x_r(t), x_w(t)$ given by (1.4) with this definition of e^{Mt} does not satisfy the original system of equations (1.2)). You could already guess from your experience with the matrix multiplication that "interesting" matrix operations cannot be performed "entrywise".

However, it is worth going back to the diagonal case. Even if M is not diagonal, one can try to *diagonalize* it by changing the basis. If we can write $M = VDV^{-1}$, where D is diagonal,

then it turns out that the straightforward definition

$$e^{Mt} = Ve^{Dt}V^{-1}$$

is indeed the correct one, and this gives the solution to (1.2) via (1.4). Note that D is diagonal, hence the "entrywise" exponentiation defines e^{Dt} correctly.

In summary, if we can diagonalize the matrix M, we can solve (1.2). Recall that diagonalization is the same problem as computing eigenvalues and eigenvectors.

Example III. Linear algebra plays an enormous role in numerical methods. You have not learned how to differentiate a function of several variables, but if you accept that such concept exists (we will learn it later in this semester under the name of *partial differentiation*), then you can imagine a differential equation involving various partial derivatives of an unknown function. This is called a *partial differential equation*, or PDE in short, and most physical phenomena and engineering questions are described by such equations (for example the waves in a fluid, the heat conduction in a material, combustion process in chemistry etc.) Probably the most important (and complex) usage of computers in engineering sciences is to solve PDE's numerically. One of the basic method to do it is to transform the PDE into a huge linear system of equations (via a method called discretization). The number of unknowns is the number of grid points in the discretization and the result is supposed to improve as the number of grid points increase (refined grid). Hence we have to solve linear equations of the form $A\mathbf{x} = \mathbf{b}$, where the size of the vector $\mathbf{x}$ (number of grid points) is of order several thousand or even more, and so is the size of the matrix.

Example IV. Finally, linear algebra plays a role in computer graphics. A less deep application is the representation of planar and spatial transformations and motions via matrices. We will discuss this topic as this is important in graphical design programs. A much deeper application is in image compression. The best methods involve advanced tools of a relatively

young field of mathematics, called *wavelet-analysis* which uses serious linear algebra algorithms. We will not be able to enter this field, but we will present a simple example to show how linear algebra comes into this game.

1.3 Basic problems of linear algebra.

From the previous section it is clear that there are two fundamental problems of linear algebra:

(I.) Solve linear systems of the form

$$A\mathbf{x} = \mathbf{b}$$

where A is a given matrix and $\mathbf{b}$ is a given vector. If there is no exact solution, find the "best possible solution".

(II.) Calculate the eigenvalues and eigenvector of a given square matrix A.

In MATH1502 you learned how to solve these problems. Recall that Gauss elimination (row reduction) solves (I.) if there is a solution, and the Least Squares method gives a "best approximating solution" if there is no solution. In the special case of regular $n \times n$ system even the Cramer's formula (Section 9) is available (involving determinants), but this is rarely used in applications. To solve (II.) you need to find the roots of the characteristic polynomial, then solve linear equations to find the eigenvectors.

For theoretical purposes these solutions are perfect. However, when you implement them on a computer, you run into various problems. Keep in mind that the size of A can be huge in many important applications!

There are two basic issues: running time (cost) and precision. Despite the enormous increase of our computing power and speed, the time to run an algorithm is still an important issue (especially if you need real-time computations). It turns out that the issue of precision

is even more important and subtle. We will see examples demonstrating that a tiny rounding error (which is necessary in any computer data storage) can be disastrously magnified by the Gauss algorithm. This is called the instability of the method. Roughly speaking a method is stable if a small error occuring in the data or during the computation does not modify the final result too much. The simplest Gauss method is not stable.

After a review of the basic theoretical material, we will turn to algorithms which solve problem (I.) and (II.) in a stable fashion. The following algorithms will be considered, as time permits:

For problem (I.)

- Gauss elimination with partial pivoting
- QR factorization with Gram-Schmidt
- QR factorization with Householder method
- QR factorization with Givens rotations
- Jacobi iteration for regular matrix. Fixed point theorem.

For problem (II.)

- Power method for simple eigenvalues and their eigenvectors.
- Jacobi method of rotations for symmetric matrices.
- QR algorithm for general square matrices.

This list is far from complete. On one hand we will not consider the most general cases, rather we would like to give a glimpse of certain ideas. Hence in several cases we will assume that the matrix is square or regular or has no multiple eigenvalues etc. in order to present the main idea.

On the other hand, we will not consider algorithms designed for matrices of special form. In many cases the matrix A has a special structure just from the nature of the original problem. This special structure can be exploited to construct much more effective algorithms than the ones listed above, which are designed for "general" matrices. Most notable examples are certain types of "sparse" matrices (i.e. most entries are zero), namely "band" and "block" matrices. Matrices coming from numerical solution of PDE's are typically of this type.

There is certainly no reason to go into such details here. But you should keep in mind that such methods and algorithms are very well developed due to their enormous importance in applications. Taking into account the special structure of a matrix (if any) can increase your computing power by several orders of magnitude. Hence if you ever have to solve a "big" linear algebra problem numerically, look around in the literature before you start running the simplest method you know. In the model examples we consider in this class it does not really matter how precise and effective algorithm you use, since the size of the problem is small; the computational time and instability effects (rounding etc.) are usually irrelevant. But never forget, that in real life people do have to deal with matrices of size several thousands or even bigger. In these cases a smarter method can be a life or death question.