From Linear Equations to Eigenvalues/Eigenvectors and the QR algorithm

C. Godsalve

22 July, 2013 (revised version)
email seagods@hotmail.com

Contents

1 Introduction

Our first encounter with linear algebra is usually at school level when we are confronted with simultaneous equations. Linear algebra is a large and sophisticated area of both pure and applied mathematics. So, what is this article intended to be? By and large is it is meant to be a very informal introduction to just a fraction of a few areas in linear algebra. We shall state results mainly without proof, and not venture into what might be called "proper numerical analysis". The article might serve either as a primer to whet the appetite for the subject, or to assist a novice tackling advanced texts for the first time, or just as a brief revision article. We start off at the school level, and finish with eigenvalues, eigenvectors, and singular value decomposition. Later sections go beyond the level of most undergraduate courses.

So what are eigenvalues and eigenvectors? We leave the answer to that question to the end of §2. It shall suffice for the moment that they are important and fundamental to solving systems of differential equations, high order polynomials, and many other things

of practical importance. This article just takes the point of view that they are "sleekit beasties" [1]. That is, they are quite fascinating to study regardless of any practical import. We shall come across a whole zoo of sleekit beasties on our travels.

We start off by making the assumption that the reader already knows a bit about matrices and vectors. We don't assume much more than that, but even so we start off with a bit of "beginners" linear algebra revision. We shan't pay too much attention to definitions or rigour, that shall be left to real mathematicians! If the reader is not at all familiar with matrices or vectors, the author has written very brief, informal (and incomplete) introductions here [2].

General introductions to linear algebra are given by Antia [1], Schatzmann[2], and by Trevethen and Bau [3]. Of course there are many others, including more specialist texts by Golub and Van Loan [4] for instance.

In the more specialist area of eigenproblems we mention the classic text by Wilkinson [5], whose introduction of what is now called the "Wilkinson shift" made the numerical solution of eigenproblems a practical propostion. Golub and Van Loan [4] also have excellent sections on the QR algorithm.

The reader may regard this informal introduction as a motivation or preparation for moving on to these more rigourous texts.

We hope the reader doesn't mind bits of this article being a little sloppy. It is not the aim of the author to write a textbook. Of course the reader can just skip any section if s/he is well acquainted with any of these ideas already. We start off with with the basics, but shall arrive at a level beyond that taught in most undergraduate courses.

1.1 Layout

Since this article is fairly long, we hope a description of the layout will be useful. In §2, we introduce the indicial notation as applied to linear (or simultaneous) algebraic equations. We introduce, at a very casual level, Euclidean vectors, matrix multiplication, and determinants. In §3, we discuss determinants in detail. In doing so we introduce the Levi-Civita symbols and find the inverse of an $N \times N$ matrix and so the solution of arbitrarily large systems of linear equations. These are not practical methods, but are needed to understand the structure of such systems.

Before going on to look at practical methods for solving linear equations we take a look at the potential for severe loss of accuracy in §4. After this we look at practical methods for solving linear systems of equations and finding matrix inverses in §5 and §LUDecomp.

[1] From "Ode to a Mouse" by Rabbie Burns (1759-1796)

[2] http://seagods.stormpages.com/MathSchool.html

Since, when we come to eigenvalues and eigenvectors we shall need to look at similarity transforms, we therefore include an introduction to tensors in §7. The following section on tensors in §8 will mainly be of interest to physicists. However, continuing with the theme of tensors, we introduce covariant and contravariant vectors, and general tensors.

In §9, we return to linear algebra, and take our first look at eigenvector/eigenvalue problems. We look at 2×2, 3×3, and 4×4 matrices, and we note that these methods cannot be used for 5 by 5 matrices or larger because of the Abel-Ruffini impossibility theorem. (Polynomials of order 5 or higher have no algebraic solutions.) Because of this, we need iterative methods for high order eigenproblems.

Before we start to look at iterative methods, we take a look at Householder transforms in §12. Though interesting in themselves, they are usually used to make similarity transforms of the original matrix into upper Hessenberg form. This is the usual first step for eigenproblems. In §13, we look at some iterative methods finding single eigenvalues. This sets us up for finding eigenvalues simultaneously, and in doing so we come across invariant subspaces, and Gram-Schmidt orthogonalisation expressed in QR form. We arrive at the QR algorithm itself at the end of §14. In all this, we assume a matrix with N distinct real eigenvalues.

In §15 we take a look at the Rayleigh quotient algorithm for a single eigenvector. At this point we start to leave a lot of details to the textbooks. We introduce shift algorithms that speed up the convergence, and the Wilkinson shift which overcomes the problem of indistinct eigenvalues. These algorithms use the eigenvalues of the bottom right two by two sub-matrix of the current upper Hessenberg matrix. These can become complex, and this is how complex conjugate eigenvalue pairs are found. We simply note that when this is the case, the Francis implicit double shift method is used to avoid complex arithmetic. Finally, in §16, we include a brief section on non-square matrix multiplication and the singular value decomposition.

1.2 Warning

Before we continue, we give some advice. Don't write your own software to do this kind of stuff seriously. If you write your own code and then blow lots of company or university cash on serious supercomputer time, or tie up your departments computers for ages, you will eventually find the following to be true. For many decades extremely talented numerical analysts have developed standard code (which is free). They have devoted much of their working lives to making this stuff highly efficient and as accurate as possible. You will be wasting your time, other peoples time, and possibly a lot of cash if you ignore the work of these people. Write your own code as an exercise by all means. It is possible that, for some reason, you really must write your own code. But *netlib.org* should be your first port of call when looking for excellent software. There is more there than you could possibly write

in twenty lifetimes.

Robustness is another criterion for good computer code. It is entirely possible that a piece of computer code might be fast and accurate on test data, and suddenly fail altogether (or worse still yield plausible looking garbage) given different data.

Even if you are become well versed in the more technical aspects of these problems, and can write good robust solvers, there is also the aspect of parallel computation to consider. Many free source codes will (or may) have already dealt with this for you.

2 Some Basic Ideas and Notation

We shall use *indicial notation*. Instead of having a system of two linear equations such as

$$ax + by = c,$$

$$dx + ey = f. \tag{1}$$

The coefficients on the left hand side appear as two rows, one with an "a" and a "b", and the other with a "d" and an "e". The way the coefficients are "stacked" one above the other means we can also mentally organise the coefficients into *columns*. One column has an a above a d, the other has a b above an e. As we shall see, a system of row and column numbering is more than a little useful.

We shall write the same thing as

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 = b_2. \tag{2}$$

Comparing eqns.1 and 2, we see we have just replaced x and y with x_1 and x_2, c and f have been replaced with b_1 and b_2, and so on.

We shall call these four values of a in eqn.2 the elements of a *matrix*. Also, we shall call the two values of x and b in eqn.2 the elements of a *vector*. We shall not go into the formal definition of what is really meant by a vector until the end of this chapter, but we shall say that the term "vector" means more than just a bunch of numbers. Even so, we shall speak as though a vector was just a way of writing down a bunch of numbers. As we progress, we shall introduce various operations on rows and columns, and see that that not only the unknowns $(x_1, x_2, \ldots)$ and constants $(b_1, b_2, \ldots)$, but that the rows and columns of a matrix *along with the operations defined on them* are vectors as well. We note beforehand that the definition of a vector space we shall introduce later makes *no mention of lengths, dot products, angles, or any geometrical notions whatsoever.* These concepts are defined in vector spaces where a *metric* is also defined, or metric spaces.

Eqn.2 is written using *subscript* notation. We could easily have called x_1 and x_2, x^1 and x^2 where we would call x^2 "ex two" just as we would call x_2 "ex two". That is we could use *superscripts* as well as subscripts. Superscripts look confusing because, normally, we would think x^2 meant "x squared". We shall mostly use subscripts, but it should be clear enough from the context whether x^2 means "ex two" or "ex squared". We see later that we actually need both subscripts and superscripts in certain circumstances.

Throughout this article, unless stated otherwise, the matrix and vector elements are real, and never complex. The value of re-writing eqn.1 in the form of eqn.2 should be obvious to the extent that if we have a system of equations with many unknowns, rather than just two equations with just x and y, we shall soon run out of alphabet! So here in eqn.2, the numbers a, b, d, and e have been replaced with a_{ij} where i is understood to mean a row number and j is understood to be a column number. That is, if we look at eqn.2, we see two "rows" of linear equations. We can also see that coefficients can be thought of as being arranged in columns. So e in eqn.1 is in the second row and second column of the left hand side. So now e in eqn.1 appears as a_{22} in eqn.2. The right hand side (c and f) can be thought of as a column vector.

We write the whole caboodle as

$$\sum_{j=1}^{j=2} a_{ij}x_j = b_i, \tag{3}$$

where i=1,2. Here the $\sum$ stands for "sum over" or "addemallup".

Just in case the reader hasn't seen this before, we write this particular case out in full.

$$b_i = \sum_{j=1}^{j=2} a_{ij}x_j = a_{i1}x_1 + a_{i2}x_2, \; i = 1 \text{ or } 2. \tag{4}$$

The coefficient a_{ij} is in the ith row of the system of equations and the jth column. In this case we can see that i stands for whether the row number is 1 or is 2. The index j is the column number. It is that something that is being "summed over" The result on either side is a column vector (one column) with row number i. This shall become more obvious immediately below.

We state that eqn.3 is the rule for matrix-vector multiplication. The coefficients a_{ij} can be written as a square *array* or a *matrix*, and eqn.3 can be written in the form

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{5}$$

We restate that, although b_1 and b_2 are just a pair of numbers, we shall call the right hand side a vector. In eqn.5 we see a matrix A with elements a_{ij} multiplying a column vector x with elements x_j. This operation yields another vector b with elements b_i. The rule for multiplying a vector by a matrix is just that given in eqn.3.

Usually, we first come across vectors as things that have magnitude and direction. A vector is just a member of a *vector space* and does not have these properties. Having said that, the axioms defining a vector space can be added to and so a *Euclidean* vector space can be defined. Euclidean vectors have lengths (or norms) and given two members of a Euclidean vector space, an angle can be defined between them. Mostly, we will make no such distinction. Obviously, there is no need to define lengths or directions in solving eqn.1, but we shall speak as if all vectors are Euclidean vectors. In this chapter, we shall speak of vectors only on an informal footing. Later, we she shall introduce the axiomatic definition of a *vector space*.

Given two vectors b and c, we shall call the sum $b_1c_1 + b_2c_2 + \ldots$ the *dot product* of b and c. (In fact this is *only* true for a rectangular Cartesian reference frame, but we shall use the term anyway. We shall address the reasons for this later.) In this way we can see that the matrix A can be thought of as a set of row vectors. When looked upon as such we see that each element of b is a dot product of the column vector x and one of the row vectors of A.

We must emphasise that unless specifically stated we only speak of *square* matrices, by which we mean that the number of rows and the number of columns are the same. At any rate, this is the usual matrix/vector way of writing down a system of equations, and can be written as $A\mathbf{x} = \mathbf{b}$ or just $Ax = b$. That is upper case letters are meant to be matrices and lower case letters are meant to be vectors. When we mentioned running out of alphabet, we meant that just as well as dealing with the two by two matrices we have written down so far, we could be dealing with one trillion by one trillion matrices. So, in eqn.3 we would just as well as have a maximum value for i and j of 1000000000. In fact, very large matrices are commonplace in engineering and physics problems, and many other practical applications.

At this point we can introduce some very basic algebraic ideas. We can see immediately that we can have a matrix I which is has elements equal to 1 for all the diagonal elements (row number=column number, with rows counted top down and columns counted left to right) and zero for any other elements. Given such a matrix we immediately see that $Ix = x$. In other words I is the multiplicative *identity* matrix. Now, we had $Ax = b$. That is the matrix A is an operator, and the result of A operating on the vector x is the vector b. Supposing we then have another matrix B operating on b so that $Bb = c$. Is there a matrix C that operates on x such that $Cx = c$? It is just really a matter of looking at low order cases. (We call a 2 by 2 matrix a matrix of order 2 and so on.) It is easily seen that

$$C_{ij} = \sum_k A_{ik}B_{kj}. \tag{6}$$

Eqn.6 is the rule for matrix multiplication: the matrix C is equal to the multiplicative product of A and B. The reader may just write this out for the two by two matrix case. If we look upon A as being a set of row vectors, and B as a set of column vectors, then each

element c_{ij} of C is a dot product of the ith row of A and the jth column of B.

Then s/he shall be able to see that if $C = AB$ then $AB \neq BA$, or $AB - BA \neq 0$. Matrix multiplications, in general, *do not commute*. In AB, B is said to be *pre-multiplied* by A. In BA, B is said to be *post-multiplied* by A. If we look upon A as being a set of column vectors, and B as a set of row vectors, then each element C'_{ij} of BA is a dot product of the ith column of A and the jth row of B. Matrix multiplications are *associative* however, so that $A[BC] = [AB]C = ABC$.

In geometry, a matrix A operating on a *position vector* x can be made to represent a stretching/shrinking, rotation, and reflection of the vector x. It is thus a *tensor* (from the Latin for stretch). From another point of view the vector $b = Ax$ can represent the *same vector* as x, but having different elements to $x_1, x_2, \ldots$ because it is represented *in a different coordinate system*. For instance, the vector (x_1, x_2) might be $(1, 0)$. It points straight along the x axis. We might have a different coordinate system rotated by 45° clockwise to the first coordinate system. In this coordinate system exactly the same position is represented by $(1/\sqrt{2}, 1/\sqrt{2})$. This is the subject of the theory of invariants.

At this point we shall introduce the Einstein summation convention. Note that in eqn.6, the indices i and j appear once on either side. The index k appears twice on the right hand side, but does not appear on the left hand side at all. It is just the index being summed over. The *free* index i is the row number on both sides, and the free index j is the column number of the object on both sides. The index k is a *dummy index*, it just represents the summation and if we called it l or m the meaning is entirely unchanged. We can now rewrite eqn.6 as $C_{ij} = A_{ik}B_{kj}$. Because we see the index k twice *there is an implicit summation*. The reason for this is that Einstein got really fed up with writing $\sum$ over and over. (The convention first appears in Einstein's first publication of the general theory of relativity in 1916.) Whenever you see an index twice there is the ghost of a $\sum$ sitting there. If (after some manipulation of a formula using this convention) the free indices on one side do not match the free indices on the other, you know something is wrong. If the same index appears more than twice you know something is wrong.

Now, for numbers, unless $x = 0$, given any x we have a number $1/x$ such that $1/x \times x = 1$. That is $1/x$ is the multiplicative inverse of the operation of multiplying by x. We know that the inverse does not exist if $x = 0$. We now have matrix multiplication. If, for instance we have a 2×2 matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \tag{7}$$

what is the (multiplicative) inverse of A? Is there a case where the inverse does not exist? We denote the putative inverse of A as A^{-1}, after a little slog the reader can find that

$$A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{pmatrix}^T. \tag{8}$$

(Just write the eqn.3 or eqn.5 out to get a 4 by 4 system of linear equations and solve for x_1 and $x2$. T hen compare with $x = A^{-1}b$.) We have introduced the notation A^T called the *transpose* of A. It just means "swap the rows with the columns". That is $a^T_{ij} = a_{ji}$. Now eqn.8 goes off with a bang if $D = a_{11}a_{22} - a_{12}a_{21} = 0$. In this case the matrix inverse does not exist. The quantity D is called the *determinant* of the 2×2 matrix. Clearly it determines whether the inverse exists or not.

With a fair bit of effort (a 9×9 system of equations) the reader can work out the inverse of a 3×3 matrix. If we put D for the determinant, then

$$D = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31}). \tag{9}$$

Then

$$A^{-1} = \frac{1}{D}\begin{pmatrix} (a_{22}a_{33} - a_{23}a_{32}) & -(a_{21}a_{33} - a_{23}a_{31}) & (a_{21}a_{32} - a_{22}a_{31}) \\ -(a_{12}a_{33} - a_{13}a_{32}) & (a_{11}a_{33} - a_{13}a_{31}) & -(a_{11}a_{32} - a_{12}a_{31}) \\ (a_{12}a_{23} - a_{13}a_{22}) & -(a_{11}a_{23} - a_{13}a_{21}) & (a_{11}a_{22} - a_{12}a_{21}) \end{pmatrix}^T . \tag{10}$$

We see that the matrix elements look like two by two determinants. Also, the determinant D looks like a linear combination of 2×2 determinants.

As an aside, we note that if we regard the columns of a matrix as vectors, then we can use those vectors to form either a parallelogram or a parallelepiped if the vectors have two or three dimensions. The determinant is the area (or volume) of this shape. In two dimensions, if the area is zero, the vectors must be zero or parallel. In three dimensions, if the volume is zero, the vectors must be zero, parallel, or lie in the same plane. We can see that this will extend to any number of dimensions. If we use the the vectors of A as a basis set, and write down the "volume" of the shape with integration limits $V = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \dots$ the volume will be the $JacobianDeterminant$ of the transformation from the Cartesian unit basis vectors to the coordinates with basis vectors A.

Note that D can be written down in many different ways that still look like linear combinations of other 2×2 determinants. For instance, in writing down D, we have chosen the leading coefficients to be the entries in the first row. The reader can verify that the same thing can be written down with the leading coefficients to be *any row or column.* The reader can also verify that if we swap any row or column in the matrix, the D will change sign. This explains the motivation for developing some of the entities encountered in section 3.

It is obvious that doing this for a 4×4 matrix is going to be tough! However we already see a pattern emerging and this will help us out. For instance, if we denote the elements of A^{-1} as a^{-1}_{ij}, then the 2×2 determinants that occur in a^{-1}_{ij} for the inverse of a 3 by 3 matrix do not contain a_{ij}. There is no a_{11} in a^{-1}_{11} for instance. Later we shall look at this problem for the general $N \times N$ matrix.

So far we have not even mentioned the word eigenvector or eigenvalue. The word eigen is German for "belongs to" or "is special to". We must mention first that when we say a matrix is multiplied by a number, then every element is multiplied by the number. That is $(5A)_{ij} = 5a_{ij}$. As well as this we can add matrices and vectors element-wise, by which we mean that $(A + B)_{ij} = a_{ij} + b_{ij}$. Things look like normal algebra apart from the fact that matrix multiplication doesn't commute.

We shall see that (sometimes) given an $N \times N$ matrix A we can find a special set of vectors x_1, x_2, $\ldots x_N$ such that $Ax_n = \lambda_n x_n$ $(n = 1, 2, \ldots, N)$. Confused? There *no summation implied here*. The clue is that there are no subscripts on A, but as it upper case, A is a matrix. What $Ax_n = \lambda_n x_n$ says is that the matrix A times the nth eigenvector x_n is the nth eigenvalue times the nth eigenvector. At any rate, the λ_n are numbers associated with each x_n. What is going on? We are finding a set of vectors such that when A operates on them, their direction is unchanged. Any other vectors will change direction when operated on by A.

These eigenvectors are special to that particular A, for these particular vectors, the operator A looks like the identity I multiplied by a particular number (the eigenvalue). These special vectors are just scaled. In general, each eigenvector will have its own scaling or eigenvalue. In some cases all the eigenvalues might be the same, but mostly they will be different.

So, we have encountered four linear algebra problems so far. First: given a system of equations, what is the solution vector? Second: Given a matrix A, what is its determinant? Is it zero?. Third, given a Matrix A, what is its multiplicative inverse. Fourth: given a Matrix A what are its eigenvalues and eigenvectors.

3 Determinants and Inverse Matrices

This section is on how *not* to find matrix inverses! That is to say if you were to write a computer program to find the inverse of an $N \times N$ matrix, or solve a linear system of equations, the methods here would be far too inefficient. The point here is to look at the mathematical structure and admire the view. (From a physicists point of view determinants are important in the general theory of relativity and also in quantum mechanics.) We shall look at how we can make those numbers go crunch later.

We shall be using an onject called the *alternating tensor*. While at first sight, this may seem like a pointless abstraction, using this alternating tensor is important for gaining a good understanding of the fundamentals of linear algebra, and indeed, of tensor analysis. We include some tensor analysis in this article, as it is connected to ideas in linear algebra also.

The alternating tensor is an Nth rank tensor denoted $\epsilon_{ijklmn...}$ (sometimes e is used rather than ϵ). The alternating tensor is also known as the Levi-Civita symbol in tensor calculus). The value of ϵ is $\pm$ depending on whether the numbers $ijklmn\ldots$ are *even or odd permutations* of $123456\ldots N$. The value of $\epsilon_{ijklmn...}$ is $+1$ if the subscripts $ijk\ldots$ are an even permutation of $123456\ldots N$, and is -1 if the subscripts are an odd permutation. *The value of* $\epsilon_{ijklmn...}$ *is zero if the numbers* $ijklm\ldots$ *are* **not** *a permutation of* $123456\ldots N$. For two by two systems of equations we need just ϵ_{ij} with i and j being 1 or 2. For a three by three system of equations we will need ϵ_{ijk} where the indices are 1, 2, or 3, and so on.

We shall clarify what we mean by a permutation. If we write down the numbers 1 to N in order we can *define* $123\ldots N$ as an even permutation of the numbers 1 to N. Swapping adjacent pairs of numbers turns an even permutation into an odd permutation, or an odd permutation into an even one. Obviously, any even number of swaps *never* changes the evenness or oddness of a permutation, but any odd number of swaps *always* does.

If we swap any adjacent pair of numbers in $123\ldots N$, we will now have an odd permutation ($p1$ say). If we continue and swap another adjacent pair, we shall get another permutation of $123\ldots N$, which we shall call p_2, and so p_2 shall be an even permutation as there were two swaps.

Take nine numbers, say 952814376. Can we keep swapping adjacent pairs of numbers until we get 123456789? (call this p_0). Well, look this list of swaps. After swap 1 we have 952184376, then 951284376, 915284376, 915284376, 195284376, 195284376. After two more swaps the sequence of numbers can begin with 12, and so we can end up with 123456789 by this process. We shall suppose that this has taken us S swaps. (Note the reverse process gets us back to 952814376 and the number of swaps to get from 123456789 to 952814376 is also S: we can determine the evenness or oddness of a permutation by the number of swaps to get to p_0.)

However, we shall continue from 952814376 with 159284376, 152984376, 1528943764376. The result is the same as just swapping the 1 and the 9 in the first sequence 952814376. This took nine swaps of adjacent pairs and so $\epsilon_{952814376} = -\epsilon_{152894376}$. This illustrates that if we swap *any pair* of the subscript indices in $\epsilon_{ijklm...}$ and the result will be a change in sign.

Our example nine numbers 952814376 are an even permutation if it takes an even number of swaps to get to 123456789. Swap 1 and 9 in 952814376 we have 152894376. So it goes to $125894376 \rightarrow 123894576 \rightarrow 123498576 \rightarrow 123458976 \rightarrow 123456978 \rightarrow 123456798 \rightarrow 123456789$.

If the author has counted right, that is 8 swaps, so it is an even permutation and $\epsilon_{952814376} = 1$. Of course *if any pair of numbers is repeated we can never get to* $123\ldots N$! Such a list of numbers is *not* a permutation of the set of numbers $123\ldots N$ and then the value of alternating tensor $\epsilon_{ijk...}$ is zero.

The reader can show that the sign of the permutation as discussed is independent of the way the numbers are swapped to get to 12345 . . .. That is to say the sign of the permutation of a given set of integers us entirely unique.

Given any object with indices, if a pair of indices are swapped and the object is unchanged, the object is said to be *symmetric* with respect to that pair of indices. If a pair is swapped but only the sign of the object changes it is said to be *skew symmetric*. So, if we have a matrix A, it is symmetric if $a_{ij} = a_{ji}$ and is skew symmetric if $a_{ij} = -a_{ji}$. (Note that the diagonal elements of a skew symmetric matrix must be zero.) The alternating tensor is skew symmetric for any swap of any pair of indices. That is to say it is *totally* skew symmetric.

Now given a matrix A, we can think of it as a set of column vectors a_{i1} (column one), a_{i2}, and so on. Then we *define* the determinant of A as

$$DetA = \epsilon_{ijklm\ldots} a_{i1} a_{j2} a_{k3} a_{l4} a_{m5} \ldots \tag{11}$$

where we are using the summation convention. (Otherwise that would have a lot of $\sum$s!) We make the comment that given two matrices A and B then in the determinant of $A + B$ we would have terms in brackets $(a_{i1} + b_{i1})(a_{i2} + b_{i2} \ldots)$ and we can see that $Det(A + B) \neq DetA + DetB$. The determinant of the sum of two matrices is *not* the sum of the determinants.

Let's write this down for just a two by two matrix.

$$DetA = \epsilon_{11} a_{11} a_{12} + \epsilon_{12} a_{11} a_{22} + \epsilon_{21} a_{21} a_{12} + \epsilon_{22} a_{21} a_{22}. \tag{12}$$

Now 11 and 22 are not permutations of 12. The value of ϵ for these subscript pairs is zero — so we didn't even need to bother writing these terms down. Then 12 is an even permutation of 12 so ϵ is 1, and 21 is an odd permutation of 12 (one swap) so ϵ is -1. We immediately see that this is precisely the determinant of a 2×2 matrix. The reader can easily check out that it works for a 3×3 determinant.

It may seem a bit odd just to say we *define* a determinant in this way. However, we have seen some kind of motivation. Just see what happens next! We shall leave the reader to just play with this strange game, but at the same time we shall mention a few important things that can be found by playing such games. For instance, we could just as equally define $DetA$ in terms of rows rather than columns. That is to say

$$DetA = \epsilon_{ijklm\ldots} a_{1i} a_{2j} a_{3k} a_{4l} a_{5m} \ldots \tag{13}$$

The meaning is entirely unchanged. The reason is that all the combinations $a_{11} a_{22} \ldots$ appear in both definitions with the same signs.

In fact we may write $\alpha\beta \ldots$ for the column numbers as well as $ijk \ldots$ for the row numbers and have

$$D_{\alpha\beta\gamma\delta\epsilon\ldots} = \epsilon_{ijklm\ldots} a_{i\alpha} a_{j\beta} a_{k\gamma} a_{l\delta} a_{m\epsilon} \ldots \tag{14}$$

This is another totally skew symmetric object (equal to $\pm DetA$). So we can generalise the definition of the determinant to

$$DetA = \frac{1}{N!}\epsilon_{\alpha\beta\gamma\delta\ldots}\epsilon_{ijkl\ldots}a_{i\alpha}a_{j\beta}a_{k\gamma}a_{l\delta}a_{m\epsilon}\ldots \tag{15}$$

Where does the $N!$ come from? We have an $N \times N$ system, and clearly $\epsilon_{ijkl\ldots}\epsilon_{ijkl\ldots} = N!$. That is the total number of distinct permutations on N numbers 1 to N. For instance

$$\epsilon_{ijkl}\epsilon_{ijkl} = 24 = 4!$$

$$= \epsilon_{1234}\epsilon_{1234} + \epsilon_{1243}\epsilon_{1243} + \epsilon_{1324}\epsilon_{1324} + \epsilon_{1342}\epsilon_{1242} + \epsilon_{1423}\epsilon_{1423} + \epsilon_{1432}\epsilon_{1432} + \epsilon_{2134}\epsilon_{2134} + \epsilon_{2143}\epsilon_{2143}$$

$$+\epsilon_{2314}\epsilon_{2314} + \epsilon_{2341}\epsilon_{2341} + \epsilon_{2413}\epsilon_{2413} + \epsilon_{2341}\epsilon_{2341} + \epsilon_{3124}\epsilon_{3124} + \epsilon_{3142}\epsilon_{3142} + \epsilon_{3214}\epsilon_{3214} + \epsilon_{3241}\epsilon_{3241}$$

$$+\epsilon_{3412}\epsilon_{3412} + \epsilon_{3421}\epsilon_{3421} + \epsilon_{4123}\epsilon_{4123} + \epsilon_{4132}\epsilon_{4132} + \epsilon_{4213}\epsilon_{4213} + \epsilon_{4231}\epsilon_{4231} + \epsilon_{4312}\epsilon_{4312} + \epsilon_{4321}\epsilon_{4321}. \tag{16}$$

(*Always remember the summation convention!*)

Remember that if we swap any pair, the result is a change in sign. There is only one number x that does the following two tricks. Trick one is $x = -x$ and trick two is $x = kx$ where k is some (non zero) constant: that number is zero. So, by definition (using eqns.11, 13, or eqn.15), if any two rows (or two columns) of a matrix are equal, we can swap them and find $DetA = -DetA$. That is, if in A, *any two rows or any two columns are the same then the determinant is zero.* If so A has no inverse. Also by definition , if we multiply any row or column by some constant the determinant is multiplied by that constant (in complete contrast to multiplying a matrix by a number where all the entries are multiplied by the constant). It follows immediately that if *any* row/column is a linear combination of other rows/columns then the determinant is zero. It is glaringly obvious from the definition that if any row or column consists entirely of zeroes, then the determinant is zero.

What is the determinant of the product of two matrices? We shall only need to look at a 3×3 matrix (N!=6). Suppose that $C = AB$ or $c_{ij} = a_{ir}b_{rj}$. Then

$$DetC = \frac{1}{6}\epsilon_{\alpha\beta\gamma}\epsilon_{ijk}c_{i\alpha}c_{j\beta}c_{k\gamma}$$

$$= \frac{1}{6}\epsilon_{\alpha\beta\gamma}\epsilon_{ijk}a_{ir}b_{r\alpha}a_{js}b_{s\beta}a_{kt}b_{t\gamma}. \tag{17}$$

(Here r, s, and t are dummy indices. They *have* to have different names because it is a triple summation: remember no index appears more than twice.) Let's just rearrange this a little

$$DetC = \frac{1}{6}\epsilon_{ijk}a_{ir}a_{js}a_{kt}\epsilon_{\alpha\beta\gamma}b_{r\alpha}b_{s\beta}b_{t\gamma}. \tag{18}$$

Now we multiply both sides by 6, or should I say $\epsilon_{rst}\epsilon_{rst}$.

$$6DetC = \frac{1}{6}\epsilon_{ijk}\epsilon_{rst}a_{ir}a_{js}a_{kt}\epsilon_{\alpha\beta\gamma}\epsilon_{rst}b_{r\alpha}b_{s\beta}b_{t\gamma} = DetA \times 6DetB. \tag{19}$$

Now, it takes a while to get the hang of this kind of stuff, but is that cool or what? If $C = AB$ then $DetC = DetADetB$. If the matrix has a factor with no inverse, the matrix has no inverse.

Now, we come across a simple looking beastie — the Kronecker delta. (Kronecker made important contributions to tensor analysis.) It is written δ_{ij} which is equal to 1 if $i = j$ and zero otherwise. Yes, it is an identity matrix. We define the following entity which we call the ϵ *system*

$$\epsilon_{ijk\ldots}^{\alpha\beta\gamma\ldots} = \epsilon_{ijk\ldots}\epsilon_{\alpha\beta\gamma\ldots} = Det\begin{pmatrix} \delta_{i\alpha} & \delta_{i\beta} & \delta_{i\gamma} & \ldots \\ \delta_{j\alpha} & \delta_{j\beta} & \delta_{j\gamma} & \ldots \\ \delta_{i\alpha} & \delta_{k\beta} & \delta_{k\gamma} & \ldots \\ \vdots & \vdots & \vdots & \end{pmatrix}. \tag{20}$$

Now the Kronecker δ is the identity, and the identity transformation is the same in any coordinate system. We don't even have to transform it to recognise its tensor character. Does this relation make ϵ a tensor too? The answer is "yes and no"...When we swap rows and columns in in a matrix it is like renaming the x axis to be the y axis and so on. That it is to say it is a reflection. So an even permutation is like a rotation, and an odd permutation is like a combination of a reflection and a rotation (an *improper rotation*). Strictly speaking, because of the change in sign under improper rotations the ϵ system is a *psuedotensor*.

We have almost forgotten that determinants are also written as $DetA = |A|$ and so eqn.20 can also be written as

$$\epsilon_{ijk\ldots}^{\alpha\beta\gamma\ldots} = \epsilon_{ijk\ldots}\epsilon_{\alpha\beta\gamma\ldots} = \begin{vmatrix} \delta_{i\alpha} & \delta_{i\beta} & \delta_{i\gamma} & \ldots \\ \delta_{j\alpha} & \delta_{j\beta} & \delta_{j\gamma} & \ldots \\ \delta_{i\alpha} & \delta_{k\beta} & \delta_{k\gamma} & \ldots \\ \vdots & \vdots & \vdots & \end{vmatrix}. \tag{21}$$

What is going on here is that if we start with the identity ($DetI = 1$) and start swapping the rows and columns we get the product of the alternating tensors. In three dimensional vector analysis, there are a couple of very useful identities. Here we can just write out these determinants in full to find that

$$\epsilon_{ijk}\epsilon_{iqr} = \delta_{jq}\delta_{kr} - \delta_{jr}\delta_{kq}. \tag{22}$$

Also we find

$$\epsilon_{ijk}\epsilon_{ijn} = 2\delta_{kn}. \tag{23}$$

These relations are sometimes called ϵ tensor contractions.

Well, that was a lot of fun and games, but where has it got us? Given all the above, the solution to an $N \times N$ system of equations almost just gives up and writes itself down!

We are after the inverse of a matrix. If we rewrite eqn.15 as follows,

$$DetA = \frac{1}{N} a_{i\alpha} \frac{1}{(N-1)!} \epsilon_{\alpha\beta\gamma\delta\ldots} \epsilon_{ijkl\ldots} a_{j\beta} a_{k\gamma} a_{l\delta} a_{m\epsilon} \ldots, \tag{24}$$

we see that we have just "pulled out" a matrix A to the front. This is the clue to see where things are going. As soon as we do this we see that the $N \times N$ determinant is the (weighted) sum of a lot of $(N-1) \times (N-1)$ determinants. These $(N-1) \times (N-1)$ determinants are (weighted) sums of $(N-2) \times (N-2)$ determinants and so on. (In fact the word matrix is Latin for mother because it has lots of little determinants as its children!) Let's make things a little simpler and re-examine eqn.11.

$$\begin{aligned} DetA &= a_{1i} \epsilon_{ijklm\ldots} a_{2j} a_{3k} a_{4l} a_{5m} \ldots \\ &= a_{11} \epsilon_{1jklm\ldots} a_{2j} a_{3k} a_{4l} a_{5m} \ldots + a_{12} \epsilon_{2jklm\ldots} a_{2j} a_{3k} a_{4l} a_{5m} \ldots \\ &\quad + a_{13} \epsilon_{3jklm\ldots} a_{2j} a_{3k} a_{4l} a_{5m} \ldots + \ldots \end{aligned} \tag{25}$$

The sum that a_{11} multiplies (in effect) has no terms from column one in it. That is because if any of j, or k, or l, or m is equal to one we have

$$\epsilon_{11klm} = \epsilon_{1j1lm} = \epsilon_{1jk1m} = \epsilon_{1jkl1} = 0. \tag{26}$$

if $j = 1$. Similarly the coefficient a_{12} multiplies a sum with no terms from the second column, the coefficient a_{13} multiplies a sum with no terms from the third column. None of the coefficients multiply any sum containing anything in the first row. All the sums that the a_{1i} multiply are $\pm$ determinants of an $(N-1) \times (N-1)$ matrix which is just the matrix A with row one and column one missing! From the above, we could have chosen *any* row, and we could equally do the same evaluation using columns instead of rows since, for instance.

$$\begin{aligned} DetA &= a_{i1} \epsilon_{ijklm\ldots} a_{j2} a_{k3} a_{l4} a_{m5} \ldots \\ &= a_{11} \epsilon_{1jklm\ldots} a_{j2} a_{k3} a_{l4} a_{m5} \ldots + a_{21} \epsilon_{2jklm\ldots} a_{j2} a_{k3} a_{l4} a_{m5} \ldots \\ &\quad + a_{31} \epsilon_{3jklm\ldots} a_{j2} a_{k3} a_{l4} a_{m5} \ldots + \ldots \end{aligned} \tag{27}$$

If we "strike out" row i and column j from a determinant, the resulting $(N-1) \times (N-1)$ determinant is called a *Minor* and is denoted M_{ij}. The *cofactor* A_{ij} is then

$$A_{ij} = (-1)^{i+j} M_{ij}. \tag{28}$$

If you see upper case letters with subscripts you know they are minors or cofactors. They are numbers, but there array nature is clear as well. Comparing with eqn.24,

$$A_{i\alpha} = \frac{1}{(N-1)!} \epsilon_{\alpha\beta\gamma\delta\ldots} \epsilon_{ijkl\ldots} a_{j\beta} a_{k\gamma} a_{l\delta} a_{m\epsilon} \ldots \tag{29}$$

where none of the $jkl\ldots$ are equal to i and none of the $\beta\gamma\ldots$ are equal to α.

In the following, we shall find that

$$A_{ik}a_{ij} = DetA \times \delta_{kj}. \tag{30}$$

The identity matrix on the r.h.s. means we have found our inverse! It is well worthwhile considering just a 3×3 system.

$$A_{i\alpha} = \frac{1}{2!}\epsilon_{ijk}\epsilon_{\alpha\beta\gamma}a_{j\beta}a_{k\gamma}, \tag{31}$$

we can form $B_{\alpha\omega}$

$$B_{\alpha\omega} = A_{i\alpha}a_{i\omega} = \epsilon_{ijk}\epsilon_{\alpha\beta\gamma}a_{j\beta}a_{k\gamma}a_{i\omega}. \tag{32}$$

So for instance B_{11} is

$$B_{11} = \frac{1}{2}\epsilon_{ijk}\epsilon_{1\beta\gamma}a_{i1}a_{j\beta}a_{k\gamma}, \tag{33}$$

where $jk \neq i$ and $\alpha\beta$ are 23 or 32. Whatever the value of i, neither $a_{j\beta}$ or $a_{k\gamma}$ are ever equal to a_{i1}. Examining B_{11} in detail

$$B_{11} = \frac{1}{2}\epsilon_{ijk}\epsilon_{1\beta\gamma}a_{i1}a_{j\beta}a_{k\gamma}$$

$$= \frac{1}{2}(\epsilon_{ijk}\epsilon_{123}a_{i1}a_{j2}a_{k3} + \epsilon_{ijk}\epsilon_{132}a_{i1}a_{j3}a_{k2})$$

$$= \frac{1}{2}(\epsilon_{ijk}a_{i1}a_{j2}a_{k3} - \epsilon_{ijk}a_{i1}a_{j3}a_{k2}). \tag{34}$$

Now, $\epsilon_{ijk}a_{i1}a_{j2}a_{k3} = DetA$ by definition, and $\epsilon_{ijk}a_{i1}a_{j3}a_{k2} = -DetA$ because column 2 and 3 have been swapped. That is $B_{11} = DetA$. Now

$$B_{22} = \frac{1}{2}\epsilon_{ijk}\epsilon_{2\beta\gamma}a_{i2}a_{j\beta}a_{k\gamma}$$

$$= \frac{1}{2}(\epsilon_{ijk}\epsilon_{213}a_{i2}a_{j1}a_{k3} + \epsilon_{ijk}\epsilon_{231}a_{i2}a_{j3}a_{k1})$$

$$= \frac{1}{2}(-\epsilon_{ijk}a_{i2}a_{j1}a_{k3} + \epsilon_{ijk}a_{i2}a_{j3}a_{k1}). \tag{35}$$

Because the indices are dummy ones we can rename them. For instance in the first sum we can put $i \to j$ and $j \to i$, In the second sum we can write $k \to i$, $i \to j$, $j \to k$. So we can equally well write this as

$$B_{22} = \frac{1}{2}(-\epsilon_{jik}a_{i1}a_{j2}a_{k3} + \epsilon_{kij}a_{i1}a_{j2}a_{k3})$$

$$= \frac{1}{2}(+\epsilon_{ijk}a_{i1}a_{j2}a_{k3} + \epsilon_{ijk}a_{i1}a_{j2}a_{k3}). \tag{36}$$

So, we can see all the diagonal elements of $B_{(\alpha,\omega)}$ will be $DetA$. This is fairly obvious for the $N \times N$ case. For any particular $\beta, \gamma, \epsilon \ldots$ with $\alpha = \omega$ we get the definition of $DetA$. There will be $(N-1)!$ terms.

For off diagonal terms consider just B_{12}.

$$B_{12} = \frac{1}{2}\epsilon_{ijk}\epsilon_{1\beta\gamma}a_{i2}a_{j\beta}a_{k\gamma}$$

$$= \frac{1}{2}(\epsilon_{ijk}\epsilon_{123}a_{i2}a_{j2}a_{k3} + \epsilon_{ijk}\epsilon_{132}a_{i2}a_{j3}a_{k2}).$$

$$= \frac{1}{2}\epsilon_{ijk}a_{i2}a_{j2}a_{k3} - \frac{1}{2}\epsilon_{ijk}a_{i2}a_{j3}a_{k2}). \tag{37}$$

Both terms are determinants of a matrix with two columns the same. If we swap the identical columns we get $Det = -Det = 0$. This is clearly true for the $N \times N$ version of $B_{\alpha\omega}$. If $\alpha \neq \omega$, we always get a sum of determinants which have two identical columns and so are all exactly zero. If $\alpha = \omega$ no two columns are the same, and by renaming the indices we see that they are all equal to $DetA$ and there $(N-1)!$ of them. This proves eqn.30.

Now we can solve our system of $N \times N$ equations! Yep, that's right, we can just write down the solution of any system of linear equations in this way. If

$$a_{ij}x_j = b_i, \tag{38}$$

we only need to multiply (and sum over)

$$A_{ik}a_{ij}x_j = DetA\delta_{kj}x_j = A_{ik}b_i$$

$$x_k = \frac{1}{DetA}A_{ik}b_i = \frac{1}{DetA}A^T_{ki}b_i,$$

or

$$x_i = a^{-1}_{ij}b_j \tag{39}$$

Note that eqn.8 and eqn.10 are precisely of this form. In the last step of eqns.39 we have renamed the dummy index i to j and renamed the free index k to i. (We can rename a free index to anything which is not a dummy, and after the previous step the dummy had been renamed $j\ldots$) The meaning of the expression is unchanged. Also we have the elements of A^{-1} as expressed in terms of cofactors of A as a^{-1}_{ij}. Now we have $A^{-1}A = I$, the reader can confirm that $AA^{-1} = I$.

At this point, we can look at the transpose operation a little more closely. We see immediately that $DetA^T = DetA$. If we expand $DetA$ via the first row we get exactly the same as when we expand $DetA^T$ via the first column. If $C = AB$ where A, B, and C are matrices then $c_{ij} = a_{ik}b_{kj}$. If we transpose both sides we get $c_{ji} = a_{jk}b_{ki} = b^T_{jk}a^T_{ki}$. That is $C^T = B^TA^T$. Naturally, if we have matrices E, F G, then $(EFG)^T = G^TF^TE^T$. Now consider $(A^{-1}A)^T = I^T = I$. We immediately have $A^T(A^{-1})^T = I$. That is $(A^{-1})^T = (A^T)^{-1}$. We know this because that we get the identity if we pre- or post-multiply a matrix by an inverse. The transpose of the inverse is the inverse of the transpose. We shall denote it A^{-T}. The reader can also check the cofactors of A^T, and shall not only see that $(A^{-1})^T = (A^T)^{-1}$, but that (unless A is known to be symmetric) $A^{-T}A \neq I$.

Finally, if we regard x^T as being a row vector, with the same elements as a vector x. Then x^TA^T is a row vector with exactly the same elements (left to right) as the column vector given by Ax. There we have it, no matter how large the system, we have written down the solution to $Ax = b$.

Now, the reader can confirm, that it takes the order of N^3 operations to calculate a determinant of an $N \times N$ matrix. To calculate an inverse, we need N^2 $(N-1) \times (N-1)$ Determinants! If we want an inverse matrix and N is large, we are in dire sraights. We shall see that calculating a determinant, an inverse, or a solution vector for an $N \times N$ syste, of equations can all be reduced to processes of order N^3 (or less for "sparse systems"). The use of determinants as described in this chapter is of no use in practical calculation. That said, it is important in the analysis of systems of equations, and also important in the theories of general relativity and quantum mechanics.

4 Accuracy

We make a bit of a detour in this section. The reason for this is that in the following sections on the solution of linear equations and matrix inversion we write down the methods as if there were no bear-traps for the unwary to fall into. We mention the methods (called pivoting) to avoid bear traps later. Here we wish to point out that a simplistic approach can lead to disastrous results. If your bridge design relies on a simplistic approach to the solution of linear equations, your bridge might well fall down with serious consequences (quite apart from the jail time you may face because of your miscalculations).

In practice, solving even a modest system of equations will always be done on a computer. Computer programs to do this (and much more) are provided by $LAPACK$: a package of computer programs freely available from *netlib.org*. This latter site provides an enormous amount of other numerical software and should be the first port of call when looking to do any numerical analysis involving linear algebra. It is possible to do things in *arbitrary precision* arithmetic on a computer (to a million decimal places if you want). This is rarely practical, and it is quite often the case that 4 bytes or 8 bytes are used to store a number. (Obviously very high arbitrary precision calculations are also very slow.) In FORTRAN, these 4 and 8 byte representations of real numbers are called REAL, and DOUBLE PRECISION. In $C/C++$ they are called float and double. Holding a number in just 4 bytes is *very* often too imprecise for any serious calculation. The 8 byte DOUBLE PRECISION or double should be the default. (Many compilers have higher precision options.)

The reader will have obviously come across 2×2 systems at school. We will imagine we have to solve the following on a computer that can only hold numbers to six digits. We solve it both the "school way" and the inverse matrix way. We shall get exactly the same answer — which just happens to be right! The lesson to take away from this is *not* that both methods are equally good. The lesson here is that we can see we we are sailing *very close to disaster* both ways! Consider

$$0.780x_1 + 0.563x_2 = 0.217$$

$$0.913x_1 + 0.659x_2 = 0.254. \tag{40}$$

We divide both by the first coefficient to get

$$x_1 + 0.721795x_2 = 0.278205$$

$$x_1 + 0.721796x_2 = 0.278204. \tag{41}$$

(The determinant is 0.514020-0.514019 and so the system is very close to not having an inverse. It is close to to being *singular*).

From eqn.41 we get $10^{-6}x_2 = -10^6$, or $x_2 = -1.0$. But look at where we get $10^{-6}x_2$ from. That is 0.721796-0.721795. Note, if we had more digits this would read 0.721796xxxx-0.721795xxxx. By writing "xxxx" we just mean we haven't a clue what those digits would have been. Considering rounding up and down means we could equally have got 0.721795xxxx-0.721794xxxxx. It could also have been 0.721796xxxx-0.721794xxxx or 0.721795xxxx-0.721795xxxx. Although we get $1\times10^{-6}x_2$= we really have no idea whether it's 1.812×10^-6x_2 or $0.452 \times 10^{-6}x_2$ =. The same thing goes for the right hand side with 0.278204-0.278205. The truth is, though we get $x_2 = -1$ as the numerical answer, we do know we don't have any idea what x_2 actually is! We are on our last digit of accuracy. *Small differences between (relatively) large numbers mean a drastic loss in accuracy.* In this case the 0.278204 is relatively large compared to 0.000001.

We "carry on regardless"[1] in our hapless comedic solution, and now we get x_1-0.721795=.278205, and x_1=1.0. Now suppose instead we do this via the matrix inverse method. We have already calculated the determinant, so we have

$$A^{-1} = 1000000 \begin{pmatrix} 0.659 & -0.913 \\ -0.563 & 0.780 \end{pmatrix}^T = \begin{pmatrix} 659000 & -563000 \\ -913000 & 780000 \end{pmatrix}. \tag{42}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 659000 & -563000 \\ -913000 & 780000 \end{pmatrix} \begin{pmatrix} 0.217 \\ 0.254 \end{pmatrix} = \begin{pmatrix} 143003 - 143002 \\ -198121 + 198120 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{43}$$

In fact, the reader may check by hand that (1.0, -1.0) is the *exact* solution. In general, we cannot know what the exact answer is. We can plug the numbers back in to the original and check we get the difference between this and the original right hand side (on finding our values x the difference $Ax - b$ is called the *residual*). This gives us a handle on the accuracy, right?

But, before considering this, there is another matter. What would have happened the original data for the matrix element 0.563 had been 0.562999 and we (or someone else) had rounded this number up up to 0.563? (This kind of thing could easily happen when numbers are given by a computer program using formatted output and written to a data file.)

[1]Carry on Regardless was a 1960's comedy film.

Supposing the numbers in the matrix above were slightly different from the intended values. That is, for whatever reason, we had tiny errors in the entries. Then we would have, instead of A, a matrix $A + E$, where all the entries in E are small. We "pretend" that we actually have A^{-1} and x. If we were given an $A + E$ instead, we we would find a vector $x' + (A^{-1}E)x' = x$. Our calculated x' would be different to x by a vector $(A^{-1}E)x'$. In the above calculation, the matrix elements of A^{-1} were getting on toward a million. Our tiny error in rounding 0.562999 to 0.563 would mean the x values we find as a solution are worthless!

We take the following example from Van Loan [1]. (See also Golub and Van Loan's "Survey of Matrix Computations" [4].)

Suppose We had exactly the same matrix as above. Then, we got two different answers from two different computer programs, different compilers, and different operating systems. That is we got

$$x' = \begin{pmatrix} 0.341 \\ -0.087 \end{pmatrix}, \quad x'' \begin{pmatrix} 0.999 \\ -1.00 \end{pmatrix}. \tag{44}$$

How do we choose which is the most accurate solution? We could call $r = Ax - b$ the *residual*. Surely the residual will tell us which of the two answers is better? Whichever has the smallest residual would be be best, right? If we calculate the residuals we get

$$r' = \begin{pmatrix} 0.0000001 \\ 0.0 \end{pmatrix}, \quad r'' \begin{pmatrix} 0.001343 \\ 0.001572 \end{pmatrix}. \tag{45}$$

In other words, the size of the residual is no guarantee whatsoever of the accuracy. The largest residual comes from the most accurate solution. (In fact, in some cases the smaller residual solution might be "best". It depends on what you need those values of x for.

At any rate, we see that knowing how good a solution is isn't a simple problem. Later, in §16, we introduce the *condition number* of the matrix which will give us a good handle on the accuracy of the solution.

As we shall see later, we can find the inverse of a matrix and find our x that way. This can be done by what is called a LU decomposition (or factorisation). We also can just solve for x using Gaussian elimination. We shall simply state for the moment that the condition number of a product of two matrices is larger than the condition number of either. That is the condition number of the inverse is larger than that of L or U. The condition number of a Gaussian elimination only depends on L. If the inverse of a matrix is not actually required (it is required for many purposes though) solving by Gaussian elimination is the best option. In general it will give a more accurate solution.

[1] Charles Van Loan, A Survey of Matrix Computations, Cornell Theory Centre Technical Report

5 Gaussian Elimination

For large systems of equations using determinants as in eqn.39 would take the order of N^5 operations. We write "the order of N^5" as $O(N^5)$. If we needed to invert a very large system, this would be for too much computation, and may take a lifetime! Given an N by N system $Ax = b$ we can solve for x in a manner that takes the order of N^3 operations. We can also find A^{-1} and this takes a similar number of operations.

Before we begin we comment on another property of determinants. Suppose we add some multiple λ of one column (or row) to another. We have, for example

$$\epsilon_{ijklm} \ldots a_{i1} a_{j2} (a_{k3} + \lambda a_{k4}) a_{l4} a_{m5} + \ldots$$

$$= \epsilon_{ijklm} \ldots a_{i1} a_{j2} a_{k3} a_{l4} a_{m5} + \ldots + \lambda \epsilon_{ijklm} \ldots a_{i1} a_{j2} a_{k4} a_{l4} a_{m5} + \ldots$$

$$= \epsilon_{ijklm} \ldots a_{i1} a_{j2} a_{k3} a_{l4} a_{m5} + \ldots = |A| \tag{46}$$

Here we have added λ times column 4 to column 3. This gives us the sum of two determinants, but the second determinant is zero because it has two copies of column 4. The determinant is entirely unchanged by adding so many times column 4 to column 3

In the following we shall consider just a 4×4 system by way of illustration. At first we shall proceed as if nothing could possibly go wrong. We shall also proceed as if space (RAM) and timing were not a consideration. We solve this system by Gaussian elimination.

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 + a_{14} x_4 = b_1$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 + a_{24} x_4 = b_2$$

$$a_{31} x_1 + a_{32} x_2 + a_{33} x_3 + a_{34} x_4 = b_3$$

$$a_{41} x_1 + a_{42} x_2 + a_{43} x_3 + a_{44} x_4 = b_4. \tag{47}$$

We divide row one by a_{11} row 2 by a_{21} row 3 by a_{31} and so on. Then $b_1 \to b_1/a_{11}$, $a_{32} \to a_{32}/a_{31}$, and so on. We put $a'_{32} = a_{32}/a_{31}$, $b'_1 b_1/a_{11}$, ..., and on doing all these divisions we get

$$1 \times x_1 + a'_{12} x_2 + a'_{13} x_3 + a'_{14} x_4 = b'_1$$

$$1 \times x_1 + a'_{22} x_2 + a'_{23} x_3 + a'_{24} x_4 = b'_2$$

$$1 \times x_1 + a'_{32} x_2 + a'_{33} x_3 + a'_{34} x_4 = b'_3$$

$$1 \times x_1 + a'_{42} x_2 + a'_{43} x_3 + a'_{44} x_4 = b'_4. \tag{48}$$

Now we subtract the first row from all the others (again we keep the same letters unless something is one or zero). The determinant of this system will be entirely unchanged.

$$1 \times x_1 + a'_{12} x_2 + a'_{13} x_3 + a'_{14} x_4 = b'_1$$

$$0 \times x_1 + a''_{22} x_2 + a''_{23} x_3 + a''_{24} x_4 = b''_2$$

$$0 \times x_1 + a''_{32}x_2 + a''_{33}x_3 + a''_{34}x_4 = b''_3$$
$$0 \times x_1 + a''_{42}x_2 + a''_{43}x_3 + a''_{44}x_4 = b''_4. \tag{49}$$

Now we shall leave row one alone. We divide row 2 by a''_{22}, then row 3 by a''_{32}, and row 4 by a''_{42}. Next we subtract row two from all the rows below it.

$$1 \times x_1 + a'_{12}x_2 + a'_{13}x_3 + a'_{14}x_4 = b'_1$$
$$0 \times x_1 + 1 \times x_2 + a''_{23}x_3 + a''_{24}x_4 = b''_2$$
$$0 \times x_1 + 0 \times x_2 + a'''_{33}x_3 + a'''_{34}x_4 = b'''_3$$
$$0 \times x_1 + 0 \times x_2 + a'''_{43}x_3 + a'''_{44}x_4 = b'''_4. \tag{50}$$

It is obvious what is going on, next we divide row 3 by a'''_{33} and row 4 by a'''_{43} , and then subtract row 3 from the only remaining row below it. Finally we divide row through by a''''_{44} to get.

$$1 \times x_1 + a'_{12}x_2 + a'_{13}x_3 + a'_{14}x_4 = b'_1$$
$$0 \times x_1 + 1 \times x_2 + a''_{23}x_3 + a''_{24}x_4 = b''_2$$
$$0 \times x_1 + 0 \times x_2 + 1 \times x_3 + a''''_{34}x_4 = b''''_3$$
$$0 \times x_1 + 0 \times x_2 + 0 \times x_3 + 1 \times x_4 = b'''''_4. \tag{51}$$

We drop the primes, on the understanding that n one of the coefficients or the b_i are anything like the ones we started with. So now we have $x_4 = b_4$, then we have $x_3 = b_3 - a_{34}x_4$, then we have $x_2 = b_2 - a_{23}x_3 - a_{24}x_4$ and finally $x_1 = b_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4$. This is exactly what we called "the school way" before, and is called Gaussian elimination. Of course we could have written all this in matrix vector form.

Before continuing we shall make some fairly obvious statements. We could have started with the bottom row and worked up. We could have started with the first column and worked right, and we could have started with the last column and worked left. We could start with any row or column and proceed in any manner we please! However, if we write a computer program it's fairly easy to keep all the counters straight if we use this first row down approach. The row and column numbers step up in ones.

Now, we said we would act as if nothing could possibly go wrong. It is obvious what could go wrong: we could end up dividing by zero! The aim of the first step is to get all the elements in the column one below row one to be zero. So if a_{41} already is zero, or a_{31} is zero then all we have to do is leave the row alone. The work has already been done for us. Similarly, we want all the coefficients below row 2 of column 2 to be zero. Again if a_{32} is zero, we can leave out the step of dividing by row 3 of both sides by a_{32}.

So far, so good. But what if a_{11} were zero? If we were doing it by hand we could proceed in any manner we please. We could look at the matrix and decide whether we want to swap row 1 with some row where the first element is non zero, or swap column 1 with a comlumn

which has a non zero first (top) element. Doing things by hand, we may be able to see which row or column we must swap, and may do as we please, but we have to automate this.

We could subtract row-wise (top down) and swap columns, subtract row-wise and swap rows, subtract column-wise (left to right) and swap columns, or subtract column-wise and swap rows. If we are automating things, it would be easiest to choose one of the above four options and stick to it. (In memory, a 3×3 array is stored column-wise in fortran. That is, the matrix has nine elements in memory stored as $a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{a32}, a_{13}, a_{23}, a_{33}$. In C/C++ the array is stored row-wise as $a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{a23}, a_{31}, a_{32}, a_{33}$. This might influence the decision, but the counter arithmetic is trivial compared to the double precision operations.)

Whichever we choose, what happens to the vector x or the vector b when we swap rows or columns in A? We shall leave this for later, and for the moment we suppose we are subtracting row-wise and swapping columns.

We are in severe trouble if *any* of the diagonal elements is zero. Suppose all the first row elements are zero apart from a_{14}. (If all are zero, the determinant is zero and there is no solution). All we do is swap column 1 with column 4, and all is well.

We are emphatically *not* out of trouble though. Supposing that none of the first row are zero, but that they are all small apart from a_{14}. We divide by a_{11} and then $a_{14} \to a_{14}/a_{11}$ is huge. Now the trouble with this is that we have limited precision. We can have huge numbers but only with a fixed number of digits. If we were limited to just 6 digits, then 123123123123 would be represented as 123123E11. This is 123123000000. When we subtract the new a_{14} we would be subtracting the wrong number. Worse still our low precision would mean that the result of, say 10000-123123000000 would be -123123000000. As we have seen already in §4, *small differences between large numbers are trouble.* $LAPACK$ routines [2] often require a subroutines such as d1mach [3]. These provide numbers such as the largest number that, when added to 1.0 gives the answer 1.0. (This will vary from machine to machine.) These numbers allow the algorithms to maintain as high an accuracy as is possible.

If we choose to swap rows or columns to avoid dividing by zero, we must also swap rows or columns to avoid this kind of numerical error. So if we are swapping columns, when we come across the point where we would be dividing by a diagonal element in the above example, we look along the column to the right and find the a_{ij} that has the largest absolute value. Whatever that column is in, we swap with the current column.

Often, the choice is made to write the program such that either rows are always swapped or columns are always swapped. If this is the case, this algorithm step is known as *partial*

[2] Lapack is freely avaible from netlib.org and is often packaged with Linux distributions.

[3] This is part of the BLAS package, also available from netlib and often packaged with linux distributions

pivoting and is commonly used. If we are swapping rows instead, we look for the largest (in magnitude) element in the column below the current diagonal. We could do a mixture where we find the largest number to the right and below the diagonal element and swap a row *and* a column accordingly. This is *total pivoting*. Mostly, partial pivoting is used.

Now we could just write down the matrix/vector equation and see what the equivalent system is when a swap is made. However we shall formalise this in terms of matrices. The reader shall remember the Kronecker delta representation of row and column permutations of eqn.20. We proceed with a simple 3×3 example. That is

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}. \tag{52}$$

If we take the identity I, swap row 2 and row 3, and pre-multiply A with the result, this effectively swaps rows 2 and 3 in A. In the case of $Ax = b$, if we pre-multiply with this permutation of the identity on both sides, the elements of x are unchanged. The right hand side column vector b has elements 2 and 3 swapped.

Now consider

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{13} & a_{12} \\ a_{21} & a_{23} & a_{22} \\ a_{31} & a_{33} & a_{32} \end{pmatrix}. \tag{53}$$

If we swap column 2 and column 3 in the identity but now post multiply A, columns 2 and 3 of A are swapped. Now if we post multiply twice, we swap them back again. That is the application of two successive swaps with the same *permutation matrix* must be the identity. Lets look at $Ax = b$.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}. \tag{54}$$

Now we post multiply A by the first permutation matrix and pre multiply b by the second. We get

$$\begin{pmatrix} a_{11} & a_{13} & a_{12} \\ a_{21} & a_{23} & a_{22} \\ a_{31} & a_{33} & a_{32} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}. \tag{55}$$

So if we swap two columns in the matrix, and swap the same numbered rows in the x vector, we get a representation of the original system of equations. So, we can sum this up as follows. Swap rows in matrix A, swap rows in vector b: Swap columns in matrix A swap rows in vector x.

We finish off with just a couple of remarks. If you wrote a computer program to do this you wouldn't swap anything at all! What you would do is keep track of all the swaps that *would* have been made. The other thing is, that you might think you need an inverse matrix because you have the same matrix but a lot of right hand sides. No: just use Gaussian elimination and do exactly the same operation for each right hand side. The $LAPACK$ set of routines will do all this for you.

Most of these routines will destroy the matrix and the vector you give it! These are built for robustness, speed, and accuracy on large systems. As well as that they are meant to be efficient in memory too. The matrix on the left hand side and the vector on the right are gradually overwritten so that only one matrix needs to be stored in RAM.

6 Matrix Inversion and the LU Decomposition

Forgetting pivoting for the moment, when we did our Gaussian elimination of a four by four matrix we ended up with somethin looking like

$$\begin{pmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} \tag{56}$$

Now the matrix on the left of eqn.56 is called *upper triangular*. We shall denote it U. It is obvious that the only the diagonal elements and those above the diagonal can be non zero in an upper triangular matrix. Since all the diagonal elements are equal to 1, it is a special case of the set of upper triangular matrices and is called a *unitriangular* matrix. Similarly we can have lower triangular matrices. It is easily verified that an upper triangular matrix multiplied by another upper triangular matrix gives us an upper triangular matrix. A similar result holds for lower triangular matrices. If we multiply a lower by an upper triangular matrix we get a square matrix.

To find the inverse of of some matrix A we need to take the lid off what was happening to the b vector in $Ax = b$. and how it got to be the c vector in Ux=c. Initially we start off with

$$rhs. = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}. \tag{57}$$

In matrix notation, we then multiplied by $diag(1/a_{11}, 1/a_{21}, 1/a_{31}, 1/a_{41})$ where *diag* means a matrix which only has nonzero elements on the diagonal. The diagonal elements in $diag(\alpha, \beta, \gamma, \ldots, \omega)$ go from top left to bottom right with $a_{11} = \alpha$ and $a_{NN} = \omega$. The identity is a special diagonal matrix. We now have

$$rhs. = \begin{pmatrix} 1/a_{11} & 0 & 0 & 0 \\ 0 & 1/a_{21} & 0 & 0 \\ 0 & 0 & 1/a_{31} & 0 \\ 0 & 0 & 0 & 1/a_{41} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}. \tag{58}$$

The matrix M_{r1} that subtracts the elements in row one from all the other three rows is

$$M_{r1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}. \tag{59}$$

Clearly it has an inverse

$$M_{r1}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}. \tag{60}$$

A matrix of this form, that has 1 for each diagonal element and has non zero numbers *only below the diagonal of just one column* is called an *atomic* upper/lower matrix or a Gauss transformation matrix. The matrix that transforms b shall be built one atom at a time.

The combination of the first two steps getting us from eqn.47 to eqn.49 gives us

$$rhs = \begin{pmatrix} 1/a_{11} & 0 & 0 & 0 \\ -1/a_{11} & 1/a_{21} & 0 & 0 \\ -1/a_{11} & 0 & 1/a_{31} & 0 \\ -1/a_{11} & 0 & 0 & 1/a_{41} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}. \tag{61}$$

Obviously, we do not store this in a square array, nor do we multiply lots of numbers by zero and add all the zeroes to get zero. Our CPU and our RAM have better things to do.

After the next two steps we have

$$rhs = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/a_{22} & 0 & 0 \\ 0 & -1/a_{22} & 1/a_{32} & 0 \\ 0 & -1/a_{22} & 0 & 1/a_{42} \end{pmatrix} \begin{pmatrix} 1/a_{11} & 0 & 0 & 0 \\ -1/a_{11} & 1/a_{21} & 0 & 0 \\ -1/a_{11} & 0 & 1/a_{31} & 0 \\ -1/a12 & 0 & 0 & 1/a_{41} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}. \tag{62}$$

Clearly the next two steps are the same as

$$rhs \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/a_{33} & 0 \\ 0 & 0 & -1/a_{33} & 1/a_{43} \end{pmatrix} \times rhs. \tag{63}$$

In the final two steps we have just

$$rhs \rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/a_{44} \end{pmatrix} \times rhs. \tag{64}$$

The product of all these transformations is that

$$rhs. = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \rightarrow rhs \begin{pmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}. \tag{65}$$

So, after doing our Gaussian elimination, we end up with eqn.51 which consists of an upper triangular matrix U multiplied by x on the left, and a lower triangular matrix times b on the right.

Now we ask, what is the inverse of the upper triangular matrix on the left? Lets look at the determinant of *any* upper triangular matrix.

$$DetU = \begin{vmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{vmatrix} \tag{66}$$

Instead of expanding it via the first row, we expand it via the first column. All the coefficients are zero apart from u_{11}. So, we get

$$DetU = u_{11} \begin{vmatrix} u_{22} & u_{23} & u_{24} \\ 0 & u_{33} & u_{34} \\ 0 & 0 & u_{44} \end{vmatrix} = u_{11}u_{22} \begin{vmatrix} u_{33} & u_{34} \\ 0 & u_{44} \end{vmatrix} = u_{11}u_{22}u_{33}u_{44}. \tag{67}$$

A determinant of an upper (or lower) triangular matrix is just the product of all the diagonals. What about the co-factors of U? Strike out the row, and column for any of u_ij above the diagonal and you still have an upper triangular matrix, but it will have a column column of zeroes. All these $N-1$ sub-determinants are zero. We get non zero sub-determinants for the diagonal or below. The matrix of cofactors of an upper triangular matrix is lower triangular.

Remember, we use the transpose of the matrix of cofactors (with the chequer board pattern of plus and minus signs) to get the inverse, so the inverse of an upper triangular matrix is also an upper triangular matrix. On taking the transpose, all those non zero sub-determinants in the lower half end up in the upper half. Similarly the inverse of a lower triangular matrix is lower triangular. We have ended up with something of the form $Ux = L^{-1}b$ (We have called the lower triangular matrix on the r.h.s L^{-1} for a reason.) If we premultiply by L, we have $LUx = b$ or $x = (LU)^{-1}b = U^{-1}L{-}1 = A^{-1}b$.

We have factorised the square matrix A into a product of a lower and an upper triangular matrix. This LU factorisation is a workhorse in linear algebra. If we actually want L and U rather than $A^{-1} = U^{-1}L^{-1}$ we just invert L^{-1} to get L rather than inverting U, and premultiply both sides rather than by L to get $(LU)x = Ax = b$

We can store L^{-1} as we proceed with Gaussian elimination. In most computer code, both L and U will gradually overwrite the original matrix. These can be very large systems indeed, and we might be exceed our RAM capacity if we store both A. and a separate matrix containing L and U. The routines to do this "know" that the diagonal of the U matrix are all unity so the diagonal will just be the upper diagonal of L^{-1}. Extra workspace vectors are also needed to keep track of things. So, we have L^{-1} already.

Bearing in mind that the inverse of an upper/lower triangular triangular matrix is also upper/lower triangular, take a look at the 4×4 system below.

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ 0 & m_{22} & m_{23} & m_{24} \\ 0 & 0 & m_{33} & m_{34} \\ 0 & 0 & 0 & m_{44} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{68}$$

Here the m_{ij} are just unknown coefficients of some matrix M. If we multiply out MU we see that $m_{11}u_{11} = 1$, so we have the value of m_{11}. Then we have $m_{11}u_{12} + m_{12}u_{12} = 0$. So we now have m_{12}. We get the first row of M by back substitution just as we got the values of x in Gaussian elimination. We also see, $m_{22}u_{22} = 1$, and so on. Of course if any diagonal element of U is zero then its determinant is zero. In this case we know the determinant=1 since the diagonals of U are all 1.

We shall now look back to the topic of pivoting. (For convenence, we drop the steps where we divide through by the first non zero element in a row — we only want to look at the structure.) We shall start by trying eliminating by row, and pivoting by row swap. As we proceed with Gaussian elimination, we get

$$[M_{r(N-1)}P_{N-1} \ldots M_{r2}P_2Mr1P_1A]x = M_{r(N-1)}P_{N-1} \ldots M_{r2}P_2M_{r1}P_1b. \tag{69}$$

(It is $N-1$ because we subtract the row $N-1$ times in an $N \times N$ system.) The product $[M_{r(N-1)}P_{N-1} \ldots M_{r2}P_2M_{r1}P_1A]$ is upper triangular . If no permutation is needed fo a particular M_r, then the corresponding permutation matrix is the identity (or null permutation if you like). The product $[M_{r(N-1)}P_{N-1} \ldots M_{r2}P_2M_{r1}P_1]$ doesn't seem to be a lower trianular matrix because of the permutation matrices interspersed between the atomic matrices. Next we see that despite all these permutation matrices the matrix pre-multiplying b *actually does turn out to be lower triangular*. We demonstrate why this is next.

In the following, we consider a 4×4 system, and we remember that each permutation matrix is its own inverse. We shall eliminate by row and use row swaps. On the right hand side we end up with the product

$$M_3P_3M_2P_2M_1P_1 = M_3P_3M_2[P_3P_3]P_2M_1[P_2P_2]P_1$$

$$= M_3[P_3M_2P_3][P_3P_2M_1P_2P_3]P_3P_2P_1 = M_3[P_3M_2P_3][P_3P_2M_1P_2P_3]P_3P_2P_1 \qquad (70)$$

We shall come analise things like BAB^{-1} in the next chapter where we look at *similarity transforms* amongst other things. For the moment, take a look at the possibilites for P_1, P_2 and P_3. For P_1 we have,

$$P_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{or} \quad P_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{or} \quad P_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \qquad (71)$$

For P_2 we have,

$$P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{or} \quad P_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \qquad (72)$$

And for P_3 we have only one possibility. Namely

$$P_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \qquad (73)$$

Now lets look at term $P_2M_1P_2 = P_2M_1P_2^{-1}$ If we first swap row 2 and 3 and then swap colmun 2 and 3,

$$M_1 \to P_2M_1 \to P_2M_1P_2$$

becomes

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}. \qquad (74)$$

In the above, in terms like $P_3P_2M_1P_2P_3]$ the subscripts on the Ps are *all* greater than that on the M. This means that whatever happens, the situation is always akin to Fig.1. This means eqn.70 is just

$$M_3P_3M_2P_2M_1P_1 = M_3[P_3M_2P_3][P_3P_2M_1P_2P_3]P_3P_2P_1 = M_3M_2M_1P_3P_2P_1. \qquad (75)$$

Now the r.h.s of eqn.69 is a lower triangular matrix (L^{-1}) which is post multiplied by a permutation matrix $P = P_3P_2P_1$ and we still have our inverse matrix (once we put the divisions back in) from the decomposition in the form
$Ux = L^{-1}Pb$, or $[P^{-1}LU]x = b$. Of course $(P_{N-1} \dots P_2P_1)^-1$ is the same as
$(P_1^{-1}P_2^{-1} \dots P_{N-1}^{-1})$ which is just $P_1P_2 \dots P_{N-1}$.

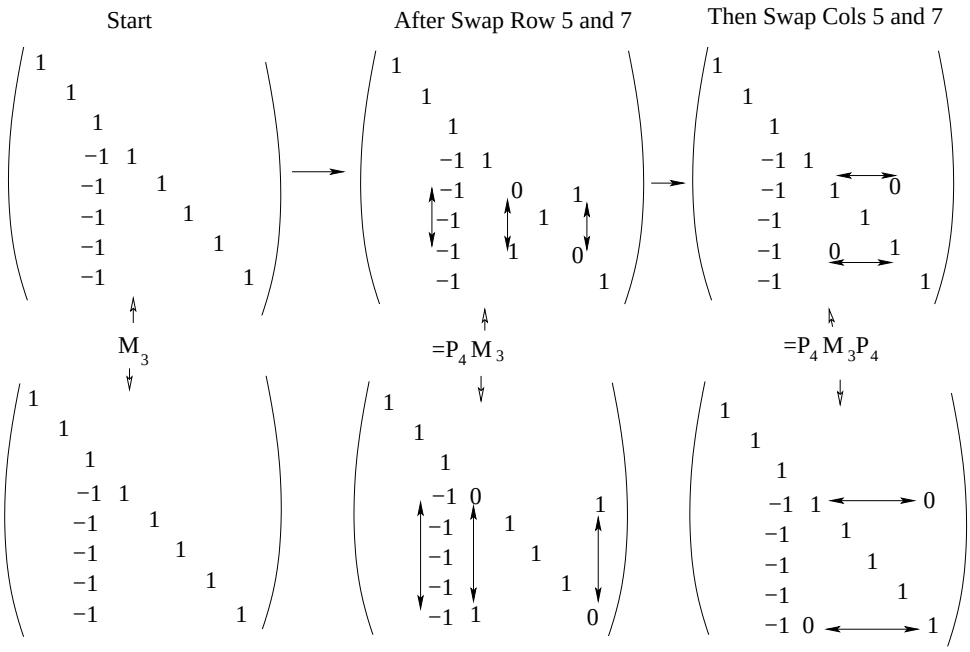

Figure 1: Since the swapping of rows in M_3 in P_4M_3 can only affect row numbers greater than row 3, when we swap the same column numbers after (as in $P_4M_3P_4$) we get the original M_3.

6.1 Variations on a Theme

We have taken a look at eliminating row-wise and swapping rows, but of course we could try other variations, such as eliminating row-wise but swapping columns. Other variations might be involve eliminating row-wise after swapping a row *or a column* depending whether the largest element in the column below the diagonal element is larger than the largest element to the right. We could also try eliminating row-wise.

We first look at eliminating by row and swapping columns instead. Then we would have got

$$M_{r1}Ax = M_{r1}AP_1P_1x = [M_{r1}AP_1]P_1x = M_{r1}b,$$

$$M_{r2}[M_{r1}AP_1]P_1x = M_{r2}[M_{r1}AP_1]P_2P_2P_1x = [M_{r2}M_{r1}AP_1P_2]P_2P_1x = M_{r2}M_{r1}b,$$

$$[M_{r(N-1)}\dots M_{r2}M_{r1}AP_1P_2\dots P_{N-1}][P_{N-1)}\dots P_3P_2P_1]x = M_{r(N-1)}\dots M_{r2}M_{r1}b. \quad (76)$$

This time we get the form $UPx = L^{-1}b$ or $A = LUP$.

What about eliminating column-wise? Well we could try looking at *post-multiplication* by atomic matricesM_c that will let us eliminate column-wise from left to right. A matrix

that will thus subract column 1 from the rest in a 4×4 system is

$$M_{c1} = \begin{pmatrix} 1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{with} \quad M_{c1}^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{77}$$

The reader will see that without pivoting this will yield $Ax = [AM_{c1}]M_{c1}^{-1}x = b$ where L is lower triangular. As we proceed, we shall arrive at $[AM_{c1}M_{c2}\ldots][M_{c(N-1)}^{-1}\ldots M_{c2}^{-1}M_{c1}^{-1}]x = b$. The reader can verify that (when we put the divisions back in) that $[AM_{c1}M_{c2}\ldots]$ will be lower triangular. Obviously $[M_{c(N-1)}^{-1}\ldots M_{c2}^{-1}M_{c1}^{-1}]$ is upper triangular and we have the familiar the form $Ax = LUx = b$.

If we pivot by swapping columns in this proceedure, the r.h.s stays the same, while on the left hand side

$$Ax \to [AP_1M_{c1}][M_{c1}^{-1}P_1]x \to [AP_1M_{c1}P_2M_{c2}][M_{c2}^{-1}P_2M_{c1}^{-1}P_1]x$$

and eventally

$$\to [AP_1M_{c1}P_2M_{c2}P_3M_{c3}\ldots P_{N-1}M_{c(N-1)}^{-1}[M_{c(N-1)}P_{N-1}\ldots M_{c3}^{-1}P_3M_{c2}^{-1}P_2M_{c1}^{-1}P_1]x$$

or

$$U[M_{c(N-1)}\ldots M_{c3}^{-1}P_3M_{c2}^{-1}P_2M_{c1}^{-1}P_1]x = b. \tag{78}$$

In the following we note that if $\alpha > \beta$, $\alpha > \gamma$, and $\alpha > \delta$, then

$$P_\alpha M_\beta M_\gamma M_\delta P_\alpha = P_\alpha M_\beta P_\alpha P_\alpha M_\gamma P_\alpha P_\alpha M_\delta P_\alpha = M_\beta M_\gamma M_\delta. \tag{79}$$

We want to consider

$$[M_{c(N-1)}P_{N-1}\ldots M_{c3}^{-1}P_3M_{c2}^{-1}P_2M_{c1}^{-1}P_1]. \tag{80}$$

Lets just put $M_{cj}^{-1} \to M_j$, and put $N - 1$=4.

$$= [M_4P_4M_3P_3M_2P_2M_1P_1]$$

$$= [M_4P_4M_3P_3M_2[P_2M_1P_2]P_2P_1] = [M_4P_4M_3[P_3M_2M_1P_3]P_3P_2P_1]$$

and by eqn.80

$$= [M_4P_4M_3M_2M_1P_3P_2P_1] = [M_4[P_4M_3P_4]P_4M_2M_1P_3P_2P_1]$$

$$= [M_4M_3M_2M_1][P_4P_4P_3P_2P_1]. \tag{81}$$

The reader can verify that if we eliminate row-wise *bottom up* or column-wise (right to left) we end up with a UL. If we don't bother with total pivoting, we might want to mix eliminating row-wise (top down) and swapping either a row or a column (depending which gives us the new diagonal element that is largest in magnitude) might be a slight improvment, but generally most routines will just eliminate row-wise (top down) and swap rows.

6.2 The Axiomatic Definition of a Vector Space

In this chapter, we have defined operations such as subtracting rows or columns of a matrix, or making a new row/column which is a linear combination of other rows or columns. We shall list the axiomatic *definition* of a vector space below. The reader can see, that when the operations that we have defined in solving linear equations and LU decomposition, that the rows/columns of a matrix A are indeed vectors. Also, considering $Ax = b$ and $Ay = c$, we can see that $A(x + y) = (b + c)$ that x, y, b and c are all column vectors. Also if we consider $x^T A^t = b^t$, we can see that x^T is a row vector.

We shall suppose we have a set of entities that are members of the set $\mathcal{V}$. Then $\mathcal{V}$ is a vector space if the following axioms hold.

Axiom 1 $\forall\, a, b \in \mathcal{V} \,\exists\, \text{“}+\text{”}\ s.t.\ a + b = c \in \mathcal{V}.$

Axiom 2 $If\ a, b \in \mathcal{V},\ a + b = b + c.$

Axiom 3 $\exists\, 0 \in \mathcal{V}\ s.t.\ a + 0 = a.$

Axiom 4 $\forall\, a \in \mathcal{V} \,\exists\, -a \in \mathcal{V}\ s.t.\ a + (-a) = 0.$

Axiom 5 $\forall\, \alpha, \beta \in \mathbb{R},\ a \in \mathcal{V},\ \exists\, \alpha a,\ \beta a \in \mathcal{V}.$

Axiom 6 $\forall\, \alpha, \beta \in \mathbb{R},\ a \in \mathcal{V},\ \exists\, \alpha a,\ \beta a \in \mathcal{V}.$

Axiom 7 $\alpha(\beta a) = ((\alpha\beta)a) = ((\beta\alpha)a = \beta(\alpha a).$

Axiom 8 $\alpha(a + b) = \alpha a + \alpha b.$

Axiom 9 $(\alpha + \beta)a = \alpha a + \beta a.$

If 1 is the unit element of $\mathbb{R}$

Axiom 10 $1a = a.$

All this looks complicated at first glance, but remembering that $\exists$ means “there exists”, $\forall$ means “for all” and $\in$ means “in the set”, it is easy to see that the normal geometrical vectors satisfy all these axioms. Note that there is nothing stated here about magnitude or direction.

7 A Brief Aside on Invariants and Tensors

We start off by just briefly outline what is meant by a tensor from a physicists point of view. Any real physical quantity must be completely independent of any coordinate system or system of measurement that describes it. If my bath is dangerously hot, it doesn't matter if I am told its temperature in Fahrenheit or Celsius or Kelvin. As long I am told the units that the temperature is measured in as well, I know whether it's too hot or otherwise.

If I am told it is so-and-so in Fahrenheit, but I don't have a feel for those units I can transform to Celsius and understand the nature of the temperature. A temperature is an example of a *scalar* and scalars can be represented by a single number. Though different systems of measurement might be used, that number doesn't change if I transform spatial coordinates.

Now, if you tell me that I must go North I could reply with the question "Is that magnetic North or true North?" A direction or a relative position are both examples of a vector: it can be described by a set of numbers and given directions, but the set of numbers depends on those given directions. So true North and magnetic North are examples of two different given directions. To transform some vector between magnetic and true North coordinates, I need a rotation matrix. Apart from simple displacements, there will be a matrix transformation between *any* possible systems of describing the physical direction. Later we shall see that a vector is a tensor of rank one.

In the last example (a vector) there is something physical called the direction. This can be described by a set of numbers (the elements of a vector) but these elements are not at all physical. They depend on the coordinate system. However, we can transform from *any* coordinate system to *any* other coordinate system. When we do so, we see that there are things that are physically real such as a distance, or the angle between one direction and another. These distances and angles are called *invariants*. They must be *physically the same* in all coordinate systems describing the same thing in different ways. Because we can can look at the same object from the point of view of *all possible coordinate systems* we can see what depends on the coordinate system and what does not.

Of course a rotation is the same no matter how you describe the coordinate systems. If you turn 90° it doesn't matter what the coordinate system is. You turn 90° in all possible (non-rotating) coordinate systems. A rotation can be described by a matrix, and if any physical thing (such as a the physical rotation of a body) can be described by a matrix, the matrix *represents* a second rank tensor. The thing about tensors is that they are defined by the way their components transform from one coordinate system to another. We shall see how tensors of rank 1 and 2 transform later.

In physics, we can have rotating coordinates. Famously, in doing so, we have to bring in

a fictitious Coriolis force [4] to keep the physics right. If a tensor, of any rank, is zero in one coordinate system, it is zero in all coordinate systems. The Coriolis force is only non zero in rotating coordinates. It just dissappears if we transform back to a non-rotating reference frame. So the Coriolis force doesn't count as a vector, even though it looks like one when written down. From a physicists point of view, if we can find a coordinate system where something disappears, that something is not real: it is just an artefact of the reference frame. We call the simplest reference frames (where everything that can possibly vanish has done so) *inertial* reference frames. In these frames, the *description* of the physics is the simplest possible.

Given this, you might argue that velocity is not a vector because you can always transform to a moving coordinate system where the velocity is zero. This is in fact so, *absolute* velocities do not exist: there is no such thing as an absolute velocity vector. Relative velocities do exist. I shall suppose I stand at the origin of my own coordinate system, velocities are relative to the origin of my coordinate system (me) which has velocity zero. Suppose I have a body P which is moving at constant velocity in my own coordinate system (which is an inertial frame). From the point of view of someone in a moving coordinate system, the *components* of the velocity of P may even be zero, but the components of my velocity in this moving system are now non zero. The relative velocity between myself and the body P is the same in the moving system as in my own coordinate system. Relative velocities are invariant, and are thus vectors.

The point of this aside? Just because something can be written down as a 1×2 or 1×3 array of numbers doesn't mean that that something is a vector. Just because something can be written down as a 2×2 or 3×3 array of numbers doesn't mean that that something is a second rank tensor.

However, in many cases we shall speak of a matrix being a tensor regardless, just as we have called $(x, y)^T$ and $(b_1, b_2)^T$ vectors in the first two chapters of this volume. (I have used the transpose symbol T to convert from row to column vectors.)

Suppose we have an N dimensional *Euclidean space*. (An excellent model of the ordinary everyday space we live in if the number of dimensions is three [5].) If the our vector space is to describe a Eucldean space then we must define an inner product on our vector space. Namely, we define $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos \psi$. By $|\mathbf{a}|$, we mean the "length" of $\mathbf{a}$, and we mean that ψ is the anglular difference between the two "directions" $\mathbf{a}$ and $\mathbf{b}$.

Suppose also we have a set of N independent vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3 \ldots$ which can describe the position of *any* point in this space. If they can, then none of these vectors $\mathbf{a}$ can be written as a linear combination of the others. We suppose that all the information we have

[4] We discuss Coriolis forces in some detail in the section on Galilean Relativity in Vol.5 of this series.

[5] Roger Penrose emphasises that Euclidean geometry is a branch of theoretical physics as well as part of pure mathematics in his popular book "The Emperors New Mind".

is the angles ψ_{ij} between each vector $\mathbf{a}_i$ and $\mathbf{a}_j$ for each i and j from 1 to N, and the lengths of each $\mathbf{a}_i$. Then we have the dot products between them which we denote g_{ij}. That is $g_{ij} = \mathbf{a}_i \cdot \mathbf{a}_j$ which is the product of the two lengths and the cosine of the angle between $\mathbf{a}_i$ and $\mathbf{a}_j$. This g_{ij} is known as the *metric tensor*. Note, the metric tensor is by definition *symmetric* (which means $g_{ij} = g_{ji}$).

We have no idea whether or not any of the **a**s are at right angles to each other, and we have no idea whether any vector has the same length as any other. However we do take it for granted that none are zero in length, and none is a linear combination of others. We now suppose we have another set of vectors **b**. Given a common origin, then any vector **r** can be represented using either the set **a** or **b**. That is we can write the *same* vector as

$$\mathbf{r} = \alpha_1\mathbf{a}_1 + \alpha_2\mathbf{a}_2 + \alpha_3\mathbf{a}_3 + \ldots,$$

$$\mathbf{r} = \beta_1\mathbf{b}_1 + \beta_2\mathbf{b}_2 + \beta_3\mathbf{b}_3 + \ldots. \tag{82}$$

Of course, any of the **a**s can be written in terms of the **b**s. We have sets of numbers γ such that

$$\mathbf{a}_1 = \gamma_1^1\mathbf{b}_1 + \gamma_1^2\mathbf{b}_2 + \gamma_1^3\mathbf{b}_3 + \ldots,$$
$$\mathbf{a}_2 = \gamma_2^1\mathbf{b}_1 + \gamma_2^2\mathbf{b}_2 + \gamma_2^3\mathbf{b}_3 + \ldots, \tag{83}$$

and so on. (The superscripts are just labels as are the subscripts, they do not indicate powers). In the light of this we can rewrite **r** as

$$\mathbf{r} = \alpha_1\gamma_1^1\mathbf{b}_1 + \alpha_2\gamma_2^1\mathbf{b}_1 + \alpha_3\gamma_3^1\mathbf{b}_1 + \ldots$$

$$+\alpha_1\gamma_1^2\mathbf{b}_2 + \alpha_2\gamma_2^2\mathbf{b}_2 + \alpha_3\gamma_3^2\mathbf{b}_2 + \ldots. \tag{84}$$

We can write this in matrix form as

$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} \gamma_1^1 & \gamma_2^1 & \gamma_3^1 & \ldots \\ \gamma_1^2 & \gamma_2^2 & \gamma_3^2 & \ldots \\ \gamma_1^3 & \gamma_2^3 & \gamma_3^3 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \end{pmatrix}. \tag{85}$$

Given a vector with components α_i according to the observer using the basis vectors **a**, we see that according to an observer using the basis vectors **b**, that the same vector has components β_i. The β_i are related to the α_i via $\beta_i = \gamma_j^i\alpha_j$ which we can write down in matrix form. So the superscript i in γ_j^i indicates a row and the subscript j indicates a column. We shall also write this as $\beta = \Gamma\alpha$ and as $\alpha = \Gamma^{-1}\beta$. If the determinant of Γ is zero, we do not have independent vectors as a basis set. The transformation is *passive* and the meaning of the vector **r** is unchanged by the transformation of coordinates. Under this transformation, the vector **r** is an invariant. No angles or lengths have changed.

Now suppose we have an *active* transformation. We operate on **r** and map it to $\mathbf{r}'$ so that $\mathbf{r}'$ is a scaled, reflected, and rotated version of **r**. Its coordinates change in both coordinate

systems. We shall write this in terms of a matrix A. We shall put $\alpha' = A\alpha$ where α' are the components of the vector $\mathbf{r}'$ which is different to the vector $\mathbf{r}$ with the original components α. Now we want to know two things. What are the components of β' that correspond to the same vector as α'? What are the elements of a matrix, that according to the **b** reference frame, has exactly the same meaning as a matrix with elements a_{ij} in the **a** reference frame.

In the reference frame that uses the **a**s as basis vectors, the tensor that actively maps $\mathbf{r} \to \mathbf{r}'$ is denoted A. It shall be a tensor because the same vector $\mathbf{r}$ is mapped to the same vector $\mathbf{r}'$ regardless of the details of the coordinate system. According to the **a** coordinate system, the elements of the matrix that describe A are to be denoted by a_{ij}. In the reference frame that uses the **b**s as basis vectors, the matrix transformation that maps $\mathbf{r} \to \mathbf{r}'$ is also A, but according to the **b** coordinate system the matrix elements of A are a'_{ij}.

Clearly $\beta' = \Gamma\alpha'$, or $\beta' = \Gamma A\alpha$, but $\alpha = \Gamma^{-1}\beta$. This in turn gives us $\beta' = [\Gamma A\Gamma^{-1}]\beta$.

The second rank tensor that transforms (or maps) $\mathbf{r} \to \mathbf{r}'$ is $A = \Gamma A\Gamma^{-1}$. This type of transform (that leaves the actual tensor unchanged) is called a *similarity transform.* The meaning of the tensor is unchanged, but in terms of the matrix elements that represent A, we have $a'_{ij} = \gamma^i_k a_{kl}[\gamma^{-1}]^l_j$. (Naturally we are using the Einstein summation convention.) By $[\gamma^{-1}]^l_j$ we mean the element in the lth row and jth column in the matrix representation of the inverse of Γ or Γ^{-1}. If we write this out in matrix form, we get

$$\begin{pmatrix} a'_{11} & a'_{12} & a'_{13} & \dots \\ a'_{21} & a'_{22} & a'_{23} & \dots \\ a'_{31} & a'_{32} & a'_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} = \begin{pmatrix} \gamma^1_1 & \gamma^1_2 & \gamma^1_3 & \dots \\ \gamma^2_1 & \gamma^2_2 & \gamma^2_3 & \dots \\ \gamma^3_1 & \gamma^3_2 & \gamma^3_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots \\ a_{21} & a_{22} & a_{23} & \dots \\ a_{31} & a_{32} & a_{33} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} \gamma^1_1 & \gamma^1_2 & \gamma^1_3 & \dots \\ \gamma^2_1 & \gamma^2_2 & \gamma^2_3 & \dots \\ \gamma^3_1 & \gamma^3_2 & \gamma^3_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}^{-1} . \tag{86}$$

We can pre-multiply both sides by Γ^{-1}, and then post-multiply both sides by Γ to get $\Gamma^{-1}A\Gamma = A$.

Given this transformation rule the operator A has exactly the same properties in either of the coordinate systems. Indeed, we may transform between an arbitrary set of coordinate systems and the meaning of A will be the same regardless of the details of any of them. Just as with a vector, the entries in any row or column depend on the coordinate system, while the properties of A do not. As such the object A is a *second rank tensor* It is second rank because it needs two indices to represent its elements, and so a vector is a first rank tensor as we only need one index. Some tensors have the same elements in any coordinate system. It is obvious that according to eqns.86, if A has elements δ_{ij} (Kronecker δ) in one coordinate system, it will have elements δ_{ij} in all coordinate systems.

We recall that the determinant of a matrix product is the product of the determinants. On examining eqn.86, and realising that $Det\Gamma^{-1} = 1/Det\Gamma$ we see that Det$A$ is the same regardless of the coordinate system (since the determinant of a product is the product of the determinants) and is *an invariant of A.* (We recall from 3 that the detrminant can be

thought of as a volume of an N dimensional parallelipiped.)

Now, let's revisit eqn.83. Suppose we rewrite this and denote the ith vector as $\mathbf{a}_i = \gamma_i^k \mathbf{b}_k$ where we imply a sum over k. We can take the dot product with $\mathbf{a}_j$ and put $\mathbf{a}_i \cdot \mathbf{a}_j = g_{ij}$ and $\mathbf{b}_i \cdot \mathbf{b}_j = g'_{ij}$. If we write this all out in full, we end up with

$$g_{ij} = \mathbf{a}_i \cdot \mathbf{a}_j = \gamma_i^k \gamma_j^l \mathbf{b}_k \cdot \mathbf{b}_l = \gamma_i^k \gamma_j^l g'_{kl}. \tag{87}$$

This is how the elements of the metric tensor transforms from one system to another. There is an important thing to note here. There is no Γ^{-1} in eqn.87. The metric tensor transforms *in a different manner* to the the active transformation tensor A which we examined earlier. *There are different types of tensor transformation.* We shall examine this more closely in the next section.

Now we shall examine a particular type of system. Suppose our sets of vectors $\mathbf{a}$ and $\mathbf{b}$ are all the same length (unity), and that in both sets of vectors, the basis vectors are mutually perpendicular. Then in both systems we see that the metric tensor is just δ_{ij}. That is eqn.87 would read $\delta_{ij} = \gamma_i^k \gamma_j^l \delta_{kl} = \gamma_i^k \gamma_j^k$. (Only the elements such that $l \neq k$ are non zero.) Written in matrix form this would look like $\Gamma\Gamma^T = I$. We call any matrix such that $\Gamma^{-1} = \Gamma^T$ an *orthogonal matrix*. (Since all the column (and row) vectors of the matrix are of unit length, we might want to call the matrix *orthonormal* instead — but in this context the term orthogonal is standard.)

We have seen enough on transformations to understand the what is going on when looking at eigenvalues and eigenvectors in chapter 9, but bofoer continuing we give a brief intoduction to differentiating a function of more than one variable.

8 Covariant and Contravariant Vectors.

As the title of this section suggests, there are in fact two distinct types of vectors. This distinction arises when we need to look at invariance when the transformation between two systems of coordinates is *not* orthogonal.

We saw in the last section that the active tensor A transforms differently from the metric tensor. They are different kinds of tensors. In this section we shall encounter two different kinds of transformations for vectors. This means we shall have to overhaul our notation to take this into account.

The position vectors that we have seen so far are called *contravariant vectors*. For the rest of this section, we always use a superscript for the elements of a contravariant vector. (That is β^2 is not to be confused with "beta squared", it is just "beta two" — the second element in a contravariant vector.) Only the elements of *covariant vectors* will have subscripts. We can recast eqn.85 in terms of partial derivatives

To use partial derivatives in this context may look like an over-complication, but it is the first step towards a tensor calculus where the metric may vary with position, and so we use this notation. Since (in this chapter) the metric is *not* a function of position, all the partial derivatives are just constants.

Looking at the ith row in eqn.85,

$$\beta^i = \gamma^i_1\alpha^1 + \gamma^i_2\alpha^2 + \gamma^i_3\alpha^3 + \gamma^i_4\alpha^4 + \dots$$

so

$$\frac{\partial\beta^i}{\partial\alpha^1} = \gamma^i_1, \;\; \frac{\partial\beta^i}{\partial\alpha^2} = \gamma^i_2, \;\; \frac{\partial\beta^i}{\partial\alpha^3} = \gamma^i_3, \;\; \frac{\partial\beta^i}{\partial\alpha^4} = \gamma^i_4, \dots \tag{88}$$

(Recall that in differentiating w.r.t. coordinate α^4 we regard $\alpha^1, \alpha^2, \alpha^3, \alpha^5, \dots$ as constants). We have changed over to the new notation. We know that position vectors are contravariant vectors, and so we now use superscripts rather than subcripts. We see that

$$\gamma^i_j = \frac{\partial\beta^i}{\partial\alpha^j}. \tag{89}$$

We can now write eqn.85 as

$$\beta^i = \frac{\partial\beta^i}{\partial\alpha^j}\alpha^j, \tag{90}$$

also

$$\alpha^i = \frac{\partial\alpha^i}{\partial\beta^j}\beta^j. \tag{91}$$

These partial derivatives when written in matrix form are called the *Jacobian* of the transformation. Also, the determinant of this matrix is called the Jacobian determinant and is often denoted $|J|$. Instead of writing $\beta = \Gamma\alpha$ we also write $\beta = J\alpha$. Recall the active matrix A. In the light of what we have said, the matix elements transform as

$$[a']^i_j = \frac{\partial\beta^i}{\partial\alpha^k}\frac{\partial\alpha^l}{\partial\beta_j}a^k_l = \gamma^i_k[\gamma^{-1}]^l_j a^k_l. \tag{92}$$

Eqn.92 an inverse of the matrix representing Γ as in eqn.90. Note how that in the summation, the subscript k in γ^i_k is matched by a superscript k in a^k_l. Also the superscript l in $[\gamma^{-1}]^l_j$ is matched by a subscript l in a^k_l. The free index i appears as a superscript on both sides, and the free index j appears as a subscript on either side. This must always be the case. Whenever the same sub/superscript occurs twice, the implied summation has a superscript and a subscript. Also, if if i is a free index on the l.h.s., not only must it appear once and only once on the r.h.s., but it must be a super/subscript on both sides too. If, after any manipulation, any of these rules are broken, the manipulation has gone horribly wrong!

Suppose we have some position vector $\mathbf{r}$ with components α^i: what is its length? In just two dimensions, we can write it down in the following form

$$|\mathbf{r}|^2 = \begin{pmatrix}\alpha^1, & \alpha^2\end{pmatrix}\begin{pmatrix}g_{11} & g_{12}\\ g_{21} & g_{22}\end{pmatrix}\begin{pmatrix}\alpha^1\\ \alpha^2\end{pmatrix}. \tag{93}$$

Now we shall introduce another vector with coefficients α_i such that

$$\begin{aligned}\begin{pmatrix}\alpha_1, & \alpha_2\end{pmatrix} &= \left[\begin{pmatrix}g_{11} & g_{12}\\ g_{21} & g_{22}\end{pmatrix}\begin{pmatrix}\alpha^1\\ \alpha^2\end{pmatrix}\right]^T \\ &= \begin{pmatrix}\alpha^1 & \alpha^2\end{pmatrix}\begin{pmatrix}g_{11} & g_{12}\\ g_{21} & g_{22}\end{pmatrix}. \end{aligned} \tag{94}$$

We can do this because because the metric is (by definition) symmetric. Then we can write $|\mathbf{r}|^2 = \alpha_i\alpha^i$. The reader might think this is somewhat artificial, however we shall soon see that it is not just notational convenience, but we shall find that vectors like this occur quite naturally.

The vector with coefficients α_i is *by definition a covariant* vector. That is, for each contravariant vector with coefficients α^i there is a covariant vector with components $\alpha_i = g_{ij}\alpha^j$.

The components α_i does not transform to another coordinate system in the same way as the contravariant position vectors of eqn.90. In the last section we saw that the metric tensor doesn't transform in the same way as our active transformation A in eqn.86 and 92. We shall revisit this now.

$$|\mathbf{r}|^2 = g_{ij}\alpha^i\alpha^j = g'_{ij}\beta^i\beta^j = g'_{ij}\frac{\partial\beta^i}{\partial\alpha^k}\frac{\partial\beta^j}{\partial\alpha^l}\alpha^k\alpha^l = g'_{kl}\frac{\partial\beta^k}{\partial\alpha^i}\frac{\partial\beta^l}{\partial\alpha^j}\alpha^i\alpha^j,$$

so

$$g'_{ij} = \frac{\partial\alpha^k}{\partial\beta^i}\frac{\partial\alpha^l}{\partial\beta^j}g_{kl}. \tag{95}$$

The metric tensor's transformation involves two inverses of the matrix representing Γ as in eqn.90, and the metric tensor is completely covariant.

Let's examine $g'_{ij}\beta^j$ we get a covariant vector β_i,

$$\beta_i = g'_{ij}\beta^j = \frac{\partial\alpha^k}{\partial\beta^i}\frac{\partial\alpha^l}{\partial\beta^j}\frac{\partial\beta^j}{\partial\alpha^m}g_{kl}\alpha^m = \frac{\partial\alpha^k}{\partial\beta^i}\delta^l_m g_{kl}\alpha^m = \frac{\partial\alpha^k}{\partial\beta^i}g_{km}\alpha^m,$$

but

$$g_{km}\alpha^m = \alpha_k, \text{ and so } \beta_i = \frac{\partial\alpha^k}{\partial\beta^i}\alpha_k, \tag{96}$$

is the rule for transforming a covariant vector.

So, eqns.90 and 96 give us two different types of vector transformation. We have already seen two types of second rank tensor transformation. We shall now see that this can be generalised to all manner of tensors. Suppose we have some object $X^{\alpha\beta\gamma\ldots}_{ijk\ldots}$ with P superscripts and Q subscripts. If (and only if) it transforms to another coordinate system with P transformation matrices of the form seen in eqn.90 and Q transformation matrices of the form of eqn.96 then X is an Nth rank tensor ($N = P + Q$). If P or Q is zero then it is a covariant or a contravariant tensor: otherwise it is called a mixed tensor. In eqn.92,

the active transformation A can be seen to transform as a mixed second rank tensor and in eqn.95 the metric is seen as a completely covariant second rank tensor.

Vectors (rank 1 tensors) can be combined with other vectors to form tensors as we have seen already in constructing the metric. (In fact the vector cross product of a vector $\mathbf{x}$ and $\mathbf{y}$ in 3D is actually a skew symmetric second rank tensor formed by $C^{ij} = x^i y^j - x^j y^i$ — it can be written as a vector since in 3D there are only three distinct components.) Similarly we can form outer products of two second rank tensors, for instance given A^i_j and B^i_j we can form $C^{ik}_{jl} = A^i_j B^k_l$. Similarly we may form lower rank tensors from higher rank tensors by *contraction* — that is we set two indices to be the same letter and do the summation that this implies. Hence the scalar dot product of two vectors x^i and y_j can be seen as the contraction of the tensor formed by $x^i y_j$.

We may form a covariant vector from a contravariant vector by multiplying by the metric tensor. Conversely, we can form a contravariant vector from a covariant vector by multiplying by the inverse of the metric tensor ($\alpha^i = g^{ij}\alpha_j$) with $g^{ik}g_{kj} = \delta^i_j$. If we have two contravariant vectors with components ρ^i and σ^i, then the dot product is $g_{ik}\rho^k\sigma^k = \rho_i\sigma^i = \rho^i\sigma_i$ is invariant under any transformation of the coordinates. We mentioned earlier that all this looks rather artificial. We shall now see why it is a lot more than a notational convenience (or annoyance). It is in fact, quite essential from the point of view of physics.

Suppose we have some scalar field $\phi(\alpha^1, \alpha^2, \ldots)$ and form the *gradient* of ϕ which is written as $\nabla\phi$. This vector field is defined as

$$\nabla\phi = \frac{\partial\phi}{\partial\alpha^1}\mathbf{a_1} + \frac{\partial\phi}{\partial\alpha^2}\mathbf{a_2} + \ldots \tag{97}$$

If ϕ is a potential energy for instance, the force $\mathbf{F}$ is given by $\mathbf{F} = -\nabla\phi$. Another example is that if ϕ is a temperature, $\mathbf{J} = -K\nabla\phi$ is an energy flux vector. We see that ϕ is a scalar field, and therefore must have exactly the same physical meaning in both coordinate systems.

In more compact notation,

$$\nabla\phi_i = \frac{\partial\phi}{\partial\alpha^i}. \tag{98}$$

In another system of coordinates

$$\nabla\phi = \frac{\partial\phi}{\partial\beta^1}\mathbf{b_1} + \frac{\partial\phi}{\partial\beta^2}\mathbf{b_2} + \ldots \tag{99}$$

(We use a subscript for $\nabla\phi$ because, as we shall now see, the vector is covariant.) So we have the components of $\nabla\phi$ in the system using the $\mathbf{a}$ vectors. Given these components, what are the components of $\nabla\phi$ in the $\mathbf{b}$ system? The function of a function rule for partial differentiation tells us

$$\frac{\partial}{\partial\beta^i}\phi(\alpha^1(\beta^1, \beta^2, \ldots),\ \alpha^1(\beta^1, \beta^2, \ldots),\ \alpha^1(\beta^1, \beta^2, \ldots),\ \ldots)$$

$$= \frac{\partial \alpha^1}{\partial \beta^i}\frac{\partial \phi}{\partial \alpha^1} + \frac{\partial \alpha^2}{\partial \beta^i}\frac{\partial \phi}{\partial \alpha^2} + \frac{\partial \alpha^3}{\partial \beta^i}\frac{\partial \phi}{\partial \alpha^3} + \dots \tag{100}$$

Or in shorthand

$$\frac{\partial \phi}{\partial \beta^i} = \frac{\partial \alpha^j}{\partial \beta^i}\frac{\partial \phi}{\partial \alpha^j}, \quad \frac{\partial \phi}{\partial \alpha^i} = \frac{\partial \beta^j}{\partial \alpha^i}\frac{\partial \phi}{\partial \beta^j}. \tag{101}$$

Vectors such as $\nabla\phi$, do not transform like the usual contravariant position vectors, but transform as covariant vectors.

9 Eigenvalues and Eigenvectors

10 The Basic Problem

The problem to be addressed in this chapter is as follows. Given a matrix A we want to find vectors x such that

$$Ax = \lambda x, \tag{102}$$

where λ is just some number. If x satisfies eqn.102 then it is called an *eigenvector* of A, and the number λ is the *eigenvalue*. This innocent looking problem is of great importance for both applied and pure mathematics. A classic text on the subject is Wilkinson's "Algebraic Eigenvalue Problem" [5]. We see that this is equivalent to asking, "Are there special vectors x such that Ax is in the same direction as x?

The first thing we notice that we can use the identity I to write the problem as

$$(A - \lambda I)x = 0. \tag{103}$$

where $x \neq 0$. (In this context, 0 means a vector in which every element is zero.)

When can a non-zero matrix times a non-zero vector result in the zero vector? Consider this two by two system

$$\begin{aligned} ax + by &= 0 \\ cx + dy &= 0. \end{aligned} \tag{104}$$

We can write this as $Ax = 0$, or $x = A^{-1}0$. For a non-trivial solution ($(x, y) \neq (0, 0)$) to exist, then A must be singular. Eliminating x from eqn.104 immediately tells us that $(ad-bc)$ must be zero, so the matrix is singular as expected. If we can make the determinant zero by putting $d = bc/a$ we see that any vector (x, y) is mapped onto $(ax+by, (c/a)\times[ax+by]) = (0, 0)$. Any of the infinity of possible input vectors on the line $y = -ax/b$ is mapped to the origin. This is why the matrix has no inverse. The second equation in eqns.104 contains no new information and the system has no solution. In other words, non-trivial solutions of eqn.103 requires that $Det(A - \lambda I) = 0$. So, for our eigenvalue/eigenvector problem, we need to seek out values of λ that make this determinant vanish.

Now, given this, we write eqn.102 out and solve it for a particular 3×3 case.

$$\begin{pmatrix} 1 & 3 & 0 \\ 3 & -2 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{105}$$

Then $Det(A - \lambda I)$ gives us

$$\begin{vmatrix} 1-\lambda & 3 & 0 \\ 3 & -2-\lambda & -1 \\ 0 & -1 & 1-\lambda \end{vmatrix} = 0. \tag{106}$$

Next we write down the expansion of the determinant using the first row.

$$(1-\lambda)[(-2-\lambda)(1-\lambda)-1] - 3(1-\lambda) = 0.$$

$$(\lambda - 1)(\lambda - 3)(\lambda + 4) = 0. \tag{107}$$

We have three eigenvalues 1,3, and -4.

We obviously get a quadratic in λ for the 2×2 case. We shall always get a cubic for the 3×3 case and so on. This follows from the definition of a determinant.

For an $N \times N$ matrix there are N eigenvalues and N eigenvectors. To see this, we write our determinant as

$$Det(A - \lambda I) = 0 = \epsilon_{ijk\ldots}(a_{i1} - \lambda\delta_{i1})(a_{j2} - \lambda\delta_{j2})(a_{ik} - \lambda\delta_{ik})\ldots$$

$$Det(A - \lambda I) = 0 = \epsilon_{ijk\ldots}(a_{1i} - \lambda\delta_{1i})(a_{j2} - \lambda\delta_{j2})(a_{k3} - \lambda\delta_{k3})\ldots \tag{108}$$

Note that this means that A^T automatically has the same eigenvalues as A, since the a_{i1} of A^T are the a_{1i} of A. All this is saying is that if we expand $Det(A^T - \lambda I)$ by column 1, we get the same thing when we $Det(A - \lambda I)$ by row one and vice-versa.

We have arranged this particular 3×3 matrix so that the cubic is easy to factorise. Solving the general cubic can be quite tough. Now the Abel-Ruffini impossibility theorem (which we come across in §**??**) tells us that (in general) we can't solve the polynomial for $N > 4$. To be more precise, we mean that if $N > 4$, the general polynomial cannot be solved in terms of radicals [6] or in a finite number of steps — which in turn, means that iterative methods will be required. So for large systems, we can't solve an algebraic equation for the eigenvalues. Also, just because the entries in the matrix are real doesn't mean that the eigenvalues or the entries in the eigenvectors are real. For now, we just look at this 3×3 case.

Lets substitute the first eigenvalue, $\lambda = 1$ in the matrix of eqn.103. We get

$$\begin{pmatrix} 0 & 3 & 0 \\ 3 & -3 & -1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \lambda \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{109}$$

[6] By a "radical" we meen something involving a $\sqrt{}$, or a $\sqrt[3]{}$, and so on.

So we get $3x_1 - 3x_2 - x_3 = 0$, or $x_3 = 3x_1 - 3x_2$. But since we also have $x_2 = 0$, *any* vector in the direction $x_3 = 3x_1$ with $x_2 = 0$ is an eigenvector of this matrix. For instance (1,0,3) is an eigenvector. We shall always choose our eigenvectors to have a "length" of 1. So we choose $1/\sqrt{10} \times (1, 0, 3)$ to be our unit eigenvector. Our other unit eigenvectors are $1/\sqrt{14} \times (3, 2, -1)$ for $\lambda = 3$ and $1/\sqrt{35} \times (-3, 5, 1)$ for $\lambda = -4$.

The eigenvalues of a 2×2 are trivial to work out. Both the 3×3 and 4×4 cases have closed form solutions for the eigenvalues: no iterative proceedures are required. We present the analysis for solving cubic and quartic equations later in §**??**. For the moment we work out the coefficients for the cubic equation that give the eigenvalues of a 3×3 matrix, and the quartic equation that give the eigenvalues of a 4×4 matrix.

10.1 The Eigenvalues of a 3×3 Matrix

When doing pen and paper algebra, subscripts can easily get mixed up. So, we shall use individual letters for the matrix elements. We shall miss out the letters i, j, k, l, m, n, and o when naming matrix elements. Our matrix is

$$A = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & p \end{vmatrix}. \tag{110}$$

Then the equation for the eigenvalues if A is

$$\begin{vmatrix} a-\lambda & b & c \\ d & e-\lambda & f \\ g & h & p-\lambda \end{vmatrix} = 0. \tag{111}$$

We can quickly work out that this translates to

$$(a-\lambda)[(e-\lambda)(p-\lambda) - fh] - b[dp - d\lambda - fg] + c[dh - eg + g\lambda] = 0. \tag{112}$$

After a bit of tidying, the cubic equation we want is

$$\begin{aligned} \lambda^3 - \lambda^2(a+e+p) + \lambda(ap + ae + ep - fh - bd - cg] \\ -[a(ep - fh) - b(dp - fg) + c(dh - eg)] = 0. \end{aligned} \tag{113}$$

The constant term in eqn.113 is just $-Det\ A$.

10.2 The Eigenvalues of a 4×4 Matrix

We shall need eqn.113 to work out the eigenvalues of a 4×4 matrix. We shall also need this result.

$$\begin{vmatrix} a & b & c \\ d & e-\lambda & f \\ g & h & p-\lambda \end{vmatrix} = a\lambda^2 - \lambda[(ae-bd)+pa-cg)] + a(ep-hf) - b(dp-fg) + c(dh-eg). \tag{114}$$

This time the matrix A is a 4×4 matrix. We want to find values for λ such that

$$\begin{vmatrix} a-\lambda & b & c & d \\ e & f-\lambda & g & h \\ p & q & r-\lambda & s \\ t & u & v & w-\lambda \end{vmatrix} = (a-\lambda)M_1 - bM_2 + cM_3 - dM_4 = 0. \tag{115}$$

The Ms are minors of the first row of A. We shall look at them in reverse order.

$$M_4 = \begin{vmatrix} e & f-\lambda & g \\ p & q & r-\lambda \\ t & u & v \end{vmatrix} = - \begin{vmatrix} e & f-\lambda & g \\ t & u & v \\ p & q & r-\lambda \end{vmatrix} = + \begin{vmatrix} t & u & v \\ e & f-\lambda & g \\ p & q & r-\lambda \end{vmatrix} \tag{116}$$

So, if we use eqn.114, with $a \to t$, $b \to u$, $c \to v$, $d \to e$, $e \to f$, $f \to g$, $g \to p$, $h \to q$, $p \to r$, we see that

$$M_4 = t[(f-\lambda)(r-\lambda) - gq] - u[e(r-\lambda) - gp] + v(eq - fp + p\lambda)$$

$$= \lambda^2 t - \lambda[t(f+r) - ue - vp] + [g(pu - qt) + r(ft - eu) + v(eq - fp)]. \tag{117}$$

The next minor is

$$M_3 = \begin{vmatrix} e & f-\lambda & h \\ p & q & s \\ t & u & w-\lambda \end{vmatrix} = - \begin{vmatrix} p & q & s \\ e & f-\lambda & h \\ t & u & w-\lambda \end{vmatrix}. \tag{118}$$

So, if we use eqn.114, with $a \to p$, $b \to q$, $c \to s$, $d \to e$, $e \to f$, $f \to g$, $g \to t$, $h \to u$, $p \to 2$, we see that This determinant gives us

$$-M_3 = p[(f-\lambda)(w-\lambda) - hu] - q[e(w-\lambda) - ht] + s[eu - t(f-\lambda)]$$

$$= \lambda^2 p - \lambda[p(f+w) - qe - st)] + [p(fw - hu) + q(ht - ew) + s(eu - ft)]. \tag{119}$$

The laast minor of this form is M_2,

$$M_2 = \begin{vmatrix} e & g & h \\ p & r-\lambda & s \\ t & v & w-\lambda \end{vmatrix}. \tag{120}$$

For M_2 we use eqn.114, but this time with $a \to e$, $b \to g$, $c \to h$, $d \to p$, $e \to r$, $f \to s$, $g \to t$, $h \to v$, $p \to w$, we see that

$$M_2 = \lambda^2 - \lambda[(er - gp) + (ew - ht)] + e(rw - sv) - g(pw - st) + h(pv - rt). \tag{121}$$

Lastly, we need

$$M_1 = \begin{vmatrix} f-\lambda & g & h \\ q & r-\lambda & s \\ u & v & w-\lambda \end{vmatrix}. \tag{122}$$

we use the result of eqn.113 along with $a \to f$, $b \to g$, $c \to h$, $d \to q$, $e \to r$, $f \to s$, $g \to u$, $h \to v$, $p \to w$, which gives us

$$M_1 = -\lambda^3 - \lambda^2(f+r+2) - \lambda[(fw-hu)+(fr-gq)+(rw-sv)] + f(rw-sv) - g(qw-su) + h(qv-ru). \tag{123}$$

When we combine all this together, the quartic we need is

$$\begin{gathered}\lambda^4 - \lambda^3(a+f+r+w) \\ +\lambda^2[(fw-hu)+(fr-gq)+(rw-sv)+(af-be)+(ar-cp)+(aw-dt)] \\ -\lambda\{[f(rw-sv)-g(pw-su)+h(qv-ru)]+[a(fw-hu)-b(ew-ht)+d(eu-ft)] \\ +[a(fr-gq)-b(er-gp)+c(eq-pf)]+[a(rw-sv)-c(pw-st)+d(pv-rt)]\} \\ +DetA = 0..\end{gathered} \tag{124}$$

We can see that there is a lot of structure with minors of A and minors of minors of A floating about here. We always pick up the sum the diagonals, and we always pick up $+Det\ A$ as well. (Usually, we leave the lead term in the polynomial as $-\lambda^N$, so the $Det\ A$ term is always positive.) It is easy to see the structure of the polynomial when we write all this out using the definition of a determinant. In the eigenvalue problem, we must have (for a 4×4 matrix)

$$\epsilon_{ijkl}(a_{i1} - \delta_{i1}\lambda)(a_{j2} - \delta_{j2}\lambda)(a_{k3} - \delta_{k3}\lambda)(a_{l4} - \delta_{l4}\lambda) = 0. \tag{125}$$

We could have proceeded this way in the first place, and we can write down the coefficients of the polynomial for any order of matrix. However, since we have no general solution for the polynomials that need to be solved beyond the 4×4 matrix, we do not bother

11 Symmetric and Skew-Symmetric Matrices

Though we may have real entries for all the elements in a matrix, we can have clearly have complex eigenvalues: In general, a polynomial with real coefficients can have real roots, and/or roots that come in complex congugate pairs [7].

Are there matrices that always have real eigenvalues? We have been writing down a matrix times a column vector to yield a column vector. We can also define an operation where we multiply a row vector by a matrix to give another row vector. That is

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}. \tag{126}$$

[7] If i is the unit imaginary number ($i = \sqrt{-1}$) then the number $z = a + bi$ (where a and b are real numbers) is a complex number. Its complex congugate is denoted z^* and is defined as $z^* = a - ib$. The value of $zz^* = a^2 + b^2$ is the square of the modulus of z denoted $|z|$.

We can also write this as $A_{ij}x_i = b_i$, or $x^T A = b^T$. Here x^T is the row vector which is the transpose of the column vector x.

Now, suppose we have $Ax = \lambda x$ and then we multiply both sides by x^T. Then we get $x^T Ax = \lambda x^T x$. Suppose we have the eigenvectors and eigenvalues for A and A^T. The eigenvalues of A are the same set of numbers as the eigenvalues of A^T. If we have complex eigenvalues, they will occur in complex conjugate pairs. Suppose we take one such complex conjugate pair λ and λ' with corresponding eigenvectors x and x'. The eigenvalue that belongs to A^T has eigenvector x' and eigenvalue λ'. We have

$$A_{ij}x_j = \lambda x_i$$

and

$$A^T_{ij}x'_j = \lambda' x'_i \tag{127}$$

Now we multiply the first by x'^T and the second by x^T. Then

$$A_{ij}x'_i x_j = \lambda x_i x'_i$$

and

$$A^T_{ij}x_i x'_j = \lambda' x'_i x_i.$$

and subtracting gives

$$(A_{ij} - A^T_{ij})x'_i x_j = (\lambda - \lambda')x_i x'_i. \tag{128}$$

If A is a symmetric matrix then $A = A^T$. For any symmetric matrix the left hand side is zero. This means that we must have $\lambda - \lambda' = 0$ whenever A is a symmetric matrix. But remember, λ and λ' were chosen to be a complex conjugate pair! The imaginary parts must be zero.

This is an important result. *The eigenvalues of a real symmetric matrix are real.* If A had complex values for a_{ij} and $A = A^{T*}$ where the asterisk means complex conjugate then A is *Hermitian*. A similar proof exists that *the eigenvalues of a Hermitian matrix are real.* Usually, if you see A^H and A is a matrix, then A^H is meant to mean the complex conjugate of the transpose of A or its *Hermitian conjugate*[8].

If we A had been *skew* symmetric so that $a_{ij} = -a_{ji}$, the eigenvalues of A are still the same as A^T. Expanding the determinant of $A - \lambda I$ by column 1 is exactly the same as expanding $A^T - \lambda I$ by row 1. If we repeat the above exercise, we find at the last step that $\lambda + \lambda' = 0$. This means that λ has no real part and *the eigenvalues of a skew-symmetric are pure imaginary.*

As for complex eigenvalues, consider the rotation matrix R in two dimensions

$$R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \tag{129}$$

[8]Some authors use A^* or $A^\dagger$ for the Hermitian conjugate of A.

This is the sum of a symmetric matrix (the identity times $\cos\theta$) and a skew symmetric matrix with two off diagonal elements which are $\pm\sin\theta$. The eigenvalues are readily found to be $\lambda = \cos\theta \pm i\sin\theta$. Unless $\sin\theta = 0$, the eigenvalues are complex or imaginary ($\cos\theta = 0$). In general, the eigenvectors are set to have complex entries. Clearly, no real vector is an eigenvector of a general rotation matrix.

What about the eigenvectors of a skew symmetric matrix? Are they too pure imaginary? A simple example will show that the eigenvectors are, in general, complex. We examine

$$\begin{pmatrix} 0 & a \\ -a & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}. \tag{130}$$

In this case, the eigenvalues are $\pm ia$. If we substtute $\lambda = ia$ back into eqn.130, then we get

$$ay = iax$$

$$-ax = iay. \tag{131}$$

For our eigenvector, we shall use $y = ix = i(x_R + ix_I)$ where x_R and x_I are pure real. We assume the elements of the eigenvector are complex until (or if) we can show otherwise. If we substitute the vector $(x_R + ix_I, i(x_R - x_I)^T$ and $\lambda = ia$ back into eqn.130 we get

$$\begin{pmatrix} 0 & a \\ -a & 0 \end{pmatrix} \begin{pmatrix} x_R + ix_I \\ -x_I + ix_R \end{pmatrix} = \begin{pmatrix} -iax_I\frac{i}{i} + iax_R \\ -iax_R\frac{i}{i} - iax_I \end{pmatrix} \tag{132}$$

That is

$$\begin{pmatrix} 0 & a \\ -a & 0 \end{pmatrix} \begin{pmatrix} x_R + ix_I \\ -x_I + ix_R \end{pmatrix} = ia \begin{pmatrix} -\frac{x_I}{i} + x_R \\ -\frac{x_R}{i} - x_I \end{pmatrix} = ia \begin{pmatrix} x_R + ix_I \\ -x_I + ix_R \end{pmatrix}. \tag{133}$$

Of course $(-x_I + ix_R) = i(x_R + ix_I)$. We see that $(x_R + ix_I, i(x_R + x_I))^T$ is indeed an eigenvector corresponding to the pure imaginary eigenvalue $\lambda = ia$. Though both the eigenvalues and eigenvectors of a (real) symmetric matrix are pure real, the eigenvalues of a (real) skew symmetric matrix are pure imaginary, but the corresponding eigenvectors are complex.

It is also worth examine the 3×3 case for a skew symmetric matrix. This time we start off with

$$\begin{pmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ x \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{134}$$

It is easily seen that the determinant of the matrix in eqn.134 is zero. The matrix is singular: in fact *if N is odd, then an $N \times N$ skew symmetric matrix is always singular.* The eigenvalues are readily seen to be zero, and $\pm i\sqrt{a^2 + b^2 + c^2}$. The singular nature of the matrix means that it has only two independent eigenvectors.

11.1 Diagonalisation and Orthogonal Transforms

Now we shall see that if A is symmetric with *distinct* eigenvectors x_i and eigenvalues λ_j

$$Ax_i = \lambda_i x_i$$
$$x_j^T A^T = \lambda_j x_j^T. \tag{135}$$

We can pre-multiply the first by x_j^T and post-multiply the second by x_i to get

$$x_j^T (A - A^T) x_i = (\lambda_i - \lambda_j) x_j^T x_i. \tag{136}$$

Now, the left hand side is zero because A is symmetric. The eigenvalues are distinct, and $i \neq j$, so it follows that $(\lambda_i - \lambda_j) \neq 0$. The quantity $x_i^T x_j$ must then be equal to zero. But this is just the dot product of the ith and the jth eigenvectors. These dot products being zero mean that the eigenvectors are orthogonal. *Given a symmetric matrix that has distinct eigenvalues, we know the eigenvectors are orthogonal.*

Given the unit eigenvectors of a symmetric matrix A, we can form a form a matrix R^T of the column eigenvectors, and we have $RR^T = I$, and so a column vector of R^T is scaled in AR^T. Recalling §7, this R will be an orthogonal transformation that takes us to a coordinate system where the basis vectors are the orthogonal eigenvectors of A. In such a system, $A' = RAR^T$ is diagonal. (Here we regard A as the matrix rather than the tensor it describes) Any similarity transform leaves the eigenvalues unchanged, so the diagonal elements which are eigenvalues of A' are the eigenvalues of the original matrix.

So, *given any real symmetric matrix with distinct eigenvalues we can find an orthogonal transformation RAR^T in which A is diagonal.* We can immediately see what the effect of A is on a vector in these coordinates.

We shall take a look at two simple examples. In the first we shall look at a symmetric 2×2 matrix. Consider the eigen-problem

$$\begin{pmatrix} 1 & 2 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}. \tag{137}$$

This gives us

$$\begin{vmatrix} 1-\lambda & 2 \\ 2 & -2-\lambda \end{vmatrix} = 0. \tag{138}$$

The eigenvalues are 2 and -3. Given the eigenvalue of two, we want a solution for

$$\begin{pmatrix} -1 & 2 \\ 2 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{139}$$

The only information we have is that $x = 2y$. The vector $(2, 1)$ is an eigenvector and $(2/\sqrt{5}, 1/\sqrt{5})$ is a unit eigenvector. The other eigenvector is just as easy to work out.

$$\begin{pmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ 1/\sqrt{5} & -2/\sqrt{5} \end{pmatrix} \begin{pmatrix} 1 & 2 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ 1/\sqrt{5} & -2/\sqrt{5} \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 0 & -3 \end{pmatrix}. \tag{140}$$

We shall look at another eigen-problem. This one has a slight twist to the plot. The eigenvalues are *not* distinct! The eigen-problem is

$$\begin{pmatrix} 5 & 2 & 2 \\ 2 & 5 & 2 \\ 2 & 2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{141}$$

Then,

$$\begin{vmatrix} 5-\lambda & 2 & 2 \\ 2 & 5-\lambda & 2 \\ 2 & 2 & 5-\lambda \end{vmatrix} = 0. \tag{142}$$

If we add rows two and three to row one,

$$\begin{vmatrix} 9-\lambda & 9-\lambda & 9-\lambda \\ 2 & 5-\lambda & 2 \\ 2 & 2 & 5-\lambda \end{vmatrix} = (9-\lambda) \begin{vmatrix} 1 & 1 & 1 \\ 2 & 5-\lambda & 2 \\ 2 & 2 & 5-\lambda \end{vmatrix} = 0. \tag{143}$$

Now, if we subtract column 1 from columns 2 and 3 we get

$$(9-\lambda) \begin{vmatrix} 1 & 0 & 0 \\ 2 & 3-\lambda & 0 \\ 2 & 0 & 3-\lambda \end{vmatrix} = 0. \tag{144}$$

So, the eigenvalues are 9, 3, and 3. For $\lambda = 9$ we have

$$\begin{pmatrix} -4 & 2 & 2 \\ 2 & -4 & 2 \\ 2 & 2 & -4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{145}$$

We can easily see that $(1, 1, 1)$ is a solution so one unit eigenvector is $(1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})$. Otherwise we only have

$$\begin{pmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{146}$$

Because $\lambda_2 = \lambda_3 = 3$ we don't have a direction. All we know is that any vector in the plane $z = -x - y$ is an eigenvector. Let's just pick a vector in the plane! If $z = 0$, then $(1, -1, 0)$ will do: a unit eigenvector is $(1/\sqrt{2}, 1/\sqrt{2}, 0)$. We can take the cross product of this and our first eigenvector to get a third unit eigenvector. Our third eigenvector is $(1/\sqrt{6}, 1/\sqrt{6}, -2/\sqrt{6})$. The reader may verify that

$$\begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{6} & 1/\sqrt{6} & -2/\sqrt{6} \end{pmatrix} \begin{pmatrix} 5 & 2 & 2 \\ 2 & 5 & 2 \\ 2 & 2 & 5 \end{pmatrix} \begin{pmatrix} 1/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{3} & -1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{3} & 0 & -2/\sqrt{6} \end{pmatrix} = \begin{pmatrix} 9 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}. \tag{147}$$

11.1.1 The Trace of Real Symmetric Matrix

We have seen that an orthogonal transformation of a real symmetric matrix A leaves the eigenvalues unchanged. Also, if the transformation diagonalises A we can just read off the the eigenvalues from the diagonal.

Suppose that before any transformation, we have found the polynomial for the eigenvalues and determined the roots. Then we can rewrite the polynomial in terms of the roots (which are of course the eigenvalues) as

$$(\lambda - \lambda_1)(\lambda - \lambda_2)\ldots(\lambda - \lambda_N) = \prod_{i=1}^{N}(\lambda - \lambda_i) = 0. \tag{148}$$

Also, since the eigenvalues are unchanged by an orthogonal transformation, the polynomial that determines the eigenvalues are have the same coefficients *before and after* the transform. The coefficient of λ^{N-1} has a very simple form both before and after the transform. We can equate them and find

$$(a_{11} + a_{22} + a_{33} + \ldots + a_{NN}) = (\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_N). \tag{149}$$

The sum of all the diagonal elements is an invariant under orthogonal transforms and is equal to the sum of the eigenvalues.

This quantity turns out to be important, and this new invariant is called *the Trace* of a matrix. It is denoted $tr(A)$ or $tr.A$, so we have

$$Tr.A = (a_{11} + a_{22} + a_{33} + \ldots + a_{NN}) = (\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_N). \tag{150}$$

11.2 Quadratic Forms and Pricipal Axes

Given a real symmetric matrix with or without distinct eigenvalues, the *Quadratic form* Q is defined as

$$Q(x) = x^T A x. \tag{151}$$

For instance, if A represented a metric tensor, then $Q(x)$ is $|x|^2$.

In the coordinates of the diagonalised version of A, (and in two dimensions with $x^T = (x', y')$), the value of $Q(x) = x^T A' x$ is just $\lambda_1 x'^2 + \lambda_2 y'^2$. If both the eigenvalues of A are positive, then the quadratic form is *positive definite*. A curve, such that $Q = const.$ is an ellipse which is symmetric about the axes of the new coordinate system (given distinct eigenvalues). We shall set the constant term to be unity. Then minimum and maximum values of the the x', y' occur at $x' = \pm 1/\sqrt{\lambda_1}$, $y' = 0$, and $x' = \pm 1/\sqrt{\lambda_2}$, $x' = 0$. In three dimensions, we get an ellipsoid rather than an ellipse.

The use of principal axes can be applied to things like the motion od a rigid body as described by equations involving the inertia tensor of the of the body.

Our next goal will be to address the eigenvalue/eigenvector problem where we cannot solve the polynomial for the eigenvalues in closed form, and must use iterative methods. Before we go on, we ask a simple question.

Do non-symmetric matrices necessarily have at least some complex eigenvalues? The answer is no. We shall give an example of one kind of real non-symmetric matrix that always has real eigenvalues. We can write the values of the eigenvalues at a glance! If A is upper or lower triangular then $Det(A - \lambda I) = 0$ gives $(a_{11} - \lambda)(a_{22} - \lambda)(a_{33} - \lambda) \ldots = 0$. The eigenvalues are just the diagonals. Of course, the same holds if A is diagonal. For a diagonal matrix we can immediately write down the eigenvectors too. They are the column vectors of δ_{ij}. If we have a general real matrix A and pre-multiply both sides of $Ax = \lambda x$ by the L^{-1} of the LU decomposition of A we see immediately that the eigenvalues of and eigenvectors of A are *not* the eigenvectors of U. However, this does give us an insight as to how we might approach the problem of the eigenvalues of the real general matrix later on.

12 The Incredible Householder Transformation!

The reader might forgive the author for the title of this section, but it really is an ultra-cool manoeuvre (due to Alston Scott Householder) and is vital for the solution of high order eigen-problems. In any N-dimensional Euclidean space, any $N-1$ dimensional hyperplane passing through the origin can be written $(u_1x_1 + u_2x_2 + u_3x_3 + \ldots u_Nx_N) = 0$. This can be written as the dot product $u^Tx = 0$. Now this means that the vector $(u_1, u_2, u_3, \ldots, u_N)$ defines the normal of the hyperplane. We shall *define* $\mathbf{u}$ to be a unit vector. Suppose we have some vector $\mathbf{x}$ which is *not* in the hyperplane. Given its components we can write

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix} = \mathbf{u}\cdot\mathbf{x} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{pmatrix} + \begin{pmatrix} x_1 - \mathbf{u}\cdot\mathbf{x}\, u_1 \\ x_2 - \mathbf{u}\cdot\mathbf{x}\, u_2 \\ x_3 - \mathbf{u}\cdot\mathbf{x}\, u_3 \\ \vdots \\ x_N - \mathbf{u}\cdot\mathbf{x}\, u_N \end{pmatrix}. \tag{152}$$

The first vector on the right hand side is the component of $\mathbf{x}$ normal to the plane defined by $\mathbf{u}$, and the second vector is in the hyperplane defined by $\mathbf{u}$. If we subtract twice the normal component, we have a reflection in the plane. That is, a vector $\mathbf{x}'$, which is if course, exactly the same magnitude as $\mathbf{x}$. That is

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ \vdots \\ x'_N \end{pmatrix} = \begin{pmatrix} x_1 - 2\mathbf{u}\cdot\mathbf{x}\, u_1 \\ x_2 - 2\mathbf{u}\cdot\mathbf{x}\, u_2 \\ x_3 - 2\mathbf{u}\cdot\mathbf{x}\, u_3 \\ \vdots \\ x_N - 2\mathbf{u}\cdot\mathbf{x}\, u_N \end{pmatrix}. \tag{153}$$

All well and good, but so what? Well, this is what: what we want to do is find a $\mathbf{u}$ such that $x_1' = -|\mathbf{x}|$ and all the other elements are zero. We shall look at a simple case where the vector is just $(2, -1, 2)^T$. In which case we have $|\mathbf{x}| = 3$, and $(-3, 0, 0)^T$ on the left hand side.

$$\begin{pmatrix} -3 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix} - 2 \begin{pmatrix} 5/2 \\ -1/2 \\ 1 \end{pmatrix}. \tag{154}$$

Now, what is the normal of the plane which reflects the initial vector? It's clearly got to be in the direction from $\mathbf{x}'$ to $\mathbf{x}$. So, for our little problem, this is

$$\mathbf{x} - \mathbf{x}' = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix} = 2 \begin{pmatrix} 5/2 \\ -1/2 \\ 1 \end{pmatrix}. \tag{155}$$

In this case the unit normal is just

$$\mathbf{u} = \frac{1}{\sqrt{30}} \begin{pmatrix} 5 \\ -1 \\ 2 \end{pmatrix}. \tag{156}$$

So, once we have this vector there are just $O(N)$ operations to reflect any other vector. Now, it might seem a bit daft to write this down as a matrix transformation: matrix-vector multiplication is $O(N^2)$. Nevertheless eqn.153 means that $x_i' = x_i - 2u_j x_j u_i$, which can be recast as $x_i' = (\delta_{ij} - 2u_i u_j)x_j = H_{ij}x_j$. Given the vector $\mathbf{u}$ we immediately have the Householder matrix H. In our example

$$H = \frac{1}{15} \begin{pmatrix} 15 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 15 \end{pmatrix} - \frac{2}{30} \begin{pmatrix} 25 & -5 & 10 \\ -5 & 1 & -2 \\ 10 & -2 & 4 \end{pmatrix} = \frac{1}{15} \begin{pmatrix} -10 & 5 & -10 \\ 5 & 14 & 2 \\ -10 & 2 & 11 \end{pmatrix}. \tag{157}$$

The transformation leaves any vector the same length, so it is unitary. Clearly, $H^T = H$, so H is symmetric. Also $H^2 = HH^T = I$ follows immediately from the fact that it is a reflection. That is H is both orthogonal and an *involution* [9]. Given $Ax = \lambda x$, then $A'x = HAH^T x$ reflects x, transforms it, and reflects the transformed vector back. That is, the eigenvectors of this A' are the eigenvectors of A and the eigenvalues of A' are eigenvectors of A. (The eigenvectors are expressed in the new coordinates, so the fact that the eigenvalues are the same is the more important of these results.)

Now, suppose we have a matrix A with components a_{ij}. Suppose we can find the $\mathbf{v}$ which will zero all the elements of the first column vector of A. So we have formed HA which has components a_{ij}'. What happens when we now post-multiply again by HA?. This will reflect each row vector. In general, all our zeroes will be lost! That is, in general,

[9] In general, if $f(f(x)) = x$ then the function f is is an involution. In matrix algebra, a transform T is an involution if $T^2 = I$.

none of the a''_{ij} elements of HAH are zero. This is hardly surprising, for if the first column of HAH retained all the zeroes of HA, we could by successive transformations reduce A to upper triangular form. In this case we would find solutions to high order polynomials which would be linear combinations of the matrix elements (and square roots of them). We have stumbled into the Abel-Ruffini impossibility theorem in an indirect manner.

Suppose though, we don't be greedy and zero all but the first element of the vector, but retain the first two, the first element will be left alone. We shall denote this Householder transform H^1. The length of the original is $\sqrt{a_1^2 + a_2^2 + a_3^2 + \ldots}$ and the length of the reflected vector is a$\sqrt{{a'_1}^2 + {a'_2}^2}$ and $a'_1 = a_1$. So, H^1 will leave the first element (a_1) alone, and zero the second element of the two vector $(a_2, a_3)^T$=$(-1, 2)^T$ which has length $\sqrt{5}$. Our x' is this $(2, -\sqrt{5}, 0)^T$, and so

$$\begin{pmatrix} 2 \\ \sqrt{5} \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix} - 2 \begin{pmatrix} 0 \\ -(\sqrt{5}+1)/2 \\ 1 \end{pmatrix}. \tag{158}$$

The modulus of the second vector on the right is $2/(5+\sqrt{5})$. The Householder matrix is

$$H^1 = \frac{5-\sqrt{5}}{10} \begin{pmatrix} (5+\sqrt{5})/2 & 0 & 0 \\ 0 & 3/2 & (\sqrt{5}+1)/2 \\ 0 & (\sqrt{5}+1)/2 & (3+\sqrt{5})/2 \end{pmatrix}. \tag{159}$$

In general, if we zero all but the first two elements of the column vector we have a matrix of the form

$$H^1 = \begin{pmatrix} 1 & 0 & 0 & \ldots \\ 0 & h_{11} & h_12 & \ldots \\ 0 & h_{21} & h_{22} & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{160}$$

Now we are in business! Suppose we form H^1A, where H^1 zeros all but the first two elements of the first column of A. When we post-multiply by H^1 we still have all those zero elements. We can now form

$$H^2 = \begin{pmatrix} 1 & 0 & 0 & \ldots \\ 0 & 1 & 0 & \ldots \\ 0 & 0 & h'_{11} & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{161}$$

which can zero all but the first three elements of column 2 of HAH. Again when we post-multiply by H^2 we retain those zeroes. If we keep on going, forming H^3, H^4, and so on we

shall end we end up with a matrix A' of the form.

$$A' = \begin{pmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \dots & \\ a'_{21} & a'_{22} & a'_{23} & a'_{24} \dots & \\ 0 & a'_{32} & a'_{33} & a'_{34} \dots & \\ 0 & 0 & a'_{43} & a'_{44} \dots & \\ 0 & 0 & 0 & a'_{44} \dots & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{162}$$

Using successive Householder transformations, we can transform the matrix in the original eigen-problem to one of *upper-Hessenberg* form [10]. The reduction to upper Hessenberg form is the first step of solving for the eigenvalues and eigenvectors of A.

If A happens to be symmetric, then this procedure will yield a tridiagonal matrix. This is why a finding the eigenvalues for a real symmetric matrix can be made very efficient compared to finding the eigenvalues of a general non symmetric matrix.

13 Iterative Procedures.

The simplest iterative procedure finds only one eigenvalue and its eigenvector. This is called power iteration. Suppose we have a matrix A. We haven't got a clue what its eigenvectors are. We shall name the eigenvectors of A as $\mathbf{a}_i$. Suppose we take some initial vector $\mathbf{v}_0$. We can (in principle) express $\mathbf{v}_0$ in terms of the unknown eiegenvectors $\mathbf{a}_i$. In the following we shall assume that the eigenvalues are all different from each other (or distinct) and real. Though we assume the eigenvalues are real, we do not assume the matrix is symmetric. We shall leave the case where eigenvalues can be equal or complex for a separate section.

Now, we don't know what these coefficients are, but whatever they are we can express our initial "guess" vector as

$$\mathbf{v}_0 = \alpha_1 \mathbf{a}_1 + \alpha_2 \mathbf{a}_2 + \alpha_3 \mathbf{a}_3 + \dots \tag{163}$$

We can take this multiply our initial vector by A to get

$$\mathbf{v}_1 = \alpha_1 \lambda_1 \mathbf{a}_1 + \alpha_2 \lambda_2 \mathbf{a}_2 + \alpha_3 \lambda_3 \mathbf{a}_3 + \dots \tag{164}$$

But then we could multiply $\mathbf{v}_1$ by A to get $\mathbf{v_2}$, and multiply again to get $\mathbf{v}_3$ and so on. After p iterations we get

$$\mathbf{v}_p = \alpha_1 \lambda_1^p \mathbf{a}_1 + \alpha_2 \lambda_2^p \mathbf{a}_2 + \alpha_3 \lambda_3^p \mathbf{a}_3 + \dots \tag{165}$$

Supposing that the eigenvalues are distinct, then one term will dominate more and more as we continue to iterate. The vector $\mathbf{v}_p$ gets closer and closer to an eigenvector as we

[10] The elements of an upper-Hessenberg matrix $(UH)_{ij}$ are all zero if $i > j + 1$ whereas the elements of an upper triangular matix U_{ij} are zero if $i > j$.

keep on iterating. The vector that dominates *might* be the eigenvector with the largest of the the eigenvalues, but we must remember that any of the α_i could be zero depending on the initial vector $\mathbf{v}_0$. So, all that is guaranteed is that we have an eigenvector and an eigenvalue. To stop the numbers growing too large or small we can just normalise $\mathbf{v}_0$, and normalise each new vector as we iterate. We shall call the normalised vectors $\mathbf{u}$.

Of course, we might want to know the smallest eigenvalue or the nearest eigenvalue to a given number. To find the smallest eigenvalue we need only consider that if $Ax = \lambda x$ we can multiply both sides by A^{-1} and see that $A^{-1}x = x/\lambda$. So power iteration on A^{-1} will converge on the smallest eigenvalue given that the eigenvalues are distinct. Similarly if we consider $(A - \mu I)x = (\lambda - \mu)x$ where μ is some number, then $(A - \mu I)^{-1}x = x/(\lambda - \mu)$, so power iteration on $(A - \mu I)^{-1}$ will converge on the nearest eigenvalue to μ. Both of these are called inverse power iterations. As always, we may need to try different initial vectors as our initial $\mathbf{v}_0$ may have some zero coefficients if expanded in terms of the (unknown) eigenvectors.

Now we shall look at what happens if we want to determine not just the largest eigenvalues, but the largest three, say. Again, the argument shall depend on the eigenvalues being distinct from each other. Suppose we start with three *independent* vectors $\mathbf{u}$, and try to use these to determine the three largest eigenvalues simultaneously. (Given what we know about power iteration this seems a non-starter at first.) Since A is nonsingular and has N independent eigenvalues, when we multiply our three vectors by A we still have three independent vectors.

After p iterations we have $A^p\mathbf{u_n}$ (normalising at each step) for $n = 1, 2$ and 3, then all three vectors will get closer to the largest eigenvector, and all three vectors will be almost parallel. They will be in the same directions as if we had not normalised at each step. For large enough p we know that $\lambda_1^p >> \lambda_2^p >> \lambda_3^p >> \lambda_4^p$.

Now, we suppose that A is an $N \times N$ matrix where $N > 3$. The first three (distinct) eigenvectors of A with the three largest eigenvalues form a three dimensional subspace of the N dimensional space spanned by *all* the eigenvectors of A. Note, that if x is any vector in this subspace it can be expressed in terms of these three eigenvectors. Then if if we multiply this x by A it will still be in this subspace.

Our initial guess vectors form a *different* three dimensional subspace, and generally are *not* orthogonal to the $N - 3$ eigenvectors which are not in the three dimensional subspace formed by the first three eigenvectors. In an unnormalised version, because $\lambda_3^p >> \lambda_4^p >> \ldots >> \lambda_N^p$, with each iteration, the projection of each of the three has a (relatively) smaller projection into the $N - 3$ dimensional space and a (relatively) larger projection into the 3 dimensional subspace. That is, as we iterate, our three vectors get closer and closer to spanning the same three dimensional subspace as the first three eigenvectors.

Numerically, our unnormalised version is impossible using ordinary double precision

arithmetic on a computer. Generally we end up with Inf or zero. If we normalise at each step, we get three unit vectors, but with double precision, these will be either parallel or so close to parallel as to lose all accuracy.

Suppose however, that instead of just multiplying the vectors by A and normalising over and over, we find a new set of vectors at each stage which span the same three dimensional space as the three vectors Ax. To make sure our vectors don't migrate to an almost parallel set, we make them an orthonormal set. So our algorithm is

```
1.  Choose three independent vectors v
2.  Convert to three orthonormal vectors u (spanning the same space).
3.  Multiply each u by the matrix A.
4.  Normalise.
5.  Go to step 2.
```

We get an estimate of the three eigenvectors at step 4 and can break the loop if they have converged. But how do we obtain these three orthonormal vectors spanning the same space as the given vectors?

The procedure to do this is called *Gram-Schmidt orthogonalisation*. After multiplying by A we have three vectors $\mathbf{v}_n$ which are not orthonormal. We may make a new $\mathbf{u_1}$ by normalising $\mathbf{v}_1$. Then $\mathbf{v}_2 \cdot \mathbf{u_1}$ is the component of $\mathbf{v}_2$ in the direction of $\mathbf{u_1}$. If we subtract off $(\mathbf{u_1} \cdot \mathbf{v_2})\mathbf{v}_2$ from $\mathbf{v}_2$ we have a vector orthogonal to $\mathbf{u_1}$ and so we can normalise this to get a new unit vector $\mathbf{u}_2$. Then we can form $(\mathbf{v}_3 \cdot \mathbf{u_1})\mathbf{v}_3$ and $(\mathbf{v}_3 \cdot \mathbf{u_2})\mathbf{v}_3$, and subtract these from $\mathbf{v}_3$ and the result will be orthogonal to both $\mathbf{u}_1$ and $\mathbf{u}_1$. We may normalise this to get a new $\mathbf{u}_3$.

We can write all this in matrix form. Given two *non-square* matrices B and C we can use the usual rule for matrix multiplication as long as the number of columns in B is the same as the number of rows in C. The Gram-Schmidt method turns out to be what is called a QR factorisation. A QR factorisation factors a matrix A into an orthogonal matrix Q and an upper triangular matrix R, so $A = QR$. We can make this factorisation *unique* by factorising in such a way that all the diagonal elements of R are positive. If we find a negative on the diagonal of R, all we have to do is change the sign of the row vector of R with the negative value, and then swap the signs of the corresponding column vector of Q.

For three initial vectors $\mathbf{v}$, the column vectors of the new u can be written down as

$$\begin{pmatrix} u_1^1 & u_2^1 & u_3^1 \\ u_1^2 & u_2^2 & u_3^2 \\ u_1^3 & u_2^3 & u_3^3 \\ \vdots & \vdots & \vdots \\ u_1^N & u_2^N & u_3^N \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} v_1^1 & v_2^1 & v_3^1 \dots & v_N^1 \\ v_1^2 & v_2^2 & v_3^2 \dots & v_N^2 \\ v_1^3 & v_2^3 & v_3^3 \dots & v_N^3 \end{pmatrix}. \tag{166}$$

On the l.h.s is an orthogonal matrix Q, so eqn.150 can be written as $Q = LV$. We can transpose this and see that $Q = VL^T = V\mathcal{R}$ where $\mathcal{R} = L^T$ is upper triangular.

$$\begin{pmatrix} u_1^1 & u_2^1 & u_3^1 \\ u_1^2 & u_2^2 & u_3^2 \\ u_1^3 & u_2^3 & u_3^3 \\ \vdots & \vdots & \vdots \\ u_1^N & u_2^N & u_3^N \end{pmatrix} = \begin{pmatrix} v_1^1 & v_2^1 & v_3^1 \\ v_1^2 & v_2^2 & v_3^2 \\ v_1^3 & v_2^3 & v_3^3 \\ \vdots & \vdots & \vdots \\ v_1^N & v_2^N & v_3^N \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}. \tag{167}$$

If we post-multiply by $R = \mathcal{R}^{-1}$ to get $V = QR$. Of course, we recall that the inverse of an upper triangular matrix R is also upper triangular. We can denote the coefficients of this R as r_{ij}. So Gram-Schmidt orthogonalisation is actually equivalent to a QR factorisation of the initial vectors stored in V. We have

$$\begin{pmatrix} v_1^1 & v_2^1 & v_3^1 \\ v_1^2 & v_2^2 & v_3^2 \\ v_1^3 & v_2^3 & v_3^3 \\ \vdots & \vdots & \vdots \\ v_1^N & v_2^N & v_3^N \end{pmatrix} = \begin{pmatrix} u_1^1 & u_2^1 & u_3^1 \\ u_1^2 & u_2^2 & u_3^2 \\ u_1^3 & u_2^3 & u_3^3 \\ \vdots & \vdots & \vdots \\ u_1^N & u_2^N & u_3^N \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix}. \tag{168}$$

So, given an initial set of guess vectors we have a matrix V_0 consisting of three column vectors. We then find Q_0 and R_0 such that $Q_0 = V_0 R_0$ and form $V_1 = AQ_0$, and keep going. As the iterations continue, the three unit vectors in Q get closer and closer to spanning the three dimensional subspace of the first three eigenvectors.

Let's see what's going on here. The vector $\mathbf{u}_1$ is always just normalised in the Gram-Schmidt procedure so it evolves just as if we were looking for the single largest eigenvector. (We are assuming that our initial guess vectors have no zero coefficients if we knew their expansions in terms of eigenvectors.) This means that the first column vector of Q approaches the unit eigenvector with the largest eigenvalue. The value of r_{11} approaches the largest eigenvalue.

Just for now, suppose that we had iterated just two vectors. At the end of the iteration, the first vector is the largest eigenvector. The two vectors together span the space of the first two orthogonal eigenvectors, so the second vector is an eigenvector too. But with the three vectors we discussed above, the second eigenvector evolves in exactly the same way as if the third vector were not there, so the third vector would be an eigenvector too! The components of these eigenvectors are *not* the components of the eigenvectors in the initial coordinate system (we address that in the next section) but we have the three largest eigenvalues. We are getting close. In the next section we shall see how a general algorithm might be arrived at, still with the proviso that the eigenvalues are real and distinct.

14 The QR Algorithm

The obvious question is, why not go for all the eigenvalues? If we use the same algorithm as above and we use the identity matrix as our initial guess so that $V_0 = I$, then $V_0 = I = Q_0 R_0$. Obviously $Q_0 = I$ and $R_0 = I$. Then we put $V_1 = AQ_0 = A$. Now we find Q_1 such that $V_1 = Q_1 R_1 = A$. Then using the same algorithm described in the previous section, at the next step we have $V_2 = AQ_1 = Q_2 R_2$. Then $V_3 = AQ_2$ and so on.

We see that at any stage we could form $Q_n^T A Q_n$ and we get a succession of similarity transforms. If we get convergence, successive transformations bring us closer and closer to an upper triangular matrix, and the diagonal elements of the similarity transform get closer and closer to the eigenvalues. The almost magical thing is that we can do all these similarity transforms without ever forming a matrix Q or doing a matrix multiplication. We find that we can generate these similarity transforms as follows.

First, we introduce some modifications to what we have done so far. We use A_0, A_1, ... instead of V_0, V_1, ... and instead of putting the new $A_i = AQ_{i-1}$ we put $A_i = Q_{i-1} R_{i-1}$. So we now have $A_0 = A$. Then $A = Q_1 R_1$ and $A_2 = R_1 Q_1 = Q_1^T A Q_1$. At the next step $A_2 = Q_2 R_2$, and $A_3 = R_2 Q_2 = Q_2^T Q_1^T A Q_1 Q_2$. Our modified algorithm is now

```
With A_0=A, and k=1,2,3,4,...
Form Q_k R_k from A_(k-1)
Set A_k=R_k Q_k
```

This latter algorithm is the QR algorithm. The former algorithm is the simultaneous iteration algorithm applied to all the eigenvalues using the column vectors of the identity matrix for V_0. How different are the two algorithms? Both produce a sequence of similarity transforms on A. Suppose we call the orthogonal transformation in the simultaneous iteration S and call the corresponding upper triangular matrix T. We make a table of the first few steps for both algorithms.

step	ST	QR
-1a	$X_{-1} = I$	$*$
-1b	$S_{-1}T_{-1} = I^2 = I$	$*$
0a	$X_0 = AS_{-1} = A$	$A_0 = A$
0b	$X_0 = S_0T_0 = A$	$A_0 = Q_0R_0$
0c	$X_1 = AS_0$	$A_1 = R_0Q_0$
0d	$A'_1 = S_0^TAS_0$	$A_1 = R_0Q_0 = Q_0^TAQ_0$
1a	$X_1 = S_1T_1$	$A_1 = Q_1R_1$
1b	$X_2 = AS_1$	$A_2 = R_1Q_1$
1c	$A'_2 = S_1^TAS_1$	$= Q_1^TA_1Q_1 = Q_1^TQ_0^TAQ_0Q_1$
2a	$X_2 = S_2T_2$	$A_2 = Q_2R_2$
2b	$X_3 = AS_2$	$A_3 = R_2Q_2$
2c	$A'_3 = S_2^TAS_2$	$= Q_2^TQ_1^TQ_0^TAQ_0Q_1Q_2$

Of course the A' aren't really a part of the algorithm, but we shall use them to examine any differences between the two methods. The first occasion where the two methods appear to differ is step 1a. Before this we can see that $S_0 = Q_0$ and $R_0 = T_0$. Then at step 1a we have $X_1 = S_1T_1 = AS_0$ on the one hand and $A_1 = Q_1R_1 = Q_0^TAQ_0$ on the other. But if we look at this last term we see that $Q_0Q_1R_1 = AQ_0$. That is we have an orthogonal matrix S_1 such that $S_1T_1 = AS_0$, an orthogonal matrix $Q_0Q_1R_1 = AQ_0$, and from the previous step, $S_0 = Q_0$. Now, our QR factorisation is unique. So $S_1 = Q_0Q_1$ and $T_1 = R_1$. Then of course $S_1^TAS_1 = Q_1^TQ_0^TAQ_0Q_1$, and the matrices A' and A are identical. The same must now apply to step 2, then it follows the same applies for step 3, and so on. The QR algorithm is identical to the simultaneous iteration methods for all the eigenvectors given the initial "guess" matrix is the identity.

We mentioned that we don't actually use all these matrix transformations. What we actually do is use the Householder transform. First, we transform A to upper Hessenberg form using Householder transforms so we start the QR algorithm with an upper Hessenberg matrix A_0 with elements a_{ij}^0.

Now, in the following, we actually do the matrix operations, though we would never do so in practice. We may pre-multiply by H^1 to zero a_{21}^0, then pre-multiply by H^2 to zero a_{32} and so on. We end up with an orthogonal matrix Q_0 multiplied by an upper triangular matrix R_0 with positive diagonal elements. That is we have $A = Q_0R_0$. We need to end up with a matrix with the same eigenvalues as A has, so we must complete the similarity transform and put $A_1 = R_0H^1H^2 \ldots H^{(N-1)} = R_0Q_0$. So, we can cast everything in the QR algorithm in terms of Householder transforms.

Consider the reflection of the nth column. The vector $\mathbf{v}_n$ is

$$\begin{pmatrix} a_1 \\ \vdots \\ a_{n-1} \\ a_n \\ a_{n+1} \\ 0 \\ \vdots \end{pmatrix} - \begin{pmatrix} a_1 \\ \vdots \\ a_{n-1} \\ \sqrt{a_n^2 + a_{n+1}^2} \\ 0 \\ 0 \\ \vdots \end{pmatrix} = 2 \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_n/2 - 1/2 \times \sqrt{a_n^2 + a_{n+1}^2} \\ a_{n+1}/2 \\ 0 \\ \vdots \end{pmatrix} \tag{169}$$

The matrix $u_i u_j$ (remember $\mathbf{u}$ is the normalised version of $\mathbf{v}$) will be zero apart from a two by two block. Let's look at just $H^1 H^2$. This will be

$$H^1 H^2 = \left[\begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{pmatrix} - 2 \begin{pmatrix} u_1 u_1 & u_1 u_2 & 0 & 0 & \dots \\ u_1 u_2 & u_2 u_2 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{pmatrix} \right]$$

$$\times \left[\begin{pmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{pmatrix} - 2 \begin{pmatrix} 0 & 0 & 0 & 0 & \dots \\ 0 & u'_1 u'_1 & u'_1 u'_2 & 0 & \dots \\ 0 & u'_1 u'_2 & u'_2 u'_2 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots \end{pmatrix} \right]. \tag{170}$$

The reader shall see that as we go on, we shall eventually form an upper Hessenberg matrix where the subdiagonal is just $u_1 u_2$, $u'_1 u'_2$, ... So Q is an orthogonal upper Hessenberg matrix and if we multiply out RQ we generate another upper Hessenberg matrix. The elements in subdiagonal of the new upper Hessenberg matrix will be be the diagonal elements of R multiplied a subdiagonal of element of Q which shall be less than one, however these new subdiagonals are related to the old subdiagonal on the column to the right in the original. As we iterate the subdiagonal elements will (generally) get smaller and smaller. When they are all close enough to zero, the diagonal elements are the eigenvalues of A. Convergence is linear, and may be slow.

But what about the eigenvectors? The eigenvectors of the upper triangular matrix are eigenvectors of the operator in a completely different coordinate system. For a symmetric matrix the final matrix is diagonal and the eigenvectors are the column vectors of the identity. Otherwise, the first column vector will have converged to an eigenvector. That is, $(1, 0, 0, \dots)^T$ is an eigenvector with eigenvalue a_{11}. Another eigenvector is a linear combination of this first one with the second column of the upper triangular matrix and so on. We can obtain the eigenvectors of the upper triangular matrix this way. However, this would way too slow. We have an $O(N^4)$ operations. We can readily invert the upper

triangular matrix however, and since we now have the eigenvalues we can now use inverse iteration on the linear combinations of eigenvectors very quickly. The first inverse iteration is on a 2×2 system, the second on a 3×3 system, and so on.

But these eigenvectors have components according to to some unknown reference frame. In order to get the eigenvectors in the original reference frame we need to transform back! That is we need the product of all the orthogonal transforms. We can compute this without any matrix multiplications. We start off with the identity matrix. Each Householder transform we use in transforming to upper Hessenberg is applied to the identity, and then once we have this, we apply each Householder in every iteration of the QR algorithm to the result. Again, we have no matrix multiplications in practice. Eventually we build up Q, and can transform back to the original reference frame. The cost is a fair bit extra, but it is still $O(N^3)$.

Let's just look at eigenvalues, and see the QR algorithm in operation for a couple of simple 2×2 matrices. Obviously Q is just the single Householder matrix that zeroes a_{21}. Any 2×2 matrix is already upper Hessenberg.

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}. \tag{171}$$

The matrix is symmetric, so this property will be preserved by a similarity transform. The transformation to upper Hessenberg form results in a tridiagonal matrix. Again, a 2×2 matrix is already tridiagonal. With a symmetric matrix, as the subdiagonal elements decrease, so do the elements of the single superdiagonal. The matrix becomes diagonal.

Back to our 2×2 matrix. The first iteration gives

$$\mathbf{u}_0 = \begin{pmatrix} -.526 \\ .961 \end{pmatrix},\ Q_0 = \begin{pmatrix} .447 & .894 \\ 0.894 & -0.447 \end{pmatrix},\ R_0 = \begin{pmatrix} 2.236 & .358 \\ 0 & 0.447 \end{pmatrix},\ A_1 = \begin{pmatrix} 4.2 & .4 \\ .4 & -.2 \end{pmatrix}. \tag{172}$$

The next iteration gives

$$\mathbf{u}_1 = \begin{pmatrix} -.047 \\ .9988 \end{pmatrix}, Q_1 = \begin{pmatrix} .9955 & .0948 \\ 0.0948 & -0.9955 \end{pmatrix}, R_1 = \begin{pmatrix} 4.219 & .379 \\ 0 & 0.237 \end{pmatrix}, A_2 = \begin{pmatrix} 4.236 & .022 \\ .022 & -.236 \end{pmatrix}. \tag{173}$$

The actual eigenvalues are $2 + \pm\sqrt{5}$ and are approximately 4.236068 and -0.236068 and in this instance, another two iterations gives an answer correct to seven decimal places. The off-diagonals are 0.00007 at this stage. Lets see what happens with the initial matrix

$$A_0 = \begin{pmatrix} 2 & 0 \\ 4 & 2 \end{pmatrix}. \tag{174}$$

The eigenvalues are both two. Now, even after 10 iterations the subdiagonal is still as large as 0.01 and the diagonal elements are 2.2 and 1.8: nowhere near the correct values.

Even after 1000 iterations we have 2.002 and 1.998, a 0.1% error. If we set the one of the diagonals in A^0 to be to be a bit larger than two, 2.01 say, we should have eigenvalues of 2.01 and 2.0. After 100 iterations we get 2.02607 and 1.98393. After 200 iterations we have 2.0159 and 1.994, after 400 iterations we have 2.011 and 1.998. Things are moving *very* slowly indeed. It takes 1000 iterations to get to 2.0100686 and 1.9999306, a lot better than the 0.1% error we had earlier, but still pretty ropy for 1000 iterations. If we have

$$A_0 = \begin{pmatrix} 1 & -2 \\ 1 & 1 \end{pmatrix}, \tag{175}$$

the eigenvalues are $1 \pm \sqrt{2}$, if we try our QR algorithm, the only subdiagonal term never becomes small, and we never get the eigenvalues this way.

Clearly, this raw version of the QR algorithm isn't fit for purpose. We have to get lucky and have matrices where the eigenvalues are real and widely separated. So, now we know what the basic QR algorithm is, lets dig a bit further and see how it has to be modified to get it to be more robust, and to increase the speed of the convergence.

15 Towards a Practical QR algorithm.

First of all, though we write everything down as matrix multiplications, we don't ever do any matrix multiplications. All we need to do is store the normal vectors of the reflection planes and use eqn.153 to form each column of R_i. If we find that we have a negative diagonal element of R we describe the plane normal with the negative of the original normal vector. and use the same vectors to reflect the rows of R_i to complete the operation $Q^T RQ$. There are no expensive matrix multiplications in sight.

Also, as this is not meant to be a textbook, whenever the discussion would become too involved, we shall just state some results. This article is meant to be a primer before reading the more difficult literature and we do not aim to compete with the classic textbooks.

First, we shall look at *Rayleigh quotient* iteration for a single eigenvalue. This is a variation on inverse power iteration that converges more rapidly than the simple inverse power iteration we have encountered so far. We have seen that depending on the matrix, it may take many thousands of iterations to get anywhere near the desired accuracy.

We observe that it might well be worthwhile to change the parameter μ and compute a new $(A-\mu I)^{-1}$ as we iterate. How do we choose a "new improved μ" at each stage? Given a vector x and a matrix A, *the Rayleigh quotient* $r(x)$ *is defined as,*

$$r(x) = \frac{x^T Ax}{x^T x}. \tag{176}$$

Obviously, if x happens to be an eigenvector, the Rayleigh quotient is the corresponding eigenvalue. Now suppose we have some vector $\mathbf{v}$ expanded as in eqn.163. Can we find a

number ϕ that minimises $f(\phi) = |Av - \phi v|^2$? One easily finds that

$$f(\phi) = g_{ij}[\lambda\alpha]_i[\lambda\alpha_j] - 2\phi g_{ij}\lambda_i\alpha_j|_{i\neq j} + \phi^2 g_{ij}\alpha_i\alpha_j. \tag{177}$$

Differentiating, we find that the value of ϕ that minimises $f(\phi)$ is exactly the Rayleigh quotient defined above.

This leads to the more rapidly converging Rayleigh quotient algorithm. We start off with a "guess" at a value of λ which we call λ^0. Next we replace the previous constant μ with λ^0. At each step $k = 1, 2, 3\ldots$ we solve $(A - \lambda^{k-1}I)\mathbf{w} = \mathbf{v}^{k-1}$ for $\mathbf{w}$. (This is more efficient than calculating the actual inverse.) Then we normalise $\mathbf{w}$ to get $\mathbf{v}^k$. The new value of λ^k is then set to be the Rayleigh quotient of the new $\mathbf{v}$. The convergence of this algorithm is cubic, and the values of the λ are called *shifts*.

How can we modify the QR algorithm of the previous section to incorporate the something like the Rayleigh quotient algorithm? It can be shown for the case of a symmetric matrix, that if we introduce a particular permutation matrix, that the QR algorithm can do simultaneous inverse power iteration without ever calculating the inverse matrix. However, even for the general matrix, we can accelerate to cubic convergence just as with the the Rayleigh quotient iteration.

Recall that when we took a look at how the iteration worked, data migrated from right to left and the subdiagonal elements converged in a linear fashion. Now we can make the following observation. Take A which is an N by N upper Hessenberg matrix, and suppose that $Ax = \lambda x$. We can see that if we take A', the 2 by 2 matrix at the bottom right hand corner, and the two vector x' formed by the bottom two elements of x, then $A'x' = \lambda x'$. This means that the eigenvalues of A' contain important information. If the eigenvalues of A are real and distinct, we can introduce the shift μ as the smallest of the two eigenvalues. The *Wilkinson Shift* is slightly different and solves the case where we have repeated eigenvalues. Here, we use the eigenvalue that is closest to the bottom right element of the upper Hessenberg. In the case where they are equally close, we choose one of the eigenvalues at random. This breaks the deadlock in the case of repeated eigenvalues. Whichever shift we use, we obtain much better convergence in the QR algorithm if we use

$$A^{k-1} - \mu^k I = Q^k R^k$$

$$A^k = R^k Q^k + \mu^k I. \tag{178}$$

For the symmetric matrix (with appropriate permutation matrices) this can be shown to be identical to inverse power iteration with the Rayleigh quotient accelerated iteration. However, it works in general. Note that the similarity transform is identical to what we get with no shift.

There is another way to accelerate things, and this is by a process called *deflation*. Suppose that at some stage an subdiagonal element in the upper Hessenberg matrix has

become (in effect) zero. We can then write this matrix as

$$A = \begin{pmatrix} B & C \\ 0' & D \end{pmatrix}, \tag{179}$$

where C and a (close to) zero matrix $0'$ are not necessarily square, and B and D are upper Hessenberg. To be more specific if the rth sub-diagonal element (counting from the left) is some small value ϵ_1, then B is an $(r \times r)$ matrix, and D is $(N-r) \times (N-r)$. Then the (close to) zero matrix is $(N-r) \times r$, with the only non-zero element being at the top right with value ϵ_1. Finally the matrix C is $r \times (N-r)$.

Suppose x is any eigenvector of A and λ is the corresponding eigenvector. If we split x into y and z which are r and $(N-r)$ vectors, we see that $Dz = \lambda z$ and $Ay + Cz = \lambda y$. Clearly, Cz must be a zero vector. We can discard C altogether, and continue by finding the eigenvalues of the smaller matrices B and D. In turn these matrices may be split up, and so on. As the iteration proceeds we deal with an ever smaller chunks.

The last question is, how on Earth do we get complex eigenvalues out of the QR algorithm? It is obvious that without shifts, the QR algorithm never converges. However, now we are using shifts. What happens is, that if any of the λ are complex, at some stage the eigenvalues of the bottom 2 by 2 matrices (remember we now have many sub-matrices due to deflation) will be complex. After this, the complex numbers will begin to propagate. OK, that's how we get them, but what happens in practice? First, we do not want complex matrices or to do complex arithmetic in general. This is very expensive in both time and storage. We shall not go into any detail here, but we shall state that there is an implicit double shift (due to Francis) using the complex conjugate roots. This avoids complex arithmetic altogether.

So, here we shall summarise what is going on, and make some final comments. First, we find the eigenvalues, this is the hard bit. The matrix is reduced to upper Hessenberg form. Then we use the QR algorithm with Wilkinson shifts and solve successively smaller eigenproblems due to the deflation process. It is peculiar, but that bottom right 2 by 2 matrix is all important. Whenever it has a complex conjugate, we use Francis' double implicit shift (which we have not stated here). For symmetric matrices the problem is much easier and quicker, and needs much less storage.

So, is this all there is? The answer is a clear no! There are many special problems that may occur depending on the matrix, and also many application areas that give rise to matrices of a particular form. In fact, the eigenvalue/eigenvector problem is by no means a closed book. There is a lot of current research going on. In the case of large parallel algorithms, the QR might not even be of use! Also, we have by no means given a complete picture of even basic linear algebra. We have not even seen a Cholesky factorisation or a Givens rotation.

16 Singular Value Decomposition

Recall that in §11.1 we found that if A has a symmetric square matrix A it could be diagonalised by a similarity transform. This transform used the orthogonal matrix R of the eigenvectors of A. This can be generalised to a non-square matrix (or a square non-symmetric matrix) via Singular Value Decomposition (SVD). This has a wide range of applications throughout widely differing subject areas — from statistics to quantum theory to image processing. It can be achieved by an iterative procedure which is a variation of the QR algorithm. We do not go into any details about this though.

If we have two non-square matrices, A and B so that the number of columns in A is the number of rows in B we can form $C_{ij} = A_{ik}B_{kj}$. Then C is a matrix with the same number of rows as A and the same number of columns as B. Clearly, if A is an $m \times n$ matrix, then AA^T and A^TA are $m \times m$ and $n \times n$ matrices. The idea of SVD is that we can always find orthogonal square matrices such that

$$A = U\Sigma V^T, \tag{180}$$

where U is $m \times m$, $V = n \times n$ and Σ is a diagonal $m \times n$ matrix. Obviously, there is no long diagonal if $m \neq n$, Σ's elements are denoted σ_{ij} and are zero if $i \neq j$. They may or may not be zero if $i = j$. As such we call $\sigma_{i,j=i}$ (the diagonal elements) σ_i *the singular values* of the matrix. By convention they are ordered so that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$. We state that either one of U and V *might not be unique*.

Clearly, since U and V are orthogonal, then

$$U^TAV = \Sigma. \tag{181}$$

Recall that, for a real symmetric matrix A we can always find a real orthogonal matrix R such that RAR^T is diagonal. We see at once that there are similarities and differences between this and SVD (even when $m = n$). We also see that if A is symmetric then $R = U^T$ and $V = R^T$. In this case U and V are unique. In fact the same is true if $AA^T = A^TA$. Matrices that satisfy this include symmetric matrices as a subset and are called *normal* matrices. This extends to the Hermitian conjugate if A is complex valued.

Going back to eqn.180, we soon see that

$$AA^T = U\Sigma\Sigma^TU^T$$

$$A^TA = V\Sigma^T\Sigma V^T. \tag{182}$$

Furthermore, we may post-multiply by U and V to get

$$AA^TU = U\Sigma\Sigma^T$$

$$A^TAV = V\Sigma^T\Sigma. \tag{183}$$

Now we can see what is going on. If we regard U and V as consisting of column vectors, then these column vectors consist of eigenvectors of AA^T and A^TA. Clearly AA^T and A^TA are symmetric and so U and V are orthogonal. The column vectors v of V are called the right singular vectors, and the column vectors u of U are the left singular vectors. Then $Au = \sigma v$ and $A^Tv = \sigma u$. Also, for instance if Σ is 2 by 3, then we have

$$\Sigma\Sigma^T = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix},$$

$$\Sigma^T\Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{184}$$

If the matrix is not square, either $\Sigma\Sigma^T$ or $\Sigma^T\Sigma$ has a row of zeroes, that is for non-square matrices, at least $|m - n|$ of of the eigenvalues of the largest of the square matrices AA^T are zero. If a matrix has a zero eigenvalue it is singular.

Recall that in §4 we introduced the idea of a condition number for matrices which would enable us to estimate the accuracy of the solution of linear equations. We gave an example where the solution that gave the smallest residual was in fact the least accurate. We shall see that SVD used on square matrices gives us a much more powerful insight into the accuracy of linear equations, matrix inversion, and eigenproblems. The difference here is that the definition of condition number in this section has no condition numbers less than unity. We shall mention only one other practical use of SVD, and that will be the pseudo-inverse of linear least square problems.

Before we continue, we must introduce the ideas of rank, nullity, and range. Consider a matrix A which we shall regard as a set of column vectors. If these column vectors are linearly independent we can regard them as the basis vectors of an n dimensional space. If they are independent, then the matrix is said to be of full rank, or of rank n in an $n \times n$ matrix. Strictly speaking, this is the column rank. Row rank can be defined in a similar way, but it turns out that the row and column rank are always the same for a square matrix, so in this case we may just use the term rank.

If there are only $n - p$, with $p > 0$, linearly independent vectors of A then A is said to be rank deficient. It has rank $n - p$ and nullity p. Consider the obvious and simple case

$$\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{185}$$

Obviously, the determinant is zero, and the rows are not independent. Any vector in the one dimensional subspace of the (x, y) plane consisting of the line $y = -x$ is mapped to the origin. This subspace of the 2-D plane is called the null-space of the matrix and is of dimension 1 (the line $y = -x$). We may define a unit vector $\mathbf{e}_1$ pointing along $y = -x$ and

another $\mathbf{e_2}$ pointing along $y = x$. Then any vector $\mathbf{r} = a\mathbf{e}_1 + b\mathbf{e_2}$ is mapped into the one dimensional subspace along $y = x$. This other 1 dimensional subspace is called the range of the matrix. Any vector in this subspace is mapped onto another vector in the subspace. If the right hand side were a non zero vector which happened to be in the range of x then there would be an (x, y) pair that satisfies eqn.185.

For an $n \times n$ matrix A, there may be a p dimensional subspace such that any vector in this subspace is mapped to the zero vector. This subspace is the null-space of A and A has nullity p. The complement to this null-space, of dimension $n - p$ is the range of A. Any vector in the range of A is mapped to another vector in the range of A and $n - p$ is the rank of A.

Going back to eqn.185, we see the matrix has eigenvalues of 0 and 3. The corresponding (non unit) eigenvectors are $(1, -1)^T$ (eigenvalue 0), and $(1, 2)^T$. In general, the null space is spanned by the eigenvectors with eigenvalue 0, and the range is spanned by the eigenvectors with non zero eigenvectors.

Now, suppose we have a system of linear equations $Ax = b$ which is nearly singular. First we may ask, what is the rank of A? The matrix may seem to be of full rank, but if the singular values are small the matrix may be rank deficient, but only appear to be of full rank because of rounding errors or the precision of the representation of A. If σ_i/σ_1 is below some tolerance, we can decide that we cannot trust our results. Here we set these particular σ_i to zero. So we can make an approximate measure of rank which is less than the apparent full rank if we have singular values which are too small. This gives us a more practical definition of the rank of a matrix. What do we do when we have set some of the σ_i to zero? We must modify the original matrix in some way to get the best possible solution, this modification is called matrix approximation. Later we look at the concept of a matrix *norm* which we denote $|A|$, in particular the Frobenius norm. If we have an ill conditioned matrix, and must set smallest p singular values to zero so the matrix will have rank $r = min((m, n) - p)$. We have a new Σ denoted by Σ'. We shall denote our approximate matrix as A'. The approximate matrix that minimises the norm of $|A - A'|$ such that the rank of A' is r turns out to be just $A' = U\Sigma'V^T$.

Our system of equations $Ax = b$ may be expressed as $U\Sigma V^T x = b$. Then if we put $z = V^T b$ and $d = U^T b$ we have $\Sigma z = d$. If A is not square, then the *pseudo-inverse* follows. If $m > n$ the system of equations is overdetermined, and z satisfies the equations in the least squares sense, this is the only practical application of SVD we mention here. We can use it to solve least squares problems.

First, we can only have solutions if the element d_i is zero whenever σ_i is zero (or $i > n$). If we have such pairs we have $0 \times z_i = 0$. In this case we have any z_i: instead of just saying that we have no solution, we can now say that though we do not have an exact solution, we at least have a space in which a solution must lie in which we can obtain by

varying the z_i corresponding to zero σ_i and d_i pairs. The range of the possible z_i may be restricted by the nature of the problem. In other words, given a matrix A that is singular, we may find a range of right hand sides that give us possible solution spaces. Given the right circumstances, there might be *some* information to be gained from a singular systems of equations.

Now we come to the condition number of a matrix. We define the norm of a matrix A as the square root of the sum of the squares of the elements, and denote it $|A|$. In "proper" numerical analysis texts, norms are often denoted as $||A||$, and different subscripts denote different kinds of norm, we only mention the one kind here, which is known as the Frobenius norm.

Given a vector b, and an error vector e, then the ratio of the relative error in the solution to the error in b is

$$cond(A) = max\left[\frac{|A^{-1}e|/|A^{-1}b|}{|e|/|b|}\right] = max\left[\frac{|A^{-1}e|}{|e|} \times \frac{|b|}{|A^{-1}b|}\right]. \tag{186}$$

In stating this maximum value (the condition number of A) we suppose that neither b or e are zero, and it can be shown that.

$$cond(A) = |A| \times |A^{-1}|. \tag{187}$$

If we have the eigenvalues or the singular values of A, it can also be shown that

$$cond(A) = \frac{\sigma_{max}}{\sigma_{min}}. \tag{188}$$

If A is a normal matrix, the ratio of the maximum to minimum eigenvalue is also the condition number of A. With this definition, the condition number varies from unity (well conditioned) to infinity. Another result on the norm of a matrix is that

$$|A| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} = \sqrt{tr[A^T A]} = \sqrt{\sum_{i=1}^{min(m,n)} \sigma_i^2}. \tag{189}$$

We recall from §11.1.1 that the trace of a matrix A (or $(tr[A])$ is just the sum of the diagonal elements. We list a few results on the trace of a matrix. The trace is related to the eigenvalues of a matrix in that the kth power of the trace is the sum of the kth power of the eigenvalues for $k = 1, 2, 3, \ldots$. The product of the eigenvalues of A is the determinant of A. The trace of a matrix is invariant under similarity transforms. If A and B are *positive definite* ($x^T Ax > 0$, $x^T Bx > 0$ for all x) then $0 < tr(AB)^k \leq tr(A)^k tr(B)^k$. The trace of a matrix decomposition is greater or equal to the trace of the original matrix, and the result for the condition number follows.

References

[1] H. M. Antia, *Numerical Methods For Scientists and Engineers* (Birkhauser Verlag, Basel, Boston, Berlin, 2002).

[2] M. Schatzmann, *Numerical Analysis - a Mathematical Introduction* (Oxford University Press, Oxford, 2002).

[3] L. N. Trefethen and D. Bau, *Numerical Linear Algebra* (SIAM, Philidelphia, 1997).

[4] G. H. Golub and C. F. V. Loan, *Matrix Computations* (The John Hopkins University Press, Baltimore, Maryland, 1983, 1989, 1996).

[5] J. H. Wilkinson, *The Algebraic Eigenvalue Problem* (Oxford University Press, Oxford, 1965, 2004).