BRUNEL UNIVERSITY

DEPARTMENT OF MATHEMATICAL SCIENCES

SYLLABUS

Module Code:	MA2051S
Module Title:	Numerical Linear Algebra
Semester:	Spring
Level:	2
Lecturer:	Dr. M. K. Warby
Credit value:	10
Prerequisites:	MA1005A (Calculus 1), MA1006A (Calculus 2), MA1001A (Algebra 1), MA1002S (Algebra 2) and MA1051S (Numerical Methods).
Method of delivery:	24 lectures, 10 seminars and 2 revision classes. There are 3 contact hours each week with 3 lectures in each of weeks 1 and 2. In subsequent weeks there will usually be 2 lectures and 1 seminar per week with the revision classes in the last week of the teaching period. About 3 or 4 optional computer laboratory sessions are also likely to be offered to assist with the computer based assignment.
Formative feedback:	Participation in the seminars.
Method of assessment:	Two hour unseen written examination involving a choice of three questions from five (75%) and a computer based assignment (25%).

AIMS:

To provide an introduction to numerical methods for solving linear equations and the eigenvalue problem.

LEARNING OUTCOMES:

On successful completion of the module, students should have the following:

(i) An understanding and ability to implement Gauss elimination and factorization methods for solving systems of linear equations and least squares.

(ii) An ability to manipulate with vector and matrix norms.

(iii) An appreciation of the computational cost and the accuracy attainable by the methods.

(iv) An appreciation of some iterative procedures for solving linear equations and the eigenvalue problem.

MODULE CONTENT:

REVISION OF BASIC LINEAR ALGEBRA:
Vector and matrix notation, basis for $\mathbb{R}^n$ and matrix multiplication. $\underline{x}^T\underline{y}$ and $\underline{x}\,\underline{y}^T$ and the Cauchy Schwarz inequality. Partitioned matrix multiplication. Matrix classification including diagonal, triangular, banded, symmetric, positive definite and orthogonal matrices. Equivalent conditions for a matrix A to be singular. Eigenvalues, eigenvectors and the eigenvalue problem for a symmetric matrix.

DIRECT METHODS FOR SOLVING LINEAR EQUATIONS:
Gauss elimination and LU factorization.
The solution of tri-diagonal systems.
Cholesky factorization for real symmetric positive definite matrices.

VECTOR AND MATRIX NORMS:
The vector norms $\|\underline{x}\|_1$, $\|\underline{x}\|_2$ and $\|\underline{x}\|_\infty$, the subordinate matrix norms $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$ and the Euclidean norm $\|A\|_E$.

THE SENSITIVITY OF LINEAR EQUATIONS AND THE CONDITION NUMBER:
The expansion $(I-E)^{-1} = I + E + E^2 + E^3 + \cdots$ when $\|E\| < 1$.
The matrix condition number $\kappa(A) = \|A\|\,\|A^{-1}\|$.
The problems $(A+\Delta A)(\underline{x}+\Delta\underline{x}) = \underline{b}+\Delta\underline{b}$ and $A\underline{x} = \underline{b}$.

LEAST SQUARES AND THE QR FACTORIZATION:
Least squares and the normal equations. Householder matrices and the QR factorization of a matrix and applications to least squares.

ITERATIVE PROCEDURES FOR LINEAR EQUATIONS:
The iteration $\underline{x}^{(k+1)} = G\underline{x}^{(k)} + \underline{c}$.
Iterative refinement.

THE EIGENVALUE PROBLEM — ESTIMATION:
Gerschgorin's theorem and Rayleigh quotient bounds for symmetric matrices.

THE EIGENVALUE PROBLEM — METHODS:
The Power method and inverse iteration.
A survey of methods for numerically computing eigenvalues.

LECTURE NOTES

A paper version of notes, exercises and solutions to exercises are distributed during the lectures and seminars. These are also available on the www using the URL

`http://www.brunel.ac.uk/~icstmkw/ma2051/`

READING LIST

Core Texts:

No single text that I know of exists which covers all the material of this module at the appropriate level.

Supplementary Texts:

A numerical analysis text

Burden, R. L. and Faires J. D. (2001): Numerical Analysis, 7th Edition, Pacific Grove, Brooks Cole, London. `QA297.B84.`

(*Parts of this book covers many of the topics at about the right level.*)

Books dedicated to numerical linear algebra (generally advanced)

These are written for students at the post graduate level or possibly the final year undergraduate level.

Datta, B. N. (1995): Numerical Linear Algebra and Applications, Brooks/Cole Publishing Company. `QA184.D37.`

Demmel, J. W. (1997): Applied Numerical Linear Algebra, SIAM (Society of Industrial and Applied Mathematics), Philadelphia, PA. USA. `QA184.D455`

Golub, G. H. and Van Loan, C. F. (1996): Matrix Computations, Third Edition, The Johns Hopkins University Press, Baltimore, Maryland, USA. `QA188.G65.`

Trefethen, L. N. and Bau, D. (1997): Numerical Linear Algebra, SIAM (Society of Industrial and Applied Mathematics), Philadelphia, PA. USA. `QA184.T74.`

Some other numerical analysis books

Atkinson, Kendall E. (1993), Elementary numerical analysis, 2nd ed. Wiley, New York, Chichester. `QA297.A83`

(*This is on the reading list of* `MA1051S`. *Chapter 8 is relevant to this module.*)

Wheatley, P. O. (1999), Applied numerical analysis, 6th ed., Addison-Wesley, Reading, Mass., Harlow. `QA297.G47`

(*Chapter 2 is relevant to this module.*)

Many books with titles involving numerical methods or numerical analysis are likely to have chapters relevant to this module.

Linear algebra books

Horn, R. A. and Johnson, C. R. (1985): Matrix Analysis, Cambridge University Press. `QA188.H67.`

(*This is quite an advanced book. It is good as a reference.*)

Anton, H. and Rorres, C. (1994): Elementary Linear Algebra (Applications Version), 7th Edition. Wiley, New York, Chichester. `QA184.A576.`

(*This is one of your first year Algebra texts. It contains information relevant to a few of the sections.*)

Strang, G. (1988): Linear Algebra and its Applications, Third Edition, Harcourt Brace Jovanovich, Inc. Publishers, Orlando, Florida, USA. `QA184.S8`.

MATLAB books

Hanselman, D. and B. Littlefield: Mastering Matlab 6, A Comprehensive Tutorial and Reference, Prentice Hall, New Jersey, 2001, `QA297.H293`.

Higham, D. J. and N.J. Higham: Matlab guide, SIAM (Society for Industrial and Applied Mathematics), Philadelphia, 2000, `QA297.H5217`.

Sigmon, Kermit. and Timothy A. Davis (2002): MATLAB primer. (6th ed.) CRC Press, Boca Raton; London. `QA297.S4787`.

Van Loan, C. F. (1997): Introduction to Scientific Computing. A matrix-vector approach using Matlab. Prentice Hall, NJ. `QA76.9.M35V36`.

There are a number of new Matlab books and newer editions of existing Matlab books becoming available at this time following the recent release of Matlab version 6.

MA2051S Numerical Linear Algebra, Spring 2002 Teaching Schedule

Lecture number (approx)	Topics
1	INTRODUCTION TO THE MODULE
1–3	REVISION OF BASIC LINEAR ALGEBRA: Vector and matrix notation, basis for $\mathbb{R}^n$ and matrix multiplication. $\underline{x}^T\underline{y}$ and $\underline{x}\,\underline{y}^T$ and the Cauchy Schwarz inequality. Partitioned matrix multiplication. Matrix classification including diagonal, triangular, banded, symmetric, positive definite and orthogonal matrices. Equivalent conditions for a matrix A to be singular. Eigenvalues, eigenvectors and the eigenvalue problem for a symmetric matrix.
4–8	DIRECT METHODS FOR SOLVING LINEAR EQUATIONS: Forward and back substitution. Reduction to triangular form by Gauss elimination. Matrix description and the LU factorization. Pivoting. The solution of tri-diagonal systems. Cholesky factorization for real symmetric positive definite matrices.
8–10	VECTOR AND MATRIX NORMS: The vector norm axioms and the norms $\|\underline{x}\|_1$, $\|\underline{x}\|_2$ and $\|\underline{x}\|_\infty$. The matrix norm axioms, subordinate matrix norms and representation of the specific norms $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$. The Euclidean norm $\|A\|_E$. The norm–spectral radius connection.
11–13	THE SENSITIVITY OF LINEAR EQUATIONS AND THE CONDITION NUMBER: The expansion $(I-E)^{-1} = I + E + E^2 + E^3 + \cdots$ when $\|E\| < 1$. The matrix condition number $\kappa(A) = \|A\|\,\|A^{-1}\|$. The problems $(A+\Delta A)(\underline{x}+\Delta\underline{x}) = \underline{b} + \Delta\underline{b}$ and $A\underline{x} = \underline{b}$.
14–16	LEAST SQUARES AND THE QR FACTORIZATION: Least squares and the normal equations. Householder matrices and the QR factorization of a matrix and applications to least squares.
17–18	ITERATIVE PROCEDURES FOR LINEAR EQUATIONS: The iteration $\underline{x}^{(k+1)} = G\underline{x}^{(k)} + \underline{c}$. Iterative refinement.
19	THE EIGENVALUE PROBLEM: A revision of eigenvalues and eigenvectors and a summary of previous parts of the module which have involved eigenvalues.
20–22	THE EIGENVALUE PROBLEM — ESTIMATION: Gerschgorin's theorem, Rayleigh quotient bounds for symmetric matrices and the sensitivity of the symmetric eigenvalue problem.
23–24	THE EIGENVALUE PROBLEM — METHODS: The Power method and the inverse power method. A survey of methods for numerically computing eigenvalues.
24–26	Overview and revision of the module.

Introduction to the module

This module is concerned with describing methods and analysing methods for accurately and efficiently solving systems of linear equations and with determining eigenvalues. Thus there is a mixture of what you would classify as Linear Algebra (the type of material that is covered in Algebra I and Algebra II at level 1) and Numerical Analysis (the type of material covered in Numerical Methods at level 1). We consider next what this involves and we indicate some of the issues involved. We also briefly comment on some of the topics in this module.

When you first meet simultaneous linear equations you are usually just concerned with solving a system of 2 equations in 2 unknowns, e.g.

$$\begin{array}{rcrcl} 2x_1 & + & x_2 & = & 4 \\ -x_1 & + & 3x_2 & = & 5 \end{array} \quad \text{or in matrix form} \quad \begin{pmatrix} 2 & 1 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \end{pmatrix} .$$

You can solve this by hand calculation reasonably quickly by combining the equations in some way to produce a new equations which only involves one of the unknowns. For example, adding 2 times the 2nd equation to the first gives

$$7x_2 = 14 \quad \text{which implies that } x_2 = 2 .$$

Then substituting $x_2 = 2$ into either equation gives $x_1 = 1$. For this 2×2 problem we could have also computed the inverse quickly, i.e.

$$A = \begin{pmatrix} 2 & 1 \\ -1 & 3 \end{pmatrix} \quad \text{implies} \quad \det A = 2 \times 3 - (1)(-1) = 7 \quad \text{and} \quad A^{-1} = \frac{1}{7} \begin{pmatrix} 3 & -1 \\ 1 & 2 \end{pmatrix}$$

and then note that the solution of

$$A\underline{x} = \underline{b} \quad \text{is} \quad \underline{x} = A^{-1}\underline{b} = \frac{1}{7} \begin{pmatrix} 3 & -1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 4 \\ 5 \end{pmatrix} = \frac{1}{7} \begin{pmatrix} 7 \\ 14 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} .$$

The point to note here is that there is not much calculation to do using any of the available methods.

When you advance to 3 equations in 3 unknowns the amount of time that it takes to solve by hand calculation is significantly longer. For example, consider the following system

$$\begin{pmatrix} 1 & 2 & -1 \\ 2 & 0 & 3 \\ -1 & -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 11 \\ 6 \end{pmatrix} .$$

Using row operations you can use the 1st equation to eliminate x_1 from the 2nd and 3rd equations. The operations are

$$\begin{array}{rcl} (\text{equation } 2) & \to & (\text{equation } 2) - 2(\text{equation } 1) \\ (\text{equation } 3) & \to & (\text{equation } 3) + (\text{equation } 1) . \end{array}$$

The equivalent system is

$$\begin{pmatrix} 1 & 2 & -1 \\ 0 & -4 & 5 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 7 \\ 8 \end{pmatrix} .$$

You now only need to consider the new 2nd and 3rd equations. Adding 4 times the 3rd equation to the 2nd equation gives

$$\begin{pmatrix} 1 & 2 & -1 \\ 0 & 0 & 13 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 39 \\ 8 \end{pmatrix} .$$

It is now in a 'reduced form' and much easier to solve. The 2nd equation immediately gives $x_3 = 3$, using the 3rd equation we then get $x_2 = 2$ and then using the 1st equation (which has never changed) we get $x_1 = 1$. The other technique used in the 2×2 case of first computing the inverse A^{-1} and then computing $\underline{x} = A^{-1}\underline{b}$ is longer for this problem and indeed significantly longer for systems of more than 3 unknowns. In numerical linear algebra it is rare that you need to compute A^{-1}. We write $\underline{x} = A^{-1}\underline{b}$ to describe the solution but it should not be considered or interpreted as a good approach in general for actually computing $\underline{x}$ unless there is some very special features of the problem which allows A^{-1} to be computed quickly. In fact the usual numerical techniques for computing A^{-1} is by solving linear systems of the form $A\underline{x} = \underline{b}$ for a number of different right hand side vectors $\underline{b}$.

In the previous 2 paragraphs we considered 2×2 and 3×3 systems. Consider now the 4×4 system

$$\begin{pmatrix} 1 & 0 & 3 & 2 \\ 4 & 3 & 2 & 1 \\ -1 & 1 & 1 & 1 \\ 2 & -1 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 13 \\ 8 \\ 3 \\ 11 \end{pmatrix}.$$

We can solve this by hand by row reduction techniques (i.e. Gauss elimination), which is straight forward but rather tedious, but as the number of unknowns is greater than in the previous cases it will take sometime to implement. Instead, the use of a computer should be considered as packages exist specifically to solve this type of problems. MATLAB (which stands for Matrix Laboratory) was originally designed specifically for this type of problem and has since grown into a package capable of doing many other things as well. Also, if you can program in languages such as Fortran and C then there are libraries of routines freely available for this type of problem. In the case of MATLAB a set of instructions sufficient to solve the problem are as follows.

```
% the coefficients of the linear system

s=[1  0 3 2    13;
   4  3 2 1     8;
  -1  1 1 1     3;
   2 -1 1 2    11];

% set matrix A and rhs b by extracting entries from s

A=s(1:4,1:4);
b=s(1:4,5);

% solve the system and show the solution x

x=A\b;
disp(x');
```

The output produced by the last statement is

```
    1.0000   -1.0000    2.0000    3.0000
```

Thus, as this demonstrates, solving linear systems is a builtin procedure and it is done by the statement `x=A\b` once `A` and `b` have already been set. Note that for 'small' problems such as this you can still relatively easily and quickly check that the solution given is correct by hand calculations and you can also do this in MATLAB by the statement `b-A*x` which should produce the zero vector or something very close to the zero vector.

The examples just presented illustrate the growth in the computation needed to solve larger and larger systems but all the examples are slightly artificial in that the matrix entries are all whole numbers (i.e. integers) and that each component of $\underline{x}$ is also integer. This was specifically arranged so that the hand calculations are not too messy. In general if we start with a matrix A and a vector $\underline{b}$ whose entries are all integer then we can expect the components of the solution $\underline{x}$ to involve cumbersome fractions if all the computations are done exactly as a consequence of all the divisions involved. Exact computation can be done by computer packages such as Mathematica and Maple but the amount of storage required to solve larger and larger systems and the time that this takes soon makes this impractical. Instead, much faster floating point arithmetic is used and we instead get an approximate solution which 'in many cases' is typically accurate to about 15 significant figures or something close to this. This is what MATLAB produces and this is adequate in nearly all practical situations.

In this module we consider a number of aspects relating to methods for solving a system

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

where A is an $n \times n$ non-singular matrix and $\underline{b}$ is an $n \times 1$ vector. In the case of a general dense matrix A (i.e. each entry of A is potentially non-zero and A has no special properties) the established and still preferred technique for solving the system on a computer is some form of Gauss elimination. As we will see, this actually corresponds to a matrix factorization of the form

$$A = LU \quad \text{or} \quad PA = LU$$

where L is unit lower triangular, U is upper triangular and P is a permutation matrix (these will be defined later). This is the way it is implemented in packages on a computer. Note that to store the matrix A on a computer requires space for n^2 entries which on most computers involves $8n^2$ bytes. However, as we will see, to solve the system will involve $\mathcal{O}(n^3)$ floating point operations. (Actually for large n the order of magnitude is $\mathcal{O}(2n^3/3)$.) This thus means that when n is large, increasing n by a factor of 2 requires $2^2 = 4$ times as much storage and $2^3 = 8$ times as many operations. Similarly increasing n by a factor of 4 requires $4^2 = 16$ times as much storage and $4^3 = 64$ times as many operations. Hence in this case we change a problem which takes a few seconds into a larger problem which takes a few minutes and we change a problem which takes a few minutes into a larger problem which takes a few hours. Hence on any given computer there is a limit to the size of system that can be stored in memory, the $\mathcal{O}(n^2)$ storage, and there is a time consideration involved with carrying out the $\mathcal{O}(n^3)$ floating point operations. To get a feel of what is possible with current modest computing equipment below is a table of approximate timings that I have obtained when solving a dense linear system using a library routine written in Fortran and with the programs compiled with full optimization using the `f77` or `g77` Fortran compilers as appropriate.

n	Storage ($8n^2$) in bytes	`molnir` (secs)	Pentium P100 (secs)	Pentium P450 (secs)
400	1.28×10^6	1.5	6.6	≈ 1
800	5.12×10^6	13	57	9
1600	20.48×10^6	114	458	70

The table shows that using a Pentium P450 PC (which was new in August 1999) we can solve the system with $n = 400$ in about 1 second and it takes over a minute when $n = 1600$ and requires over 20 MBytes of memory just to hold the matrix. The next case of $n = 3200$ is just about possible on a PC with 128MBytes of RAM but it is stretching the resources of the PC and you have to wait some time for the solution. Larger, faster and more expensive computers can solve these problems using less time and are capable of solving systems with larger values of n but you do not need to increase n by too much to go beyond the capability of any computer currently in existence. When $n = 10^4$ this requires 8×10^8 bytes of storage (which is less than one GigaByte) and systems of this size can now be solved on top of the range PC's but $n = 10^5$ requires 8×10^{10} bytes of storage (i.e. 80 GigaBytes) and this is I believe just beyond the capacity of todays multi-million dollar super computers.

An outcome of the previous discussion is that solving a general linear system $A\underline{x} = \underline{b}$ when n is large involves n^2 storage and $\mathcal{O}(n^3)$ floating point operations on a computer and that $n = 10^5$ is beyond what can be done anywhere at the moment. In many practical situations however the linear systems that need to be solved often involve sparse matrices (i.e. most of the entries are 0) and may have other special features such as being banded and/or symmetric. In these cases systems with much larger values of n can be solved and this is done in a number of large computational simulations in, for example, models used in weather prediction. The solution techniques used in these cases exploit the sparse structure and the actual methods used are usually iterative. Some of the work in this area is current research and it is at a level which is far beyond what will be covered in this level 2 module. However we will consider certain special situations. We will consider the special case of a tri-diagonal linear system which involves only $\mathcal{O}(n)$ storage and $\mathcal{O}(n)$ floating point operations. We will also consider symmetry and a property known as positive definite (we get a Cholesky factorization in this case), and we give a brief introduction to iterative methods for solving $A\underline{x} = \underline{b}$. Further study of iterative methods becomes problem dependent and it is more appropriate in Level 3 modules or in a Final Year Project and indeed I have had in the past several final year projects in this area.

As has already been stated, solving $A\underline{x} = \underline{b}$ on a computer with floating point arithmetic only gives an approximate solution although this will usually be accurate to about 15 significant figures. This will be the case provided the matrix A is not 'nearly singular'. In this module we will quantify what this means in terms of a condition number $\kappa(A)$ which is defined using matrix norms and which is related to the eigenvalues of A or of a related matrix. This module involves a section on norms for vectors and matrices. As we will see, norms is concerned with size and it enables us to say when one vector is close to another vector and when one matrix is close to another matrix. In the the context of linear systems it enables us to bound the size of $\Delta\underline{x}$ in terms of ΔA and $\Delta\underline{b}$ when we have the 2 linear systems

$$A\underline{x} = \underline{b} \quad \text{and} \quad (A + \Delta A)(\underline{x} + \Delta\underline{x}) = \underline{b} + \Delta\underline{b}\ .$$

What we want to know here is the effect on the solution of changing the matrix and/or changing the right hand side vector.

When the problem $A\underline{x} = \underline{b}$ involves a matrix which is not a square matrix then we either have too few equations or too many equations, i.e. the system is either under-determined or over-determined. In these situations a unique solution never exists and what is done instead is to consider least squares solutions. In this module we consider only the over-determined case and a least squares solution which is a vector $\underline{x}$ which minimises $\|A\underline{x} - \underline{b}\|_2$ (this notation will be introduced in the norms section). Roughly, this involves attempting to most nearly solve all the equations in the sense of least squares. The vector $\underline{x}$ which gives the minimum will in general be such that $A\underline{x} \neq \underline{b}$ but it will be such that

$$A^T(A\underline{x} - \underline{b}) = \underline{0}, \quad \text{i.e.} \quad A^TA\underline{x} = A^T\underline{b}$$

which is a linear system with the square matrix $A^T A$. These equations are known as the normal equations. As we will see, least squares problems can also be solved by another technique involving constructing a QR factorization of the matrix A under consideration where Q is an orthogonal matrix and R is an upper triangular matrix. This will be introduced at this stage.

Throughout the module eigenvalues and eigenvectors appear when methods are analysed although the quantities needed are often only the extreme eigenvalues (i.e. the largest and the smallest) and these are often only for symmetric matrices. For example, if you have a square matrix A and you have some set up which involves the sequence A, A^2, A^3, $\cdots$ then the interest is in what happens to A^n as $n \to \infty$. As with a geometric progression of numbers r, r^2, r^3, $\cdots$ it depends on whether something is less than 1 or greater than 1. In the case of numbers $r^n \to 0$ as $n \to \infty$ if and only if $|r| < 1$. For matrices the corresponding result is that

$$A^n \to \text{zero matrix} \quad \text{as } n \to \infty$$

if and only if the spectral radius $\rho(A)$ of A is less than 1 where

$$\rho(A) = \max\{|\lambda| : \quad \lambda \text{ is an eigenvalue of } A\} .$$

This will appear when iterative methods for linear systems are briefly considered and similar reasoning is used when the power method is described for computing the dominant eigenvalue. Before this is done, some techniques are described for estimating eigenvalues (Gerschgorin's theorem and Rayleigh quotients).

A collection of other points relevant to the module

MA2051S, MA251S

Please note that the university has, since the summer of 1997, made all module codes have 4 digits. Thus MA2051S was previously labelled as MA251S.

Remarks about the references

In the reference list I have not given a 'core text' but I have instead just listed books that I have used at some stage in preparing these notes. Of the supplementary texts, part of the book by Burden and Faires covers much of the material of this module at about the appropriate level. Many other similar numerical analysis texts may also do this. (I have listed two such books.) The books that I have listed as 'dedicated to numerical linear algebra' (and which in most cases have this in the title) are generally presented at a reasonably advanced level.

The books on MATLAB are given just to indicate possible texts that exists that describe MATLAB.

There has recently been several new books on the topic of numerical linear algebra and thus there may be some very suitable texts that I have missed which would be good to add to the list. Please let me know if you find such books. I have as yet not however met anyone teaching similar material at other universities who can suggest a suitable core text for teaching this material for students at level two.

Some Addresses and dates

`MA2051S` Lecturer: Dr. Michael Warby

Room: 301 of the Mathematics building

Tel: +44 1895 274000 ext. 2169

Fax: +44 1895 203303 (Maths Department)

Email: `mike.warby@brunel.ac.uk`

Home Page URL: `http://www.brunel.ac.uk/~icstmkw/home.html`
Notes for this module (as gzipped PostScript files) can be accessed using the URL:
`http://www.brunel.ac.uk/~icstmkw/2051/`

```
                                2002

        January                February                 March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
       1  2  3  4  5                   1  2                   1  2
 6  7  8  9 10 11 12    3  4  5  6  7  8  9    3  4  5  6  7  8  9
13 14 15 16 17 18 19   10 11 12 13 14 15 16   10 11 12 13 14 15 16
20 21 22 23 24 25 26   17 18 19 20 21 22 23   17 18 19 20 21 22 23
27 28 29 30 31         24 25 26 27 28         24 25 26 27 28 29 30
                                              31
         April                   May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6             1  2  3  4                      1
 7  8  9 10 11 12 13    5  6  7  8  9 10 11    2  3  4  5  6  7  8
14 15 16 17 18 19 20   12 13 14 15 16 17 18    9 10 11 12 13 14 15
21 22 23 24 25 26 27   19 20 21 22 23 24 25   16 17 18 19 20 21 22
28 29 30               26 27 28 29 30 31      23 24 25 26 27 28 29
                                              30
```

Semester: Monday 4 February 2002 to Friday 22 March 2002 (7 weeks) and Thursday 11 April to Wednesday 15 May 2002 (5 weeks)

Easter semester break: Monday 25 March to Wednesday 10 April 2002 (2 weeks and 3 days)

May Day holiday : Monday 6 May 2002
Spring bank holiday/Golden Jubilee: Monday 3/Tuesday 4 June 2002.

Revision week: Thursday 16 May 2002 to Wednesday 22 May 2002.

Examination period: Thursday 23 May to Friday 7 June 2002.

I will give out the assignment before the Easter break. You will have the Easter break in addition to at least 3 weeks of the semester to do it.

1 Some Background Theory

1.1 Notation

$\mathbb{R}$	denotes the set of real numbers
$\mathbb{R}^n$	denotes the set of $n \times 1$ real (column) vectors
$\mathbb{R}^{m\times n}$	denotes the set of $m \times n$ real matrices

$\mathbb{C}$, $\mathbb{C}^n$ and $\mathbb{C}^{m\times n}$ similarly refer to complex numbers, vectors and matrices.

For **column vectors** the notation $\underline{x} = (x_i) \in \mathbb{R}^n$ is shorthand for a real column vector $\underline{x}$ with n rows and with the entry in row i being x_i, i.e.

$$\underline{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} .$$

If $\underline{x}$ is a **column vector** then $\underline{x}^T = (x_1, \cdots, x_n)$ is a **row vector**. (Note that some books use a prime ' or a superscript ^{t} to denote transpose. MATLAB uses '.)

For **matrices** the notation $A = (a_{ij}) \in \mathbb{R}^{m\times n}$ is shorthand for a real matrix with m rows and n columns with the ij entry a_{ij}, i.e.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} .$$

In this module we will frequently use the following.

The **zero vector** $\underline{0}$. The size of $\underline{0}$ will depend on the context.

The $n \times n$ **identity matrix**

$$I = (\delta_{ij}) = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{n\times n} .$$

where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

The identity matrix is a special case of a **diagonal matrix**, i.e. a matrix with all non-diagonal entries equal to zero.

The **standard base vectors** for $\mathbb{R}^n$ will be denoted by $\underline{e}_1, \cdots, \underline{e}_n$ where

$$\underline{e}_i = i\text{th column of } I .$$

1.2 Basic matrix operations

1.2.1 Addition and scalar multiplications

If $A = (a_{ij})$ and $B = (b_{ij})$ are of the same size and $\alpha \in \mathbb{R}$ then

$$A + B = (a_{ij} + b_{ij}), \quad \alpha A = (\alpha a_{ij}) \ .$$

1.2.2 Matrix multiplication

For the matrix product AB to be defined we require that

$$\text{number of columns of } A = \text{number of rows of } B \ .$$

If $A = (a_{ij}) \in \mathbb{R}^{m\times p}$ and $B = (b_{ij}) \in \mathbb{R}^{p\times n}$ then $C = (c_{ij}) = AB \in \mathbb{R}^{m\times n}$ with

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} = (a_{i1}, \cdots, a_{ip}) \begin{pmatrix} b_{1j} \\ \vdots \\ b_{pj} \end{pmatrix} = (i\text{th row of } A) \times (j\text{th column of } B) \ .$$

Matrix multiplication is **associative**, i.e. $A(BC) = (AB)C$.

Matrix multiplication is not in general commutative, i.e. $AB \neq BA$ in general.

There are special situations in which matrices do commute, e.g. diagonal matrices and in situations of the form $A^2A^3 = A^3A^2 = A^5$.

1.2.3 Special cases of matrix multiplication

1. If $\underline{x}, \underline{y} \in \mathbb{R}^n$ then we can form the product

$$\underline{x}^T \underline{y} = \underline{y}^T \underline{x},$$

which is a scalar. This is the **inner product** of the vectors $\underline{x}$ and $\underline{y}$. It is the same as what is referred to as the dot product or scalar product which is the jargon more commonly used in modules in Vector Calculus or Mechanics.

2. If $\underline{x}, \underline{y} \in \mathbb{R}^n$ then we can form the product

$$A = \underline{x}\,\underline{y}^T = (x_i y_j) = \begin{pmatrix} x_1y_1 & \cdots & x_1y_n \\ \vdots & \cdots & \vdots \\ x_ny_1 & \cdots & x_ny_n \end{pmatrix} \in \mathbb{R}^{n\times n} \ .$$

This is the **outer product** of the vectors $\underline{x}$ and $\underline{y}$. A matrix of the form $A = \underline{x}\,\underline{y}^T$ with $\underline{x} \neq \underline{0}, \underline{y} \neq \underline{0}$ is a **rank 1 matrix** and it is a type of matrix which appears a number of times in this module. It is often used in situations of the form

$$A\underline{u} = (\underline{x}\,\underline{y}^T)\underline{u} = \underline{x}(\underline{y}^T\underline{u}) = (\underline{y}^T\underline{u})\underline{x},$$

where the last 2 equals are respectively as a consequence of the associative property of matrix multiplication and the definition of a scalar times a vector. The important point here is that whatever the vectors $\underline{y}$ and $\underline{u}$ are, $\underline{y}^T\underline{u} \in \mathbb{R}$ is just a scalar and hence for all vectors $\underline{u}$ the vector $A\underline{u}$ will just be a multiple of the vector $\underline{x}$. Another way of saying this is that $A\underline{u}$ only has **one direction** of more abstractly this can be written as

$$\mathrm{span}\{A\underline{u} : \underline{u} \in \mathbb{R}^n\} = \mathrm{span}\{\underline{x}\} .$$

$A\underline{u}$ only having one direction is the reason for the name rank one matrix. (The term rank was covered in MA1002S Algebra 2.) As a final remark concerning rank 1 matrices, it may be tempting when you first meet a matrix of the form $\underline{x}\,\underline{y}^T$ to expand it out to display it in matrix form. It is however usually neater and simpler to leave the matrix in the form $\underline{x}\,\underline{y}^T$ for the type of manipulation that is done in this module.

1.3 Partitioned matrix representation and matrix multiplication

In some situations it helps to write

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m\times n}$$

in different ways by grouping together the entries in various ways. It is also valid to multiply together matrices written in a partitioned form provided all the matrix multiplications are defined, i.e. the numbers of columns of the first matrix in any product must be the same as the number of rows in the second matrix. We consider this now in a few cases.

If $\underline{a}_1, \cdots, \underline{a}_n \in \mathbb{R}^m$ denote the n columns of A then

$$A = (\underline{a}_1, \cdots, \underline{a}_n)$$

is a partitioned representation of A and it is valid to write

$$A\underline{x} = (\underline{a}_1, \cdots, \underline{a}_n) \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \underline{a}_1 x_1 + \cdots + \underline{a}_n x_n = x_1\underline{a}_1 + \cdots + x_n\underline{a}_n .$$

That is $A\underline{x}$ is a vector and the vector is a linear combination of the columns of A. A special case of this is when $\underline{x} = \underline{e}_i = i$th column of I. We have

$$A\underline{e}_i = (\underline{a}_1, \cdots, \underline{a}_i, \cdots, \underline{a}_n) \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \underline{a}_i = i\text{th column of } A .$$

We consider again the situation when $A \in \mathbb{R}^{m,p}$, $B \in \mathbb{R}^{p,n}$ so that $C = AB \in \mathbb{R}^{m,n}$. If we represent B in terms of its columns $\underline{b}_1, \cdots, \underline{b}_n$, i.e.

$$B = (\underline{b}_1, \cdots, \underline{b}_n)$$

then

$$C = AB = A(\underline{b}_1, \cdots, \underline{b}_n) = (A\underline{b}_1, \cdots, A\underline{b}_n) .$$

That is the vector $A\underline{b}_i$ is the ith column of C. There are also other ways of representing this product of two matrices. If $p > 1$ and we decompose this as $p = p_1 + p_2$ where $p_1 \geq 1$ and $p_2 \geq 1$ (for any possible p_1 and p_2 satisfying this relation) then

$$c_{ij} = (C)_{ij} = (AB)_{ij} = \sum_{k=1}^{p} a_{ik} bkj = \left(\sum_{k=1}^{p_1} a_{ik} b_{kj} \right) + \left(\sum_{k=p_1+1}^{p} a_{ik} b_{kj} \right) .$$

Each bracketed term is itself the i, j entry of the product of a submatrix of A with a submatrix of B. As a consequence of this the following is valid. If

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

where $A_{11}, A_{12}, A_{21}, A_{22}$ are submatrices of A and similarly $B_{11}, B_{12}, B_{21}, B_{22}$ are submatrices of B then

$$C = AB = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

provided each of the multiplications of the smaller matrices is defined.

1.4 Euclidean length and orthogonality of vectors

If $\underline{x} = (x_i) \in \mathbb{R}^n$ then $(x_1^2 + x_2^2 + \cdots + x_n^2)^{1/2} = (\underline{x}^T \underline{x})^{1/2}$ is the **Euclidean length** of the vector $\underline{x}$. The Euclidean length is also called the 2-norm — to be covered later in the module.

A vector $\underline{x}$ is said to be **normalised** if it has **Euclidean length=1**, i.e. $\underline{x}^T \underline{x} = 1$.

Vectors $\underline{x}$ and $\underline{y}$ are **orthogonal** if

$$\underline{x}^T \underline{y} = \underline{y}^T \underline{x} = \sum_{1}^{n} x_i y_i = 0.$$

1.5 The size of $\underline{x}^T \underline{y}$ — the Cauchy-Schwarz inequality

$$(\underline{x}^T \underline{y})^2 \leq (\underline{x}^T \underline{x})(\underline{y}^T \underline{y})$$

<u>Proof</u>

The proof involves the following trick. Since all vectors have non-negative length it follows that for all real numbers t

$$0 \leq (\underline{x} + t\underline{y})^T (\underline{x} + t\underline{y}) = (\underline{x}^T \underline{x}) + 2t(\underline{x}^T \underline{y}) + t^2 (\underline{y}^T \underline{y}) = c + bt + at^2,$$

where $c = \underline{x}^T \underline{x}$, $b = 2\underline{x}^T \underline{y}$ and $a = \underline{y}^T \underline{y}$. As the quadratic cannot be negative it can have at most 1 real root which will be a double root. Thus the quadratic has 1 real double root or 2 complex conjugate root which corresponds to the condition

$$0 \geq b^2 - 4ac = (2\underline{x}^T \underline{y})^2 - 4(\underline{x}^T \underline{x})(\underline{y}^T \underline{y}), \quad \text{i.e. } (\underline{x}^T \underline{y})^2 \leq (\underline{x}^T \underline{x})(\underline{y}^T \underline{y}).$$

1.6 Special types of matrix

1.6.1 Diagonal matrix

In the case of square matrices

$$D = \mathrm{diag}\{d_1, \cdots, d_n\} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix}.$$

That is

$$(D)_{ij} = 0 \quad \text{for all } i \neq j$$

which defines a diagonal matrix of any shape.

1.6.2 Lower triangular matrix

In the case of square matrices

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

That is $A = (a_{ij})$ with $a_{ij} = 0$ for $j > i$. (This defines a lower triangular matrix of any shape.) A matrix A is said to be **strictly lower triangular** if it is lower triangular and all diagonal entries are zero. A matrix A is said to be **unit lower triangular** if it is lower triangular and all diagonal entries are equal to 1.

1.6.3 Upper triangular matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & & a_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}.$$

That is $A = (a_{ij})$ with $a_{ij} = 0$ for $i > j$. (This defines an upper triangular matrix of any shape.) A matrix A is said to be **strictly upper triangular** if it is upper triangular and all diagonal entries are zero. A matrix A is said to be **unit upper triangular** if it is upper triangular and all diagonal entries are equal to 1.

1.6.4 Tri-diagonal matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & & & 0 \\ a_{21} & a_{22} & a_{23} & & & \vdots \\ \vdots & a_{32} & a_{33} & a_{34} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & & & a_{n,n-1} & a_{nn} \end{pmatrix}.$$

That is $A = (a_{ij})$ with $a_{ij} = 0$ for $|i - j| > 1$. The entries $a_{i,i+1}$ are the **super-diagonal** entries. The entries $a_{i+1,i}$ are the **sub-diagonal** entries.

1.6.5 Banded matrix

This can be similarly defined.

1.6.6 Symmetric matrix

A matrix A is **symmetric** if $A^T = A$, that is $a_{ij} = a_{ji}$ for all i and j.

1.6.7 Positive definite matrix

A matrix A is **positive definite** if $\underline{x}^T A\underline{x} > 0$ for all vectors $\underline{x} \neq \underline{0}$.

(Observe that in components $(A\underline{x})_i = \sum_{j=1}^n a_{ij}x_j$ so that

$$\underline{x}^T A\underline{x} = \sum_{i=1}^n x_i(A\underline{x})_i = \sum_{i=1}^n \sum_{j=1}^n a_{ij}x_i x_j,$$

a **quadratic form** in $\mathbb{R}^n$)

1.6.8 Positive semi-definite matrix

A matrix A is **positive semi-definite** if $\underline{x}^T A\underline{x} \geq 0$ for all vectors $\underline{x}$.

Negative definite and **negative semi-definite** can be similarly defined.

1.6.9 Sparse and dense matrices

These are not precisely defined. Roughly speaking a matrix is **sparse** if 'most' of its entries are zero and a matrix is **dense** if 'most' of its entries are non-zero.

1.6.10 Orthogonal matrix

A real matrix Q is **orthogonal** if

$$Q^T Q = QQ^T = I, \quad \text{i.e. } Q^{-1} = Q^T \ .$$

The condition $Q^T Q = I$ means that the columns of Q are normalised and mutually orthogonal. The columns of Q give an orthonormal basis for $\mathbb{R}^n$. That is, if we write Q in the partitioned form $Q = (\underline{q}_1, \cdots, \underline{q}_n)$ then the vectors $\underline{q}_1, \cdots, \underline{q}_n$ are an orthonormal basis for $\mathbb{R}^n$ in the same way as is the standard basis $\underline{e}_1, \cdots, \underline{e}_n$ which are the columns of the identity matrix I.

1.7 Singular and non-singular (square) matrices

There are many equivalent ways of stating that a matrix A is singular. A condition which will be used several times in this module is as follows.

A is **singular** if there exists $\underline{x} \neq \underline{0}$ such that

$$A\underline{x} = \underline{0},$$

i.e. the homogeneous problem $A\underline{x} = \underline{0}$ has a non-trivial solution.

A is **non-singular** if the only solution of

$$A\underline{x} = \underline{0}$$

is $\underline{x} = \underline{0}$.

We will also use the condition that A is singular when $\det A = 0$ and non-singular when $\det A \neq 0$. Another characterization is that A is singular if it has an eigenvalue equal to 0 and non-singular if all eigenvalues are non-zero.

When A is singular we cannot solve uniquely the linear equations $A\underline{x} = \underline{b}$. When inexact floating point arithmetic is used (as in usual computer calculations) there is difficulty in solving $A\underline{x} = \underline{b}$ when A is 'nearly singular'. Such a matrix is often described as 'ill-conditioned' or 'badly-conditioned'. We will consider this aspect more precisely in this module.

1.8 Some remarks about manipulations with the transpose operation

When you transpose a matrix the rows and columns interchange, that is the ith row of A becomes the ith column of A^T and in terms of components

$$(A)_{ij} = (A^T)_{ji} \ .$$

In this module there are a number of situations where the transpose of the product of 2 or more matrices appears during algebraic manipulations. The important result relating to this is that

$$(AB)^T = B^T A^T$$

and for a more general product we have

$$(A_1 A_2 \cdots A_{m-1} A_m)^T = A_m^T A_{m-1}^T \cdots A_2^T A_1^T$$

that is the transpose of a product is the product of the transposes in *reverse* order. We have already met this in the special case of the inner product of two vectors. We have

$$\underline{x}^T \underline{y} = \sum_{i=1}^{n} x_i y_i = \underline{y}^T \underline{x} \ .$$

In the case of showing that $(AB)^T = B^T A^T$ let $\underline{a}_j^T$ denote the jth row of A and let $\underline{b}_i$ denotes the ith column of B. Then

$$\begin{aligned} i, j \text{ entry of } (AB)^T &= j, i \text{ entry of } AB \\ &= (j\text{th row of } A)(i\text{th column of } B) = \underline{a}_j^T \underline{b}_i \ . \end{aligned}$$

Similarly,

$$\begin{aligned} i, j \text{ entry of } B^T A^T &= (i\text{th row of } B^T)(j\text{th column of } A^T) \\ &= (i\text{th column of } B)^T (j\text{th row of } A)^T = \underline{b}_i^T \underline{a}_j \end{aligned}$$

and the result is proved since $\underline{a}_j^T \underline{b}_i = \underline{b}_i^T \underline{a}_j$.

One immediate result about the transpose operation is that the transpose of a transpose leaves the matrix unchanged, i.e.

$$(A^T)^T = A \ .$$

As has already been stated, the transpose of a symmetric matrix is the same matrix (by definition). Evidently also the transpose of a scalar (i.e. 1×1 matrix) is the same quantity and this will appear a number of times. For example,

$$\underline{x}^T \underline{y} = (\underline{x}^T \underline{y})^T = \underline{y}^T \underline{x}, \quad \underline{x}^T A \underline{y} = (\underline{x}^T A \underline{y})^T = \underline{y}^T A^T \underline{x} \quad \text{and} \quad (\underline{x}+\underline{y})^T(\underline{x}+\underline{y}) = \underline{x}^T \underline{x} + 2\underline{x}^T \underline{y} + \underline{y}^T \underline{y} \quad \text{etc.}$$

Manipulations like these are used a number of times in this module.

As has just been shown, the transpose of a product is the product of the transposes in reverse order. There is an analogous result for inverses. When we have matrices which are invertible we have

$$(AB)^{-1} = B^{-1}A^{-1}, \quad (A_1 A_2 \cdots A_m)^{-1} = A_m^{-1} \cdots A_2^{-1} A_1^{-1}.$$

1.9 Eigenvalues and Eigenvectors

A vector $\underline{v} \neq \underline{0}$ is an **eigenvector** of a matrix A with **eigenvalue** λ if

$$A\underline{v} = \lambda \underline{v} \ .$$

Thus the homogeneous problem

$$(\lambda I - A)\underline{v} = \underline{0}$$

has a non-trivial solution and the matrix $\lambda I - A$ is singular. In this context this condition is usually expressed as

$$\det(\lambda I - A) = 0,$$

which is the **characteristic equation**.

$$p_A(t) = \det(tI - A)$$

is known as the **characteristic polynomial**. For an $n \times n$ matrix A the polynomial $p_A(t)$ has degree n and by the fundamental theorem of algebra[1] we know that it has exactly n (complex) roots provided multiple roots are counted appropriately. Hence an $n \times n$ matrix has exactly n eigenvalues which may be complex. In the case of a real matrix the polynomial $p_A(t)$ has real coefficients which implies that the non-real eigenvalues must occur in complex conjugate pairs. As the eigenvalues $\lambda_1, \cdots, \lambda_n$ are the roots of the characteristic polynomial we can factor it to give the representation

$$\begin{aligned} P_A(t) &= \det(tI - A) = (t - \lambda_1) \cdots (t - \lambda_n) \\ &= (-1)^n \det(A - tI) = (-1)^n (\lambda_1 - t) \cdots (\lambda_n - t) \ . \end{aligned}$$

By setting $t = 0$ in the last formula gives the result

$$\det A = \lambda_1 \lambda_2 \cdots \lambda_n = \text{product of the eigenvalues} \ .$$

[1] The fundamental theorem of algebra is a result of complex analysis which you may have covered in the Complex Variable module.

Another useful thing that can be deduced quickly from the eigenvalue/eigenvector relation is an expression for the eigenvalue λ in terms the matrix A and the eigenvector $\underline{v}$. In the case that λ is real so that $\underline{v} \neq \underline{0}$ is real and $\underline{v}^T\underline{v} > 0$ we have

$$\lambda\underline{v} = A\underline{v}, \quad \lambda\underline{v}^T\underline{v} = \underline{v}^T A\underline{v}, \quad \text{i.e. } \lambda = \frac{\underline{v}^T A\underline{v}}{\underline{v}^T\underline{v}} .$$

The last expression is known as the Rayleigh quotient of the matrix A and the vector $\underline{v}$ which is something that will be considered towards the end of the module. Observe that the numerator of the expression, i.e. $\underline{v}^T A\underline{v}$, is precisely the term used when a positive definite matrix is considered. Hence we deduce immediately that when A is positive definite then we have $\lambda > 0$. As is shown in the next subsection, all the eigenvalues of a symmetric matrix are real. Thus a positive definite symmetric matrix is characterised by the property that it has positive eigenvalues and as the determinant of a matrix is the product of its eigenvalues it also has a positive determinant. We will make use of this property when we consider Cholesky factorizations.

1.10 A basis and a basis consisting of eigenvectors

For the n-dimensional space $\mathbb{R}^n$ (or $\mathbb{C}^n$) there is the standard orthonormal basis $\underline{e}_1, \cdots, \underline{e}_n$. Thus when we write $\underline{x} = (x_i) \in \mathbb{R}^n$ we mean

$$\underline{x} = \sum_1^n x_i\underline{e}_i .$$

For analysing many methods involving a matrix A it is often convenient to use a basis consisting of eigenvectors of the matrix when this is possible. If $\lambda_1, \cdots, \lambda_n$ are the eigenvalues of A and $\underline{v}_1, \cdots, \underline{v}_n$ are associated eigenvectors (which may be complex) then the question is when can $\underline{v}_1, \cdots, \underline{v}_n$ be chosen so that it is a basis? There is a precise (technical) answer to this question which is that this is possible when the geometric multiplicity of each distinct eigenvalue is the same as the algebraic multiplicity. To be much less precise the eigenvectors $\underline{v}_1, \cdots, \underline{v}_n$ are a basis for $\mathbb{C}^n$ in 'nearly all cases' in the sense that A must have 'special properties' for it not to be a basis. The most satisfactory situation is when we restrict attention to the important case of real symmetric matrices as in this case the eigenvalues are all real and the eigenvectors can be chosen to be orthonormal. As these results are important to this module we state them as follows.

1.11 Eigenvalues and eigenvectors of real symmetric matrices

1. **All the eigenvalues are real**
   This can be proved as follows. Let $\underline{v} \neq \underline{0}$ be an eigenvector of A with corresponding eigenvalue λ which we must suppose at the outset can be complex. Now if we recall some properties of the complex conjugate, we have

   $$z = x + iy, \quad x, y \in \mathbb{R}, \quad \overline{z} = x - iy \quad \text{and} \quad z\overline{z} = x^2 + y^2 = |z|^2 .$$

   Hence if $\underline{v} = (v_i)$ and we define $\overline{\underline{v}} = (\overline{v_i})$, i.e. the vector of the complex conjugates, then premultiplying

   $$A\underline{v} = \lambda\underline{v}$$

   by the row vector $\overline{\underline{v}}^T$ gives

   $$(\overline{\underline{v}}^T A\underline{v}) = \lambda(\overline{\underline{v}}^T\underline{v}) \quad \text{i.e.} \quad \lambda = \frac{\overline{\underline{v}}^T A\underline{v}}{\overline{\underline{v}}^T\underline{v}}$$

and this expression involves the positive denominator

$$\underline{\overline{v}}^T \underline{v} = \sum_1^n \overline{v_i} v_i = \sum_1^n |v_i|^2 > 0 \ .$$

To complete the proof we must show that the numerator of the fraction is real. By using the properties of the transpose and that $A^T = A$ we have

$$\underline{\overline{v}}^T A \underline{v} = (\underline{\overline{v}}^T A \underline{v})^T = \underline{v}^T A^T \underline{\overline{v}} = \underline{v}^T A \underline{\overline{v}} \ .$$

Now the complex conjugate of the product of two numbers is the same as the product of the complex conjugates which can be verified quickly by considering the polar representation of a complex number. If $z_1 = r_1 \mathrm{e}^{i\theta_1}$ and $z_2 = r_2 \mathrm{e}^{i\theta_2}$ then $\overline{z_1} = r_1 \mathrm{e}^{-i\theta_1}$ and $\overline{z_2} = r_2 \mathrm{e}^{-i\theta_2}$ and $\overline{(z_1 z_2)} = (\overline{z_1})(\overline{z_2}) = r_1 r_2 \mathrm{e}^{-i(\theta_1+\theta_2)}$. This extends to the sum of products of complex numbers and thus

$$\overline{(\underline{\overline{v}}^T A \underline{v})} = \underline{v}^T A \underline{\overline{v}},$$

i.e. $\underline{\overline{v}}^T A \underline{v}$ is equal to its complex conjugate and it is hence real.

As λ is real the eigenvector $\underline{v}$ is also real as a Gauss elimination type process can be used to obtain the vector and all the operations used are done with real numbers.

2. **A real symmetric $n \times n$ matrix has n orthonormal eigenvectors**
The full proof is more appropriate in a Linear Algebra module and will not be given here but we will now show that it is relatively straight forward to show that eigenvectors corresponding to distinct eigenvalues must be orthogonal. If we have

$$A\underline{v}_i = \lambda_i \underline{v}_i, \quad A\underline{v}_j = \lambda_j \underline{v}_j, \quad \text{with } \lambda_i \neq \lambda_j$$

then by the symmetry of A we obtain

$$\underline{v}_j^T A \underline{v}_i = (\underline{v}_j^T A \underline{v}_i)^T = \underline{v}_i^T A^T \underline{v}_j = \underline{v}_i^T A \underline{v}_j \ .$$

The eigenvector/eigenvalue relations then imply that

$$\lambda_i \underline{v}_j^T \underline{v}_i = \underline{v}_j^T A \underline{v}_i = \underline{v}_i^T A \underline{v}_j = \lambda_j \underline{v}_i^T \underline{v}_j = \lambda_j \underline{v}_j^T \underline{v}_i, \quad \text{i.e. } (\lambda_i - \lambda_j) \underline{v}_j^T \underline{v}_i = 0 \ .$$

Since $\lambda_i \neq \lambda_j$ we must have $\underline{v}_i^T \underline{v}_j = 0$.

1.12 Remarks about orthonormal vectors

If $\underline{v}_1, \underline{v}_2, \cdots, \underline{v}_n$ is a basis for $\mathbb{R}^n$ then any vectors $\underline{x} \in \mathbb{R}^n$ and $\underline{y} \in \mathbb{R}^n$ can be written in the form

$$\underline{x} = \sum_1^n c_i \underline{v}_i \quad \text{and} \quad \underline{y} = \sum_1^n d_i \underline{v}_i$$

for some real coefficients $c_1, \cdots, c_n$ and $d_1, \cdots, d_n$. For a general basis we have to solve a linear system to actually determine these coefficients. However when $\underline{v}_1, \cdots, \underline{v}_n$ are orthonormal things simplify considerably since

$$\underline{x}^T \underline{v}_j = \underline{v}_j^T \underline{x} = \underline{v}_j^T \left(\sum_1^n c_i \underline{v}_i \right) = \sum_1^n c_i \underline{v}_j^T \underline{v}_i = c_j \ .$$

That is the coefficients are simply $c_i = \underline{x}^T \underline{v}_i = \underline{v}_i^T \underline{x}$. Similarly $d_i = \underline{v}_i^T \underline{y}$.

Another simplification that we obtain with orthonormal basis occurs when we compute inner products. We have

$$\underline{x}^T \underline{y} = \sum_{i=1}^{n} \sum_{j=1}^{n} c_i d_j \underline{v}_i^T \underline{v}_j = \sum_{i=1}^{n} c_i d_i \ .$$

Orthonormal basis, or more generally orthogonal basis, are used in many areas of mathematics, compare for example Fourier series.

1.13 Some key points of the section

The following are a collection of points to note from this introductory section.

1. Take care to distinguish between column vectors and row vectors in expressions.

2. Note that the column vectors $\underline{e}_1, \underline{e}_2, \cdots, \underline{e}_n$ are the columns of the identity matrix I and are the standard base vectors for $\mathbb{R}^n$.

3. You need to become familiar with the notation, the jargon and the types of manipulations involved in the module. In the case of the notation and the manipulation observe that if $\alpha \in \mathbb{R}$ (α is a scalar), $\underline{x}, \underline{y} \in \mathbb{R}^n$ ($\underline{x}$ and $\underline{y}$ are column vectors) and $A \in \mathbb{R}^{n\times n}$ (A is a square matrix) then

   α, $\underline{x}^T \underline{y} = \underline{y}^T \underline{x}$, $\underline{x}^T A \underline{x} = \underline{x}^T A^T \underline{x}$ and $\underline{x}^T A \underline{y} = \underline{y}^T A^T \underline{x}$ are all scalars.

   $\underline{x}$, $\underline{y}$, $A\underline{x}$, $A^T\underline{x}$, $A\underline{y}$ and $A^T\underline{y}$ are all column vectors.

   A, $\underline{x}\,\underline{y}^T$, $\underline{y}\,\underline{x}^T$ are all $n \times n$ matrices.

4. Let $A \in \mathbb{R}^{n\times n}$.

   $$A \text{ is non-singular if and only if } \begin{array}{l} \det A \neq 0, \\ A\underline{x} = \underline{0} \quad \text{implies } \underline{x} = \underline{0}, \\ 0 \text{ is not an eigenvalue of } A. \end{array}$$

5. The terms real symmetric and positive definite will appear a number of times in this module. A matrix $A \in \mathbb{R}^{n\times n}$ has these properties if

   $$A = A^T \quad \text{and} \quad \underline{x}^T A \underline{x} > 0 \quad \text{for all } \underline{x} \neq \underline{0}.$$

6. When A is real and symmetric all its eigenvalues are real and it has n orthonormal eigenvectors. This is an important result which will be used a number of times in this module. Thus when A is real symmetric and positive definite and $\underline{v} \neq \underline{0}$ is an eigenvector of A the corresponding eigenvalue is

   $$\lambda = \frac{\underline{v}^T A \underline{v}}{\underline{v}^T \underline{v}} > 0.$$

   That is the eigenvalues of such matrices are positive and $\det A > 0$.

2 Direct methods for solving linear systems

In this section we consider a class of methods for attempting to solve the system of linear equations

$$A\underline{x} = \underline{b},$$

where A is an $n \times n$ matrix and $\underline{b} \neq \underline{0}$ is an $n \times 1$ column vector. This part of the module is concerned with the class of methods known as **direct methods** which generally means Gauss elimination in some form. Another type of methods to be considered briefly later is known as **iterative methods**. Roughly speaking direct methods are methods which give you the exact solution $\underline{x}$ (when it exists) in a finite number of steps when exact arithmetic is used. In practice however computer floating point arithmetic is used and thus only an approximation (usually an accurate approximation) is obtained. Iterative methods are concerned with generating a sequence of vectors $\underline{x}^{(1)}$, $\underline{x}^{(2)}$, $\cdots$ which hopefully get closer and closer to $\underline{x}$. With most iterative methods the exact answer is not achieved in a finite number of steps but it is sometimes possible to get a 'sufficiently accurate' approximation to $\underline{x}$ faster than with direct methods. Iterative methods are usually used for sparse systems and often require considerably less storage than do direct methods when implemented on a computer.

2.1 Basic idea of direct methods

The basic idea of most direct methods is to systematically transform the system $A\underline{x} = \underline{b}$ into an equivalent system (i.e. a system with the same solution) $S\underline{x} = \hat{\underline{b}}$ which is 'easy' to solve. In practice this means changing the system to an equivalent triangular system, i.e. with S as an upper or a lower triangular matrix. Thus we begin by considering how to solve a triangular system.

2.2 Triangular systems of equations

First a small example. Consider the system of equations.

$$\begin{array}{ccccccccc} x_1 & + & x_2 & & & & & = & 0 \\ & & 2x_2 & + & x_3 & & & = & 3 \\ & & & & 3x_3 & + & x_4 & = & -2 \\ & & & & & & 4x_4 & = & 4 \end{array}$$

The last equation only has one unknown and hence it can be solved immediately to give $x_4 = 1$. This is then substituted into the 3rd equation to give $x_3 = 1$ and continuing in this way we get $x_2 = 2$ and $x_1 = -2$. We now generalise the procedure.

Let $U = (u_{ij})$ be an upper triangular matrix and consider the system

$$U\underline{x} = \underline{b}$$

or in full

$$\begin{array}{ccccccccc} u_{11}x_1 & + & u_{12}x_2 & + & \cdots & + & u_{1n}x_n & = & b_1 \\ & + & u_{22}x_2 & + & \cdots & + & u_{2n}x_n & = & b_2 \\ & & \ddots & & & & & & \vdots \\ & & & & u_{n-1,n-1}x_{n-1} & + & u_{n-1,n}x_n & = & b_{n-1} \\ & & & & & & u_{nn}x_n & = & b_n \end{array}$$

This is solved by **back substitution**, i.e. we get x_n then x_{n-1} then $\cdots$ and finally x_1 by doing the following.

$$\begin{aligned} x_n &= b_n/u_{nn} \\ x_i &= \left(b_i - \sum_{k=i+1}^{n} u_{ik}x_k\right)/u_{ii}, \quad i = n-1, \cdots, 1 . \end{aligned}$$

For this to work we require that the diagonal entries $u_{ii} \neq 0$ for $i = 1, 2, \cdots, n$. To calculate x_i involves $(n-i)$ multiplications, 1 division and $(n-i)$ additions/subtractions. If you do not distinguish between the operations then the total number of such operations is

$$\begin{aligned} \sum_{i=1}^{n}(1+2(n-i)) &= (2n-1)+(2n-3)+\cdots+5+3+1 \\ &= \frac{1}{2}(2n+2n+\cdots+2n) = n^2 . \end{aligned}$$

On a computer such operations are known as floating point operations (flops) and thus back substitution involves n^2 flops. As a comparison, multiplying a full matrix A by a vector $\underline{x}$ involves $2n^2$ flops.

Lower triangular systems are similarly solved by forward substitution and have the same operation count.

2.3 Reduction to triangular form by Gauss elimination

The basic operation in Gauss elimination is the subtraction of a multiple of one equation from another in order to eliminate certain unknowns. In its basic form (assuming no pivoting is necessary) it involves using the first equation to eliminate the first unknown x_1 from the other $n-1$ equations. The new second equation is then used to eliminate x_2 from equations 3 to n and continuing in this way the kth step involves using the current kth equation to eliminate x_k from equations $k+1$ to n. To distinguish between the different equivalent systems we will use the superscript (k) for the kth system, i.e. at the end of the kth step we have

$$A^{(k)}\underline{x} = \underline{b}^{(k)}$$

and at the start $A^{(0)} = A$, $\underline{b}^{(0)} = \underline{b}$. In full the initial system is

$$\begin{array}{ccccccc} a_{11}^{(0)}x_1 & + & \cdots & + & a_{1n}^{(0)}x_n & = & b_1^{(0)} \\ a_{21}^{(0)}x_1 & + & \cdots & + & a_{2n}^{(0)}x_n & = & b_2^{(0)} \\ \vdots & & & & \vdots & & \vdots \\ a_{n1}^{(0)}x_1 & + & \cdots & + & a_{nn}^{(0)}x_n & = & b_n^{(0)} \end{array}$$

After step 1 we have

$$\begin{array}{ccccccccc} a_{11}^{(0)}x_1 & + & a_{12}^{(0)}x_2 & + & \cdots & + & a_{1n}^{(0)}x_n & = & b_1^{(0)} \qquad \text{(no change in equation 1)} \\ & & a_{22}^{(1)}x_2 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\ & & a_{32}^{(1)}x_2 & + & \cdots & + & a_{3n}^{(1)}x_n & = & b_3^{(1)} \\ & & \vdots & & & & \vdots & & \vdots \\ & & a_{n2}^{(1)}x_2 & + & \cdots & + & a_{nn}^{(1)}x_n & = & b_n^{(1)} \end{array}$$

After $k-1$ steps we have

$$\begin{array}{ccccccccc}
a_{11}^{(0)}x_1 & + & \cdots & + & \cdots & + & a_{1n}^{(0)}x_n & = & b_1^{(0)} \\
 & \ddots & & & & & \vdots & & \vdots \\
 & & a_{kk}^{(k-1)}x_k & + & \cdots & + & a_{kn}^{(k-1)}x_n & = & b_k^{(k-1)} \\
 & & a_{k+1,k}^{(k-1)}x_k & + & \cdots & + & a_{k+1,n}^{(k-1)}x_n & = & b_{k+1}^{(k-1)} \\
 & & \vdots & & & & \vdots & & \vdots \\
 & & a_{nk}^{(k-1)}x_k & + & \cdots & + & a_{nn}^{(k-1)}x_n & = & b_n^{(k-1)}
\end{array}$$

We must describe the kth step. We use the current kth equation to eliminate x_k from the current equations $k+1$ to n. The kth equation is known as the **pivotal equation**, $a_{kk}^{(k-1)}$ is the **pivot element** and

$$m_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k+1, \cdots, n$$

are the **multipliers**. We have, in terms of equations,

$$(\text{Equation } i)^{(k)} = (\text{Equation } i)^{(k-1)} - m_{ik}(\text{Equation } k)^{(k-1)}, \quad i = k+1, \cdots, n\ .$$

In components

$$\begin{array}{rclrcl}
a_{ij}^{(k)} & = & a_{ij}^{(k-1)} - m_{ik}a_{kj}^{(k-1)}, & j & = & k+1, \cdots, n, \\
 & & & i & = & k+1, \cdots, n, \\
b_i^{(k)} & = & b_i^{(k-1)} - m_{ik}b_k^{(k-1)}, & i & = & k+1, \cdots, n.
\end{array}$$

The elimination is complete at the end of step $n-1$ as $A^{(n-1)}$ is upper triangular. If you count the number of floating point operations (flops) you find that this requires the order of $2n^3/3$ flops which we write as $\mathcal{O}(2n^3/3)$. The n^3 term arises since we have 3 nested looping operations to perform (the outer k loop and the inner i and j loops). The subsequent back substitution to actually obtain $\underline{x}$ involves n^2 flops. Hence for large n the reduction part involves by far the most computation.

2.4 Pivoting and stability

As given the basic algorithm fails if any of the pivot elements $a_{kk}^{(k-1)} = 0$, e.g. it fails for the following well-conditioned[2] system

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

which has the solution $x_1 = x_2 = 1$.

The method is numerically unstable if any of the pivot elements $a_{kk}^{(k-1)}$ are very small, e.g. a system of the form

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1+\epsilon \\ 2 \end{pmatrix},$$

where ϵ is 'very small'. The exact solution is again $x_1 = x_2 = 1$. To understand what goes wrong with basic Gauss elimination suppose that $0 < \epsilon < 10^{-16}$ which is small enough such that on a computer working with 8 bytes (64 bits) for each number the computer cannot distinguish between

[2] The condition of a matrix is defined in another section of this module. Roughly speaking a well-conditioned system is one which is not close to a singular system'.

1 and $1+\epsilon$, i.e. ϵ is less than the machine precision. When this is the case the first equation rounds to

$$\epsilon x_1 + x_2 = 1 \ .$$

The elimination below the diagonal in column 1 gives the following equation for x_2,

$$\left(1 - \frac{1}{\epsilon}\right) x_2 = 2 - \frac{1}{\epsilon} \quad \text{which rounds to} \quad -\frac{1}{\epsilon} x_2 = -\frac{1}{\epsilon} \ .$$

This correctly gives $x_2 = 1$. However the first equation $\epsilon x_1 + x_2 = 1$ then gives $x_1 = 0$ which is far from the correct value.

Pivoting is a technique to try and improve the stability and roughly it involves re-ordering the equations in order to put as large as possible an entry in the pivot position. This is known as **partial pivoting**. A variant of this is **row scaled partial pivoting** which is the technique usually used. In the case of the 2×2 system these approaches would effectively just involve swapping the first and second equations and then applying the basic Gauss elimination procedure to the system

$$\begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1+\epsilon \end{pmatrix} \ .$$

Another technique of this type is **complete pivoting** which involves possibly re-ordering both the equations and the unknowns. This is the most stable pivoting algorithm but as it involves a significant increase in the computation, which is rarely necessary, and as a consequence it is not usually used in practice.

Pivoting is important for constructing an algorithm which is 'stable in practice' for a general system but it is not something which will be considered in any more detail in these notes. For certain special systems to be considered later (e.g. involving symmetric positive definite matrices) we will show that pivoting is not necessary.

2.5 A matrix description of basic Gauss elimination

The kth step of Gauss elimination involves subtracting a multiple of the kth equation from equations $k+1, \cdots, n$ in the system

$$A^{(k-1)} \underline{x} = \underline{b}^{(k-1)} \ .$$

Now

$$\underline{e}_k{}^T A^{(k-1)} = k\text{th row of } A^{(k-1)}$$

and

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ m_{k+1,k} \\ \vdots \\ m_{nk} \end{pmatrix} \underline{e}_k{}^T A^{(k-1)} = \text{matrix with each row a multiple of the } k\text{th row of } A^{(k-1)}.$$

Define

$$\underline{m}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ m_{k+1,k} \\ \vdots \\ m_{nk} \end{pmatrix} \ .$$

Then

$$\begin{aligned} A^{(k)} &= A^{(k-1)} - \underline{m}_k\, \underline{e}_k{}^T A^{(k-1)} = (I - \underline{m}_k\, \underline{e}_k{}^T)A^{(k-1)} \\ &= M_k A^{(k-1)}, \end{aligned}$$

where

$$M_k = I - \underline{m}_k\, \underline{e}_k{}^T$$

is a **Gauss transformation matrix**. Thus if we let $U = A^{(n-1)}$ be the upper triangular matrix at the end of the Gauss elimination procedure then we have

$$U = A^{(n-1)} = M_{n-1}M_{n-2}\cdots M_1 A$$

which rearranges to

$$A = (M_{n-1}\cdots M_1)^{-1}U = M_1^{-1}\cdots M_{n-1}^{-1}U = LU$$

where

$$L = M_1^{-1}\cdots M_{n-1}^{-1}\ .$$

2.6 Some properties of M_k and L

1.
$$M_k = I - \underline{m}_k\underline{e}_k{}^T = \begin{pmatrix} 1 & \cdots & & & 0 \\ \vdots & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ 0 & & & & 1 \end{pmatrix} - \begin{pmatrix} 0 & \cdots & & & 0 \\ \vdots & \ddots & & & \\ & & 0 & & \\ & & * & \ddots & \\ 0 & & * & & 0 \end{pmatrix}$$

   where $*$ indicates a possible non-zero entry. Thus M_k is **unit lower triangular**.

2.
$$M_k^{-1} = (I - \underline{m}_k\underline{e}_k{}^T)^{-1} = I + \underline{m}_k\underline{e}_k{}^T\ .$$

   This follows by considering the product

$$(I - \underline{m}_k\underline{e}_k{}^T)(I + \underline{m}_k\underline{e}_k{}^T) = I - \underline{m}_k\underline{e}_k{}^T\underline{m}_k\underline{e}_k{}^T = I$$

   since $\underline{e}_k{}^T\underline{m}_k = (k\text{th component of } \underline{m}_k) = 0$.

3.
$$L = M_1^{-1}\cdots M_{n-1}^{-1} = I + \underline{m}_1\underline{e}_1{}^T + \cdots + \underline{m}_{n-1}\underline{e}_{n-1}{}^T$$

   is **unit lower triangular**. This can be proved by induction by showing that

$$M_1^{-1}\cdots M_{r-1}^{-1} = I + \underline{m}_1\underline{e}_1{}^T + \cdots + \underline{m}_{r-1}\underline{e}_{r-1}{}^T\ .$$

   It is true when $r = 2$ by part (2). Suppose it is true for $r = k$, i.e.

$$M_1^{-1}\cdots M_{k-1}^{-1} = I + \underline{m}_1\underline{e}_1{}^T + \cdots + \underline{m}_{k-1}\underline{e}_{k-1}{}^T\ .$$

   Then

$$\begin{aligned} M_1^{-1}\cdots M_k^{-1} &= \left(M_1^{-1}\cdots M_{k-1}^{-1}\right)M_k^{-1} \\ &= \left(I + \underline{m}_1\underline{e}_1{}^T + \cdots + \underline{m}_{k-1}\underline{e}_{k-1}{}^T\right)\left(I + \underline{m}_k\underline{e}_k{}^T\right) \\ &= I + \underline{m}_1\underline{e}_1{}^T + \cdots + \underline{m}_k\underline{e}_k{}^T \\ &\qquad + \left(\underline{m}_1\underline{e}_1{}^T + \cdots + \underline{m}_{k-1}\underline{e}_{k-1}{}^T\right)\underline{m}_k\underline{e}_k{}^T\ . \end{aligned}$$

The last term vanishes since

$$\underline{e}_1{}^T \underline{m}_k = \cdots = \underline{e}_{k-1}{}^T \underline{m}_k = 0$$

as the first k components of $\underline{m}_k$ are zero.

2.7 Solving $A\underline{x} = \underline{b}$ when we have the factorization $A = LU$

In the previous subsections it has been shown that basic Gauss elimination is equivalent to an LU factorization of the matrix A with L a unit lower triangular matrix and U an upper triangular matrix. The multipliers used in Gauss elimination are precisely the strictly lower triangular entries of L. Once the factors L and U have been computed so that $A = LU$ we have

$$A\underline{x} = LU\underline{x} = \underline{b} \; .$$

To solve for $\underline{x}$ we have

$$L\underline{y} = \underline{b} \qquad \text{(solve for } \underline{y} \text{ by forward substitution)}$$

and

$$U\underline{x} = \underline{y} \qquad \text{(solve for } \underline{x} \text{ by back substitution).}$$

A point to note is that to solve $A\underline{x} = \underline{b}$ (the solution being $\underline{x} = A^{-1}\underline{b}$) we do not need to compute A^{-1}. Indeed A^{-1} is usually computed by first computing $A = LU$ and then solving $A\underline{x} = \underline{e}_i$ for $i = 1, 2, \cdots, n$ by forward and back substitution.

2.8 Uniqueness of the LU factorization

The basic Gauss elimination procedure, when it works, gives both the existence of a the LU factorization and a constructive way of computing the factorization. We show now that provided the factorization exists and $\det A \neq 0$ then the factors L and U are unique. To do this we require the following Lemma concerning properties of triangular matrices.

Lemma.

(1) If U is upper triangular and non-singular then U^{-1} is also upper triangular. Similarly for lower triangular matrices.

(2) If L is unit lower triangular then L^{-1} is also unit lower triangular. Similarly for unit upper triangular matrices.

(3) The product of 2 upper triangular matrices is upper triangular. Similarly for lower triangular matrices. The product of 2 unit lower triangular matrices is unit lower triangular. Similarly for unit upper triangular matrices.

Proof.

(1) The solution of $U\underline{x} = \underline{e}_i$ is $\underline{x} = U^{-1}\underline{e}_i$ $=i$th column of U^{-1}. In full this linear system is

$$\begin{array}{ccccccccc}
u_{11}x_1 & + & \cdots & + & \cdots & + & u_{1n}x_n & = & 0 \\
 & & \cdots & & \cdots & & \vdots & & \vdots \\
 & & u_{ii}x_i & + & \cdots & + & u_{in}x_n & = & 1 \\
 & & & & \ddots & & & & \vdots \\
 & & & & & & u_{nn}x_n & = & 0
\end{array}$$

By considering equations n, $n-1$, $\cdots$, $i+1$, back substitution gives $x_n = 0$, $x_{n-1} = 0$, $\cdots$, $x_{i+1} = 0$. Then

$$x_i = 1/u_{ii},$$

i.e. the ith column of U (which is the vector $U^{-1}\underline{e}_i$) is zero below the diagonal.

(2) The proof of part (2) is similar to the proof of part (1).

(3) If A and B are upper triangular then

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} = \sum_{k=i}^{j} a_{ik}b_{kj} = 0 \quad \text{if } i > j.$$

Thus AB is upper triangular. All other possibilities follow similarly.

Theorem. If A is non-singular and

$$A = L_1U_1 = L_2U_2$$

with L_1 and L_2 as unit lower triangular matrices and U_1 and U_2 as upper triangular matrices then

$$L_1 = L_2 \quad \text{and} \quad U_1 = U_2\ .$$

Proof. The first thing to note is that L_1, U_1, L_2 and U_2 are all non-singular. This follows from the relations

$$\det L_1 = 1, \quad \det L_2 = 1,$$

and

$$0 \neq \det A = (\det L_1)(\det U_1) = \det U_1, \quad 0 \neq \det A = (\det L_2)(\det U_2) = \det U_2\ .$$

(The result about determinants that $A = L_1U_1$ implies that $\det A = \det L_1 \det U_1$ is a result you have met in your first year Algebra modules.) Thus rearranging $L_1U_1 = L_2U_2$ gives

$$L_2^{-1}L_1 = U_2U_1^{-1}\ .$$

By the Lemma the left hand side is unit lower triangular and the right hand side is upper triangular. The only possibility for both of these to be true is when

$$L_2^{-1}L_1 = U_2U_1^{-1} = I, \quad \text{i.e. } L_2 = L_1, \quad U_2 = U_1\ .$$

□

2.9 Existence of the LU factorization

Since the LU factorization of A is equivalent to the Gauss elimination procedure the factorization exists provided zero pivots are not encountered during the elimination. As we will show a zero pivot corresponds to a principal sub-matrix of A being singular.

Definition. The $k \times k$ principal sub-matrix of A is the $k \times k$ matrix

$$\begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix}.$$

If $A = LU$ is an LU factorization then

$$a_{ij} = \sum_{k=1}^{n} l_{ik} u_{kj} = \sum_{k=1}^{\min\{i,j\}} l_{ik} u_{kj} \ .$$

The upper limit of the sum is $\min\{i,j\}$ in the above as a consequence of the properties of L and U. Thus if we restrict i, j to the range $1 \le i, j \le k$ then

$$\min\{i,j\} \le k$$

and

$$(L_k U_k)_{ij} = (LU)_{ij}$$

where L_k and U_k are the leading $k \times k$ principal submatrices of L and U respectively. Thus

$$A_k = L_k U_k \ .$$

That is $A = LU$ implies that $A_k = L_k U_k$ for $k = 1, 2, \cdots, n$. As a consequence of these relations

$$\det A_k = (\det L_k)(\det U_k) = \det U_k = u_{11} u_{22} \cdots u_{kk} = a_{11}^{(0)} a_{22}^{(1)} \cdots a_{kk}^{(k-1)} \ .$$

where, as indicated, the diagonal entries of the upper triangular matrix generated by the elimination process are precisely the pivots used. Thus if all pivots encountered are non-zero then

$$\det A_k \ne 0, \quad k = 1, \cdots, n-1 \ .$$

If the kth pivot is zero (i.e.$u_{kk} = a_{kk}^{(k-1)} = 0$) then the kth principal sub-matrix is singular and the elimination process breaks down at this point.

Observe that to have a solution to the system we only require that A is non-singular, i.e. $\det A = \det A_n \ne 0$ but for the basic Gauss elimination procedure to work we also require that $\det A_k \ne 0$, $k = 1, 2, \cdots, n-1$. These extra conditions are removed when a pivoting procedure is used.

2.10 Obtaining the LU factorization directly (*unlikely to be covered in the lectures*)

We now consider how to obtain the entries of L and U by doing the computations in a different order to that used in the basic Gauss elimination. Recall that in the basic Gauss elimination the kth row of U is determined at the end of the $(k-1)$th stage (it gives the pivotal equation at the kth step) and that at the start of the kth step the multipliers m_{ik} are computed and these are precisely the entries in the kth column of L. We consider now relations which involve these. We have

$$(A)_{kj} = (LU)_{kj} \quad \text{gives} \quad a_{kj} = \sum_{r=1}^{k} l_{kr} u_{rj} = \left(\sum_{r=1}^{k-1} l_{kr} u_{rj} \right) + u_{kj}, \quad j = k, \cdots, n$$

and

$$(A)_{ik} = (LU)_{ik} \quad \text{gives} \quad a_{ik} = \sum_{r=1}^{k} l_{ir} u_{rk} = \left(\sum_{r=1}^{k-1} l_{ir} u_{rk} \right) + l_{ik} u_{kk}, \quad i = k+1, \cdots, n.$$

Thus rearranging

$$\begin{aligned} u_{kj} &= a_{kj} - \sum_{r=1}^{k-1} l_{kr} u_{rj}, \quad \text{for } j = k, \cdots, n, \\ l_{ik} &= \left(a_{ik} - \sum_{r=1}^{j-1} l_{ir} u_{rk} \right) / u_{kk}, \quad \text{for } i = k+1, \cdots, n \ . \end{aligned}$$

2.11 Remarks on the computer implementation (*unlikely to be covered in the lectures*)

The LU factorization algorithm involves doing the above for $k = 1, 2, \cdots, n$. This involves the same operations as Gauss elimination but with a different order. Gauss elimination involved many operations of the form

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik}a_{kj}^{(k-1)}, \quad (m_{ik} = l_{ik}),$$

i.e. part of the computation of u_{kj} and l_{ik} is done at each of the elimination steps prior to step k. The LU factorization idea is to wait until the computation can be done in one go. This does not involve any less flops but it is the preferred way of doing the operation on a computer because it can better exploit the way a computer actually performs its floating point operations. It involves many computations of the form

$$\sum_1^m y_i z_i = y_1 z_1 + \cdots + y_m z_m$$

(known technically as a **BLAS 1 operation** [3]) which can be done efficiently and accurately by using the **double accumulation of inner products**. Roughly this involves the following. If the mantissa of numbers stored on a computer involve t binary digits[4] then the mantissa of a product $y_i z_i$ involves $2t$ binary digits for an exact representation. This is done internally in the computers registers but then this is usually truncated back to t digits which gives a floating point error. However if the $2t$ digits can be kept in the register then we can successively compute

$$y_1 z_1, \quad y_1 z_1 + y_2 z_2, \quad \cdots, \quad y_1 z_1 + \cdots + y_m z_m$$

to $2t$ binary digit accuracy with the truncation to t digits done only once at the end. This is thus faster and more accurate.

2.12 Remarks on pivoting

Mathematically partial pivoting corresponds to an LU factorization of a matrix of the form

$$PA = LU,$$

where P is a permutation matrix (the columns of P are a re-arrangement of the columns of the identity matrix). The matrix PA is the matrix A with the rows re-ordered which in the linear equations case $A\underline{x} = \underline{b}$ corresponds to rearranging the equations. Complete pivoting corresponds to a factorization

$$PAQ = LU,$$

where P and Q are both permutation matrices. When Gauss elimination or the equivalent LU factorization is implemented with pivoting the re-arrangements described by P (and Q in the case of complete pivoting) are determined during the process.

[3] BLAS stands for Basic Linear Algebra Software

[4] A number has a sign ($+$ or $-$) a mantissa and an exponent. In decimal form the FORTRAN number `-0.571E+07` has the sign $-$, `0.571` corresponds to the mantissa part and `E+07` corresponds to the exponent part.

2.13 Remarks about special linear systems

With general linear systems it is essential that some sort of pivoting is used in order to be confident of generating a reasonably accurate approximation to systems of equations that are not badly conditioned[5]. However, in many applications the systems which arise often have some special structure which enables the LU factorization to be performed in a stable manner without pivoting. In addition the systems sometimes also have a sparse banded structure which implies that L and U also have the sparse banded form provided pivoting is not done. (If pivoting is done for such a matrix then the bandwidths of the resulting L and U are greater than that of the original matrix A.)

In the next two subsections we consider two specific special systems. In subsection 2.14 we consider the solution of a tri-diagonal system and then in subsection 2.15 we consider the case of a linear system with a real symmetric and positive definite matrix which can always be factorized without pivoting in a 'symmetric way' in what is called as the Cholesky factorization.

2.14 A Tri-diagonal system

A tri-diagonal system involves a tri-diagonal matrix which is a matrix for which all entries are 0 except on the diagonal and on the sub- and super- diagonals. It is customarily in such situations to adopt a one-subscript notation (a_i, d_i and b_i) for the entries instead of the double subscript notation a_{ij}. Thus we will write the system as

$$\begin{aligned} d_1x_1 + b_2x_2 &= r_1 \\ a_ix_{i-1} + d_ix_i + b_{i+1}x_{i+1} &= r_i, \quad 2 \le i \le n-1, \\ a_nx_{n-1} + d_nx_n &= r_n \end{aligned}$$

or as

$$\begin{pmatrix} d_1 & b_2 & \cdots & & & 0 \\ a_2 & d_2 & b_3 & & & \vdots \\ \vdots & a_3 & d_3 & b_4 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & b_n \\ 0 & \cdots & & & a_n & d_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ \vdots \\ r_{n-1} \\ r_n \end{pmatrix}.$$

The first step in Gauss elimination involves eliminating below the diagonal in column 1 and thus in this case we need only subtract a multiple of equation 1 from equation 2 to produce the modified equation

$$d_2'x_2 + b_3x_3 = r_2', \quad \text{where} \quad l_2 = a_2/d_1,\ d_2' = d_2 - l_2b_2,\ r_2' = r_2 - l_2r_1 .$$

One step of Gauss elimination (to create d_2' and r_2' in this case) hence just involves 1 division, 2 multiplications and 2 subtractions. Continuing this process to eliminate below the diagonal in column 2, then in column 3, then in $\cdots$ and finally in column $n-1$ similarly gives

$$\begin{aligned} &l_3 = a_3/d_2', \quad &&d_3' = d_3 - l_3b_3, \quad &&r_3' = r_3 - l_3r_2, \\ &l_4 = a_4/d_3', \quad &&d_4' = d_4 - l_4b_4, \quad &&r_4' = r_4 - l_4r_3. \\ &\cdots &&\cdots &&\cdots \end{aligned}$$

[5]The term 'badly conditioned' is defined in another section of this module. Roughly speaking a system is badly conditioned if the corresponding matrix is 'nearly singular'. When a system is nearly singular it is difficult to solve accurately.

With $d_1' = d_1$ and $r_1' = r_1$ we hence have

$$l_k = a_k/d_{k-1}', \quad d_k' = d_k - l_k b_k, \quad r_k' = r_k - l_k r_{k-1}', \quad k = 2, 3, \cdots, n \ .$$

In matrix form we have created the upper triangular system

$$\begin{pmatrix} d_1' & b_2 & \cdots & & & 0 \\ & d_2' & b_3 & & & \vdots \\ & & d_3' & b_4 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & b_n \\ 0 & \cdots & & & & d_n' \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} r_1' \\ r_2' \\ \vdots \\ \vdots \\ r_{n-1}' \\ r_n' \end{pmatrix}$$

which can then be solved by back substitution. We simply have

$$\begin{aligned} x_n &= r_n'/d_n' \\ x_i &= (r_i' - b_{i+1}x_{i+1})/d_i' \quad \text{for } i = n-1, n-2, \cdots, 1. \end{aligned}$$

The point to note is that the storage is $\mathcal{O}(n)$ and the number of operations to obtain the solution is also $\mathcal{O}(n)$ in contrast to a general system which has respectively $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$.

Another way of expressing what has just been done is in terms of the LU factorization of the tri-diagonal matrix. Since Gauss elimination without pivoting is the same as an LU factorization we have that

$$\begin{pmatrix} d_1 & b_2 & \cdots & & & 0 \\ a_2 & d_2 & b_3 & & & \vdots \\ \vdots & a_3 & d_3 & b_4 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & b_n \\ 0 & \cdots & & & a_n & d_n \end{pmatrix} = \begin{pmatrix} 1 & & & & & 0 \\ l_2 & 1 & & & & \\ \vdots & l_3 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & & \\ 0 & \cdots & & & l_n & 1 \end{pmatrix} \begin{pmatrix} d_1' & b_2 & \cdots & & & 0 \\ \vdots & d_2' & b_3 & & & \vdots \\ & & d_3' & b_4 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & b_n \\ 0 & \cdots & & & & d_n' \end{pmatrix} .$$

A procedure which efficiently stores and solves tri-diagonal systems is already implemented in MATLAB. If `a`, `d`, `b` and `r` in MATLAB correspond respectively to the vectors $\underline{a}$, $\underline{d}$, $\underline{b}$ and $\underline{r}$ and `n` has already been set then in MATLAB the system can be set up and solved by the 2 commands

```
T=spdiags([a d b],-1:1,n,n);
x=T\r;
```

The `spdiags` command creates the matrix and stores it in a sparse way. The MATLAB interpreter recognises that `T` is sparse and performs the solution procedure implied by `T\r` in an efficient way as indicated above.

As another example of this consider the following script.

```
diary trid1
echo on
format rat
% set a small tridiagonal matrix in sparse form
n=5;
o=ones(n,1);
T=spdiags([o 2*o o],-1:1,n,n);

% get the LU factorization and show it
[L U]=lu(T);
fprintf('T=\n');disp(full(T));
fprintf('L=\n');disp(full(L));
fprintf('U=\n');disp(full(U));
diary off
```

This generates the following

```
diary trid1
echo on
format rat
% set a small tridiagonal matrix in sparse form
n=5;
o=ones(n,1);
T=spdiags([o 2*o o],-1:1,n,n);

% get the LU factorization and show it
[L U]=lu(T);
fprintf('T=\n');disp(full(T));
T=
      2             1             0             0             0
      1             2             1             0             0
      0             1             2             1             0
      0             0             1             2             1
      0             0             0             1             2

fprintf('L=\n');disp(full(L));
L=
      1             0             0             0             0
     1/2            1             0             0             0
      0            2/3            1             0             0
      0             0            3/4            1             0
      0             0             0            4/5            1

fprintf('U=\n');disp(full(U));
U=
      2             1             0             0             0
      0            3/2            1             0             0
      0             0            4/3            1             0
      0             0             0            5/4            1
      0             0             0             0            6/5

diary off
```

As can be seen, the matrix T is tridiagonal and L and U are bidiagonal matrices which are lower triangular and upper triangular respectively.

2.15 Real symmetric and positive definite systems

A matrix A is a real symmetric and positive definite matrix if its entries are real and

$$A = A^T \qquad \text{and} \qquad \underline{x}^T A \underline{x} > 0 \quad \text{for all } \underline{x} \neq \underline{0}.$$

There are not too many theorems in this module but we now present one that guarantees a special type of lower triangular/ upper triangular factorization of such a matrix.

Theorem. (Cholesky factorization) If A is an $n \times n$ real symmetric positive definite matrix then there exists a unique lower triangular $n \times n$ real matrix L with positive diagonal entries such that

$$A = LL^T \ .$$

(Note that here L denotes a lower triangular matrix and not a unit lower triangular matrix as in the previous pages of these notes.)

Proof.

We use induction on the size n of the matrix.

The case $n = 1$.
In this case $\underline{x}^T A \underline{x} = a_{11} x_1^2 > 0$ for all $\underline{x} \neq \underline{0}$ implies that $a_{11} > 0$. Hence take $l_{11} = \sqrt{a_{11}} > 0$ and we have $A = LL^T$.

Induction hypothesis: Assume the result holds for all $n < m$.

Let A_k be the $k \times k$ leading principal sub-matrix of A. We consider A_m in partitioned form.

$$A_m = \begin{pmatrix} A_{m-1} & \underline{b} \\ \underline{b}^T & a_{mm} \end{pmatrix}, \quad \underline{b} \in \mathbb{R}^{m-1} \ .$$

Since A is positive definite all its principal sub-matrices are also positive definite. This follows since if

$$\underline{x}^T = (x_1, \cdots, x_k, 0, \cdots, 0) \in \mathbb{R}^m \quad \text{and} \quad \underline{y} = (x_1, \cdots, x_k)^T \in \mathbb{R}^k$$

then

$$\underline{x}^T A \underline{x} = \underline{y}^T A_k \underline{y} > 0 \quad \text{for all } \underline{y} \neq \underline{0}.$$

Thus A_{m-1} satisfies the positive definite condition and by the induction hypothesis A_{m-1} has a unique Cholesky factorization

$$A_{m-1} = L_{m-1} L_{m-1}^T \ .$$

We now construct a factorization of A_m. For this consider the following product

$$\begin{pmatrix} L_{m-1} & \underline{0} \\ \underline{l}^T & l_{mm} \end{pmatrix} \begin{pmatrix} L_{m-1}^T & \underline{l} \\ \underline{0}^T & l_{mm} \end{pmatrix} = \begin{pmatrix} L_{m-1} L_{m-1}^T & L_{m-1} \underline{l} \\ \underline{l}^T L_{m-1}^T & \underline{l}^T \underline{l} + l_{mm}^2 \end{pmatrix}$$

The $(m-1) \times (m-1)$ principal submatrix of the right hand side matches that of A_m and thus for all the right hand side to agree with A_m we just need

$$L_{m-1} \underline{l} = \underline{b} \quad \text{and} \quad \underline{l}^T \underline{l} + l_{mm}^2 = a_{mm} \ .$$

This uniquely determines $\underline{l}$ (the unique solution of $L_{m-1}\underline{l} = \underline{b}$) and $l_{mm}^2 = a_{mm} - \underline{l}^T\underline{l}$. Hence we get a factorization but to complete the proof we still need to show that

$$a_{mm} - \underline{l}^T\underline{l} > 0$$

so that l_{mm} is real. To prove this the trick is to use results about determinants and a result about the eigenvalues of positive definite symmetric matrices. As we have a L_m such that $A_m = L_m L_m^T$ then we have

$$\begin{aligned} \det A_m = \det(L_m L_m^T) &= (\det L_m)(\det L_m^T) \\ &= (l_{11}\cdots l_{mm})^2 \end{aligned}$$

since L_m is lower triangular. Now since a real symmetric positive definite matrix has real positive eigenvalues and

$$\det A_m = \text{product of the eigenvalues of } A_m$$

it follows that $\det A_m > 0$ which in turn implies that $l_{mm}^2 > 0$ and thus we can choose $l_{mm} > 0$. □

Stability of the Cholesky factorization

With $A = (a_{ij})$ and $L = (l_{ij})$ the factorization $A = LL^T$ is in component form

$$a_{ij} = \sum_{k=1}^{\min\{i,j\}} l_{ik} l_{jk} \ .$$

In words, the i, j entry of A is the inner product of the ith row of L with the jth row of A. For $i = j$ (i.e. the diagonal entry)

$$a_{ii} = \sum_{k=1}^{i} l_{ik}^2 = l_{l1}^2 + l_{i2}^2 + \cdots + l_{ii}^2 \ .$$

From this last equation we have that in particular that none of the non-negative numbers $l_{i1}^2, \cdots, l_{ii}^2$ can exceed a_{ii}, i.e.

$$|l_{ij}| \le \sqrt{a_{ii}} \quad \text{for all } i \text{ and } j.$$

This result which bounds the size of the entries of L in terms of the size of the entries of A (the diagonal entries of A in this particular case) is called a stability result.

The Cholesky algorithm

From the factorization

$$A = LL^T$$

we have in component form

$$a_{ij} = \sum_{k=1}^{\min\{i,j\}} l_{ik} l_{jk} = \begin{cases} \left(\sum_{k=1}^{j-1} l_{ik} l_{jk} \right) + l_{ij} l_{jj}, & j < i, \\ \\ l_{i1}^2 + \cdots + l_{ii}^2, & j = i. \end{cases}$$

This leads to the following algorithm for computing the entries of L.

$$\begin{aligned} y_1 &= a_{11} \\ l_{11} &= \sqrt{y_1} \end{aligned}$$

For $i = 2, \cdots, n$ do the following:

$$
\begin{aligned}
l_{ij} &= \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right) / l_{jj}, \quad j = 1, \cdots, i-1, \\
y_i &= a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2, \\
l_{ii} &= \sqrt{y_i}\ .
\end{aligned}
$$

Remark

The above algorithm requires that $y_i > 0$ for $i = 1, \cdots, n$ which is ensured by the positive definite condition of A. If the algorithm is used on an arbitrary real symmetric matrix then it will fail with $y_i \le 0$ where A_i (which is the $i \times i$ principal sub-matrix of A) is the first leading principal sub-matrix which is not positive definite. This is because if it had not failed then we would have constructed a factorization

$$A_i = L_i L_i^T$$

which in turn implies that for $\underline{x}_i \in \mathbb{R}^i$, $\underline{x}_i \ne \underline{0}$

$$
\begin{aligned}
\underline{x}_i^T A_i \underline{x}_i &= \underline{x}_i^T L_i L_i^T \underline{x}_i = \left(L_i^T \underline{x}_i \right)^T \left(L_i^T \underline{x}_i \right) \\
&= (\text{sum of the squares of the components of the vector } L_i^T \underline{x}_i) > 0,
\end{aligned}
$$

i.e. A_i is positive definite. Hence the Cholesky algorithm is a way of testing whether or not a matrix is positive definite.

Example

$$
A = \begin{pmatrix} 1 & 2 & -1 \\ 2 & 5 & -1 \\ -1 & -1 & 3 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}
$$

We obtain the entries of L here row by row. First on row 1 we have

$$1 = l_{11}^2 \quad \text{gives} \quad l_{11} = 1\ .$$

Next on row 2 we have

$$
\begin{aligned}
2 &= l_{21} l_{11} & \text{gives} \quad l_{21} &= 2, \\
5 &= l_{21}^2 + l_{22}^2 = 4 + l_{22}^2 & \text{gives} \quad l_{22} &= 1.
\end{aligned}
$$

Finally on row 3 we have

$$
\begin{aligned}
-1 &= l_{31} l_{11}, & \text{gives} \quad l_{31} &= -1, \\
-1 &= l_{31} l_{21} + l_{32} l_{22} = -2 + l_{32}, & \text{gives} \quad l_{32} &= 1. \\
3 &= l_{31}^2 + l_{32}^2 + l_{33}^2 = 1 + 1 + l_{33}^2, & \text{gives} \quad l_{33} &= 1.
\end{aligned}
$$

Hence

$$
L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad LL^T = A\ .
$$

To do this with MATLAB you can type the following.

```
A=[1  2 -1
   2  5 -1
  -1 -1  3];
R=chol(A);
disp(R);
```

The output produced is

```
1    2    -1
0    1     1
0    0     1
```

i.e. `R` is upper triangular and corresponds to L^T.

Real symmetric positive definite matrices and the Cholesky factorization will appear again in this module in the section on least squares.

2.16 Some key points of the section

The following are a collection of points to note from this section on direct methods for linear systems.

1. Upper triangular systems $U\underline{x} = \underline{b}$ can be solved by back substitution in $\mathcal{O}(n^2)$ operations when $U \in \mathbb{R}^{n\times n}$.

2. The reduction of $A \in \mathbb{R}^{n\times n}$ to upper triangular form involves $\mathcal{O}(n^3)$ operations and can be described in matrix form using Gauss transformation matrices. We have

$$U = A^{(n-1)} = M_{n-1}\cdots M_1 A, \quad \text{where } M_k = I - \underline{m}_k \underline{e}_k^T.$$

   Key results in the matrix description are

$$M_k^{-1} = I + \underline{m}_k \underline{e}_k^T \quad \text{and} \quad L = (M_{n-1}\cdots M_1)^{-1} = M_1^{-1}\cdots M_{n-1}^{-1} = I + \underline{m}_k \underline{e}^T \cdots + \underline{m}_{n-1}\underline{e}_{n-1}^T.$$

   L is hence unit lower triangular.

3. Given $A = LU$ we can solve $A\underline{x} = LU\underline{x} = \underline{b}$ by solving $L\underline{y} = \underline{b}$ by forward substitution followed by solving $U\underline{x} = \underline{y}$ by back substitution.

4. When $A = LU$, $A_k = L_k U_k$ where A_k, L_k, U_k are the $k \times k$ principal submatrices of A, L and U respectively. The existence of the basic LU factorization depends on the principal submatrices all being non-singular.

5. Tri-diagonal systems involve $\mathcal{O}(n)$ storage and $\mathcal{O}(n)$ operations.

6. Symmetric positive definite matrices have a Cholesky factorization $A = L\,L^T$. Note that

$$a_{ii} = (A)_{ii} = (L\,L^T) = l_{i1}^2 + l_{i2}^2 + \cdots + l_{ii}^2 \quad \text{and thus} \quad l_{ij}^2 \le a_{ii} \quad \text{for } j = 1, 2, \cdots, i.$$

3 Norms for Vectors and Matrices

For a given object (vectors and matrices in our case) a norm is a way of giving a 'size' to the object. For example the modulus of a real or complex number defines a norm and the Euclidean length of a vector in three dimensional geometry defines a norm. With a way of giving a size to objects like vectors and matrices we can then consider when a vector $\underline{x}$ is close to another vector $\underline{y}$ by considering the size of $\underline{x}-\underline{y}$ and we can consider when a matrix A is close to another matrix B by considering the size of $A-B$. In the context of solving $A\underline{x}=\underline{b}$ situations like these arise when we have an approximate solution $\hat{\underline{x}}$ and we need to estimate or compute the size of the error $\underline{x}-\hat{\underline{x}}$ and the size of the residual term $\underline{b}-A\hat{\underline{x}}$. A closely related problem that we will consider concerns determining the size of $\Delta\underline{x}$ where $\Delta\underline{x}$ is such that $\underline{x}+\Delta\underline{x}$ satisfies $(A+\Delta A)(\underline{x}+\Delta\underline{x})=\underline{b}+\Delta\underline{b}$. The aim at this stage is to consider the sensitivity of the solution to a linear system to changes in the matrix and/or right hand side vector $\underline{b}$.

In the n dimensional space $\mathbb{R}^n$ there are many different norms which can be defined but as will be stated in this module all norms in this finite dimensional space are in a certain sense equivalent.

3.1 Vector norms – the axioms

We begin abstractly by giving the **vector norm axioms**.
A function $f:\mathbb{R}^n\to\mathbb{R}$ is a vector norm if the following conditions are satisfied:

(i) $f(\underline{x})\ge 0$ for all $\underline{x}\in\mathbb{R}^n$ with $f(\underline{x})=\underline{0}$ if and only if $\underline{x}=\underline{0}$.

(ii) $f(\alpha\underline{x})=|\alpha|f(\underline{x})$ for all $\alpha\in\mathbb{R}$, $\underline{x}\in\mathbb{R}^n$. (This is **linearity**.)

(iii) $f(\underline{x}+\underline{y})\le f(\underline{x})+f(\underline{y})$ for all $\underline{x},\underline{y}\in\mathbb{R}^n$. (This is the **Triangle inequality**.)

Notation If f is a norm then we use the notation

$$f(\underline{x})=\|\underline{x}\|\ .$$

(When more than one norm is being used or when the particular norm being used is not clear from the context we will use subscripts to distinguish between different norms.)

In this $\|.\|$ notation we repeat the norm axioms as follows:

(i) $\|\underline{x}\|\ge 0$ for all $\underline{x}\in\mathbb{R}^n$ with $\|\underline{x}\|=0$ if and only if $\underline{x}=\underline{0}$.

(ii) $\|\alpha\underline{x}\|=|\alpha|\,\|\underline{x}\|$ for all $\alpha\in\mathbb{R}$ and $\underline{x}\in\mathbb{R}^n$.

(iii) $\|\underline{x}+\underline{y}\|\le\|\underline{x}\|+\|\underline{y}\|$ for all $\underline{x}$ and $\underline{y}$ in $\mathbb{R}^n$.

If all the above axioms are satisfied except that there are non-zero vectors $\underline{x}$ with $\|\underline{x}\|=0$ then we have a **vector semi-norm**.

3.2 Commonly used vector norms

The most commonly used vector norms are the 2-norm, the $\infty-$norm and the $1-$ norm and these are defined as follows.

$$\|\underline{x}\|_2=\left(x_1^2+x_2^2+\cdots+x_n^2\right)^{1/2}=\left(\sum_1^n x_i^2\right)^{1/2}=\left(\underline{x}^T\underline{x}\right)^{1/2}$$

is the **2-norm** or **Euclidean norm**.

$$\|\underline{x}\|_\infty = \max_{1\le i\le n} |x_i|$$

is the **∞-norm**.

$$\|\underline{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n| = \sum_1^n |x_i|$$

is the **1-norm**.

3.3 Verification of the norm axioms for $\|.\|_1$, $\|.\|_2$ and $\|.\|_\infty$

Most of the axioms follow immediately except for the triangle inequality. For the 1-norm and ∞-norm the triangle inequality follows immediately from the triangle inequality for real numbers, i.e.

$$|x_i + y_i| \le |x_i| + |y_i| \ .$$

In the case of the 2-norm there is a bit more work to do. We have

$$\begin{aligned}\|\underline{x}+\underline{y}\|_2^2 &= (\underline{x}+\underline{y})^T(\underline{x}+\underline{y}) = \underline{x}^T\underline{x} + \underline{x}^T\underline{y} + \underline{y}^T\underline{x} + \underline{y}^T\underline{y} \\ &= \underline{x}^T\underline{x} + 2\underline{x}^T\underline{y} + \underline{y}^T\underline{y} \quad \text{since } \underline{x}^T\underline{y} = \underline{y}^T\underline{x}.\end{aligned}$$

Next we use the Cauchy-Schwarz inequality

$$\underline{x}^T\underline{y} \le |\underline{x}^T\underline{y}| \le (\underline{x}^T\underline{x})^{1/2}(\underline{y}^T\underline{y})^{1/2} = \|\underline{x}\|_2\|\underline{y}\|_2 \ .$$

Thus

$$\|\underline{x}+\underline{y}\|_2^2 \le \underline{x}^T\underline{x} + 2(\underline{x}^T\underline{x})^{1/2}(\underline{y}^T\underline{y})^{1/2} + \underline{y}^T\underline{y} = (\|\underline{x}\|_2 + \|\underline{y}\|_2)^2 \ .$$

3.4 The equivalence of the vector norms

The following is a result of functional analysis which applies to all norms of $\mathbb{R}^n$. This will not be proved in this module.

Theorem. All norms of $\mathbb{R}^n$ are **equivalent** in the following sense. If $\|.\|_\alpha$ and $\|.\|_\beta$ are norms on $\mathbb{R}^n$ then there exists positive constants c_1 and c_2 such that for all $\underline{x} \in \mathbb{R}^n$

$$c_1\|\underline{x}\|_\alpha \le \|\underline{x}\|_\beta \le c_2\|\underline{x}\|_\alpha \ .$$

This result means that if we have a sequence of vectors $\underline{y}_1, \underline{y}_2, \underline{y}_3, \cdots$ which are such that $\|\underline{y}_n\|_\alpha \to 0$ as $n\to\infty$ then $\|\underline{y}_n\|_\beta \to 0$ as $n \to \infty$. Hence if a sequence of vectors converges in one of the vector norms of $\mathbb{R}^n$ then it converges in all other norms of $\mathbb{R}^n$. (The statement $\underline{x}_n \to \underline{x}$ as $n \to \infty$ means that $\|\underline{x} - \underline{x}_n\| \to 0$ as $n \to \infty$. The point about the last statement is that it does not matter which norm is used.) In infinite dimensional spaces this result is not true. For example if you construct the Fourier series of a 2π periodic function f and let $f_n(x)$ be given by

$$f_n(x) = \frac{a_0}{2} + \sum_{m=1}^n (a_m \cos mx + b_m \sin mx),\ a_m := \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos mx\,dx,\ b_m := \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin mx\,dx$$

then you can have $\|f - f_n\|_\alpha \to 0$ for some norm $\|.\|_\alpha$ but with $\|f_- f_n\|_\beta \not\to 0$ for some other norm $\|.\|_\beta$. You may meet Fourier series in `MA2031S` (Linear Analysis) and `MA2111S` (Linear Methods).

We consider this result for the 1,2 and ∞ vector norms.

Theorem. For the space $\mathbb{R}^n$ we have

$$\|\underline{x}\|_\infty \le \|\underline{x}\|_2 \le \|\underline{x}\|_1 \le \sqrt{n}\|\underline{x}\|_2 \le n\|\underline{x}\|_\infty \ .$$

Proof. If position l, $1 \le l \le n$ is such that $|x_l| = \max_{1\le i\le n} |x_i| = \|\underline{x}\|_\infty$ then as $0 \le x_i^2 \le x_l^2$

$$x_l^2 \le x_1^2 + \cdots + x_l^2 + \cdots + x_n^2 = \sum_1^n x_i^2 \le n x_l^2$$

which implies that

$$\|\underline{x}\|_\infty^2 \le \|\underline{x}\|_2^2 \le n\|\underline{x}\|_\infty^2 \ .$$

For the square of the 1-norm we have

$$\|\underline{x}\|_1^2 = \left(\sum_{i=1}^n |x_i|\right)^2 = \sum_{i=1}^n \sum_{j=1}^n |x_i x_j| \ .$$

The n^2 numbers in the double sum contains the n numbers x_1^2, $\cdots$, x_n^2 and thus

$$\|\underline{x}\|_2^2 \le \|\underline{x}\|_1^2 \ .$$

By the connection between the geometric and arithmetic means we have

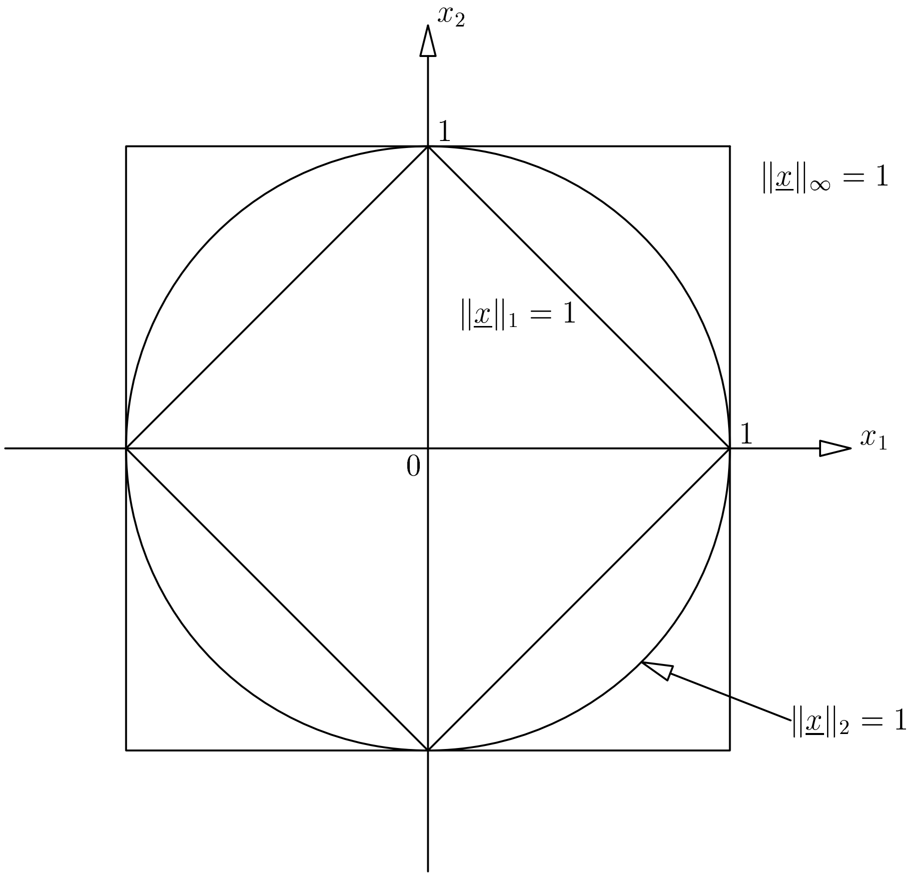
$$|x_i x_j| \le \frac{1}{2}\left(x_i^2 + x_j^2\right) \ .$$

(This is just a rearrangement of $(|x_i| - |x_j|)^2 \ge 0$.) Thus

$$\|\underline{x}\|_1^2 = \sum_1^n \sum_1^n |x_i x_j| \le \frac{1}{2} \sum_1^n \sum_1^n \left(x_i^2 + x_j^2\right) = n\|\underline{x}\|_2^2 \ . \qquad \square$$

A diagram showing vectors of norm 1 in $\mathbb{R}^2$. Observe that when $n = 2$ we have

$$\|\underline{x}\|_\infty \le \|\underline{x}\|_2 \le \|\underline{x}\|_1 \le \sqrt{2}\|\underline{x}\|_2 \le 2\|\underline{x}\|_\infty \ .$$



3.5 Matrix norms – the axioms

For square $n \times n$ matrices we begin abstractly by stating the conditions that a matrix norm must satisfy (the axioms). The axioms include all the vector norm axioms plus one additional requirement.
A function $\|.\| : \mathbb{R}^{n\times n} \to \mathbb{R}$ is a matrix norm provided the following conditions are satisfied.

(i) $\|A\| \ge 0$ for all $A \in \mathbb{R}^{n\times n}$ with $\|A\| = 0$ if and only if $A = 0$. ($A = 0$ means A=zero matrix.) The non-negative condition.

(ii) $\|\alpha A\| = |\alpha|\,\|A\|$ for all scalars $\alpha \in \mathbb{R}$ and all $A \in \mathbb{R}^{n\times n}$. The linearity condition.

(iii) $\|A + B\| \le \|A\| + \|B\|$ for all $A, B \in \mathbb{R}^{n\times n}$. The triangle inequality condition.

(iv) $\|AB\| \le \|A\|\,\|B\|$ for all $A, B \in \mathbb{R}^{n\times n}$. The **sub-multiplicative condition**.

3.6 Commonly used matrix norms – the subordinate norms

Matrices are typically used with vectors, for example given a matrix A and a vector $\underline{x}$ we are typically interested in the product $A\underline{x}$. What we wish to know now is how 'large' can the vector $A\underline{x}$ be? The size of $A\underline{x}$ will clearly depend on the size of $\underline{x}$ but how does it depend on the direction of $\underline{x}$? If $\underline{x} \ne \underline{0}$ is a vector then we can use any of the vector norms to quantify what we mean by the size of $\underline{x}$ and the vector $A\underline{x}$. What we are interested in here is the size of the ratio

$$\frac{\|A\underline{x}\|}{\|\underline{x}\|} = \|A\underline{y}\|,$$

where

$$\underline{y} = \frac{1}{\|\underline{x}\|}\underline{x}$$

is a **unit** vector. The largest this quantity can be is the matrix norm subordinate to (induced by) the vector norm. Formally we have

$$\|A\| = \sup_{\underline{x}\neq\underline{0}} \frac{\|A\underline{x}\|}{\|\underline{x}\|} = \sup_{\|\underline{y}\|=1} \|A\underline{y}\| = \max_{\|\underline{y}\|=1} \|A\underline{y}\|,$$

where sup denotes the supremum or least upper bound. However since in this case the supremum is attained we can replace sup by max. Hence for our purposes we can take as the definition of the matrix norm the following:

$$\|A\| = \max_{\|\underline{x}\|=1} \|A\underline{x}\| \ .$$

A key property of this norm is that we have for all vectors $\underline{x}$

$$\|A\underline{x}\| \le \|A\|\|\underline{x}\| \ .$$

To correctly interpret the notation and this result you should note that when $\|A\|$ or $\|\underline{x}\|$ is written the norm that is being used depends on what is put between the double $\|$ symbols. Thus $\|A\|$ means the matrix norm of the matrix A and $\|\underline{x}\|$ means the vector norm of the vector $\underline{x}$.

In the following we will consider expressions for the matrix 1, 2 and ∞ norms corresponding to the vector 1, 2 and ∞ norms. There are also matrix norms which are not subordinate to vector norms. A commonly used norm which is not subordinate to a vector norm is the **Euclidean norm** defined by

$$\|A\|_E = \left(\sum_{i,j=1}^{n} a_{ij}^2\right)^{1/2}.$$

By noting that

$$(A^TA)_{ii} = \sum_{j=1}^{n}(A^T)_{ij}(A)_{ji} = \sum_{j=1}^{n} a_{ji}^2 \quad \text{and} \quad \mathrm{tr}(A^TA) := \sum_{i=1}^{n}(A^TA)_{ii}$$

it follows that $\|A\|_E$ is also given by

$$\|A\|_E^2 = \mathrm{tr}(A^TA).$$

(In books this norm is also sometimes known as the **Frobenius** matrix norm or **Schur** norm or **Hilbert-Schmidt** norm. You need to note here that the Euclidean matrix norm is different to the matrix 2-norm which differs from the vector case when the Euclidean vector norm and the vector 2-norm are the same.)

3.7 Verification of matrix norm axioms for subordinate norms

We now verify that the subordinate matrix norm actually satisfies the matrix norm axioms.

(i) Since for vectors $\|A\underline{x}\| \ge 0$ it follows that

$$\|A\| = \max_{\|\underline{x}\|=1} \|A\underline{x}\| \ge 0 \ .$$

If $\|A\| = 0$ then $\|A\underline{x}\| = 0$ for **all** vectors $\underline{x}$ and thus by the corresponding vector norm properties $A\underline{x} = \underline{0}$ for all vectors $\underline{x}$. The standard reasoning that now follows is that we select

$$\underline{x} = \underline{e}_i = i\text{th column of the identity matrix.}$$

With this choice

$$A\underline{x} = A\underline{e}_i = i\text{th column of the matrix } A.$$

Thus the ith column of A is the zero vector $\underline{0}$ and since this reasoning applies to all values of i we deduce that

$$A = 0 = \text{zero matrix.}$$

(ii) Linearity follows from the linearity of vector norms.

(iii) By using the triangle inequality for vectors norms we have

$$\begin{aligned} \|A+B\| &= \max_{\|\underline{x}\|=1} \|(A+B)\underline{x}\| \le \max_{\|\underline{x}\|=1} (\|A\underline{x}\| + \|B\underline{x}\|) \\ &\le \max_{\|\underline{x}\|=1} \|A\underline{x}\| + \max_{\|\underline{x}\|=1} \|B\underline{x}\| = \|A\| + \|B\|. \end{aligned}$$

The first $\le$ is the triangle inequality of the vector norm. The second $\le$ is because taking the maximum for each component separately must give an equal or larger value.

(iv) By using the matrix norm definition twice we have

$$\|AB\underline{x}\| \le \|A\|\,\|B\underline{x}\| \le \|A\|\,\|B\|\,\|\underline{x}\|$$

and hence

$$\|AB\| \le \|A\|\,\|B\|\ .$$

3.8 Expressions for the matrix norms $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$

Result.

$$\begin{aligned} \|A\|_\infty &= \max_{1\le i\le n} \sum_{j=1}^n |a_{ij}| &&= \text{maximum } \textbf{row} \text{ sum of absolute values,} \\ \|A\|_1 &= \max_{1\le j\le n} \sum_{i=1}^n |a_{ij}| &&= \text{maximum } \textbf{column} \text{ sum of absolute values,} \\ \|A\|_2 &= \left(\rho(A^TA)\right)^{1/2}, \end{aligned}$$

where

$$\begin{aligned} \rho(B) &= \textbf{spectral radius} \text{ of } B \\ &= \text{modulus of the largest eigenvalue of } B. \end{aligned}$$

Remarks

To prove the results we need to maximise $\|A\underline{x}\|_\infty$, $\|A\underline{x}\|_1$ and $\|A\underline{x}\|_2$ over appropriate sets of unit vectors in each case. Before giving all the details we begin with some observations and indicate the vectors which, as is seen in the proof, give the maximum values.

In the ∞-norm case,

$$\|A\underline{x}\|_\infty = \max_{1\le i\le n} |(A\underline{x})_i| \quad \text{and} \quad (A\underline{x})_i = \sum_{j=1}^n a_{ij}x_j \ .$$

That is $(A\underline{x})_i$ just involves the entries of A on the ith row and we have

$$(A\underline{x})_i = \sum_{j=1}^n a_{ij}x_j = \sum_{j=1}^n |a_{ij}| \quad \text{if we choose } \underline{x} \text{ such that } a_{ij}x_j = |a_{ij}| \text{ for } j = 1, \cdots, n.$$

Note that the components of $\underline{x}$ are all of magnitude 1, i.e. $\|\underline{x}\|_\infty = 1$. Hence all the row sums of absolute values can be attained if we consider $\|A\underline{x}\|_\infty$ for all vectors $\underline{x}$ satisfying $\|\underline{x}\|_\infty = 1$ and in particular the maximum of the row sums can be attained.

If $\underline{x} = \underline{e}_i$, the ith unit base vector, then $A\underline{x} = A\underline{e}_i$ is the ith column of A. Observe that when the 1-norm is considered we have

$$\|\underline{x}\|_1 = \|\underline{e}_i\|_1 = 1 \quad \text{and} \quad \|A\underline{e}_i\|_1 = i\text{th column sum of absolute values.}$$

Thus for one of the base vectors the maximum column sum is attained.

In the case of the 2-norm we have

$$\|A\underline{x}\|_2^2 = (A\underline{x})^T A\underline{x} = \underline{x}^T A^T A\underline{x} \ .$$

If $\underline{x} = \underline{v}$ is an eigenvector of A^TA normalised so that $\underline{v}^T\underline{v} = 1$ with eigenvalue λ then

$$\underline{v}^T A^T A\underline{v} = \underline{v}^T(\lambda\underline{v}) = \lambda\underline{v}^T\underline{v} = \lambda \ .$$

Thus by taking $\underline{x}$, with $\|\underline{x}\|_2 = 1$, as each of the eigenvectors of A^TA the value of $\|A\underline{x}\|_2^2$ is the corresponding eigenvalue. Hence we can attain the maximum eigenvalue of A^TA using one of these eigenvectors.

PROOF.

We now consider the details of the proof. We will first prove the ∞-norm result, then the 1-norm result and finally the 2-norm result. In the case of the ∞ and 1 norms the technique is just to make use of the definition of the matrix norm, to use the triangle inequality somewhere and to use properties of taking the maximum of sets.

$\underline{\infty\text{-norm result}}$

Let $\underline{x}$ be a vector with $\|\underline{x}\|_\infty = 1$. For the vector $A\underline{x}$ we have

$$\begin{aligned}
\|A\underline{x}\|_\infty &= \max_{1\le i\le n} |i\text{th component of } A\underline{x}| \quad \text{(definition)} \\
&= \max_{1\le i\le n} \left|\sum_{j=1}^n a_{ij}x_j\right| \le \max_{1\le i\le n} \sum_{j=1}^n |a_{ij}x_j| \quad \text{(triangle inequality)} \\
&\le \max_{1\le i\le n} \sum_{j=1}^n |a_{ij}| \quad (\text{since } |x_j| \le \|\underline{x}\|_\infty = 1).
\end{aligned}$$

To complete the proof of this case we must show that the right hand side term is attained for a particular choice of $\underline{x}$ with $\|\underline{x}\|_\infty = 1$.

Suppose the maximum row sum occurs with row m. We choose a vector $\underline{x} = (x_i)$ so that

$$a_{mj}x_j = |a_{mj}|, \quad j = 1, 2, \cdots, n,$$

e.g.

$$x_j = \begin{cases} |a_{mj}|/a_{mj}, & \text{if } a_{mj} \neq 0. \\ 1, & \text{otherwise.} \end{cases}$$

With this choice

$$(A\underline{x})_m = \sum_{j=1}^{n} |a_{mj}| = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| \ .$$

<u>1-norm result</u>

Now let $\underline{x}$ be a vector with $\|\underline{x}\|_1 = 1$. The ith component of $A\underline{x}$ is

$$\sum_{k=1}^{n} a_{ik}x_k \ .$$

Thus

$$\begin{aligned} \|A\underline{x}\|_1 &= \sum_{i=1}^{n} \left| \sum_{k=1}^{n} a_{ik}x_k \right| \le \sum_{i=1}^{n}\sum_{k=1}^{n} |a_{ik}x_k| \quad \text{(triangle inequality)} \\ &= \sum_{k=1}^{n} |x_k| \left(\sum_{i=1}^{n} |a_{ik}| \right) \le \sum_{k=1}^{n} |x_k| \left(\max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| \right) \\ &= \|\underline{x}\|_1 \left(\max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| \right) = \left(\max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| \right) \ . \end{aligned}$$

As in the ∞-norm case we must show that the term on the right hand side is attained for a particular choice of $\underline{x}$ with $\|\underline{x}\|_1 = 1$.

Suppose that the maximum column sum occurs with column m. If we take $\underline{x} = \underline{e}_m$, which is a vector with unit norm, then

$$A\underline{x} = A\underline{e}_m = m\text{th column of } A = \begin{pmatrix} a_{1m} \\ \vdots \\ a_{nm} \end{pmatrix} \ .$$

Therefore

$$\|A\underline{e}_m\|_1 = \sum_{i=1}^{n} |a_{im}| = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| \ .$$

<u>2-norm result</u>

Now let $\underline{x}$ be a vector with $\|\underline{x}\|_2 = 1$. We have

$$\|A\underline{x}\|_2^2 = (A\underline{x})^T(A\underline{x}) = \underline{x}^T A^T A\underline{x} \ .$$

The matrix A^TA is symmetric and it thus it has real eigenvalues and a complete set of orthonormal eigenvectors. (This is a result stated in the background theory section.) Let $\underline{u}_1$, $\underline{u}_2$, $\cdots$, $\underline{u}_n$ be the orthonormal eigenvectors and let λ_1, λ_2, $\cdots$, λ_n be the corresponding eigenvalues, i.e.

$$(A^TA)\underline{u}_i = \lambda_i \underline{u}_i, \quad \|\underline{u}_i\|_2^2 = \underline{u}_i^T\underline{u}_i = 1 \ .$$

Observe that

$$\lambda_i = \lambda_i \left(\underline{u}_i^T \underline{u}_i \right) = \underline{u}_i^T \left(A^T A \right) \underline{u}_i = \|A\underline{u}_i\|_2^2 \geq 0 \ .$$

Since $\underline{u}_1, \cdots, \underline{u}_n$ is an orthonormal basis for $\mathbb{R}^n$ every vector $\underline{x}$ can be represented in terms of these eigenvectors. We have

$$\underline{x} = \sum_{i=1}^{n} c_i \underline{u}_i \quad \text{with} \quad \underline{x}^T \underline{x} = \sum_{i=1}^{n} c_i^2 = 1 \ .$$

Then

$$A^T A \underline{x} = \sum_{i=1}^{n} c_i \lambda_i \underline{u}_i$$

and using the orthonormality property

$$\underline{x}^T A^T A \underline{x} = \sum_{i=1}^{n} c_i^2 \lambda_i \ .$$

Thus

$$\begin{aligned} \underline{x}^T A^T A \underline{x} &= \sum_{i=1}^{n} c_i^2 \lambda_i \leq \max_{1 \leq j \leq n} \lambda_j \sum_{i=1}^{n} c_i^2 \\ &= \max_{1 \leq j \leq n} \lambda_j = \rho(A^T A) \ . \end{aligned}$$

The right hand side is attained with $\underline{x} = \underline{u}_m$ where the index m is such that

$$\lambda_m = \max_{1 \leq j \leq n} \lambda_j = \rho(A^T A) \ . \qquad \square$$

3.9 Remarks and some subordinate matrix norm results

From previous definitions and the above result we can relatively easily compute the Euclidean norm $\|A\|_E$ and the norms $\|A\|_1$ and $\|A\|_\infty$. The computation of $\|A\|_2$ involves computing the largest eigenvalue of $A^T A$ which in general can only be done by numerical techniques. However as we show below the matrix 2-norm $\|A\|_2$ can easily be bounded and it has certain useful properties.

(1) We have the following inequality relating the spectral radius of A with the norm of A.

$$\rho(A) \leq \|A\| \ .$$

This follows since if $\underline{v}$ is an eigenvector of A which is chosen so that $\|\underline{v}\| = 1$ then

$$\lambda \underline{v} = A \underline{v}$$

and taking norms of this expression gives

$$|\lambda| = \|\lambda \underline{v}\| = \|A\underline{v}\| \leq \|A\| \, \|\underline{v}\| = \|A\| \ .$$

In the case of real symmetric positive definite matrices and the matrix 2−norm we can replace the inequality by an equality. If A is such a matrix with positive eigenvalues

$$0 < \lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$$

then the eigenvalues of $A^T A = A^2$ are

$$\lambda_n^2 \leq \lambda_{n-1}^2 \leq \cdots \leq \lambda_1^2 = \rho(A^2) \ .$$

Hence

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sqrt{\rho(A^2)} = \sqrt{\lambda_1^2} = \lambda_1 = \rho(A) \ .$$

(2) $\|I\| = 1$ for all subordinate matrix norms and $\|I\| \ge 1$ for all possible matrix norms (i.e. for non-subordinate matrix norms as well as subordinate matrix norms). This follows since

$$\|A\| = \|IA\| \le \|I\|\|A\|$$

for all matrix norms and when we have a subordinate norm

$$\|I\| = \max_{\|\underline{x}\|=1} \|I\underline{x}\| = \max_{\|\underline{x}\|=1} \|\underline{x}\| = 1 \; .$$

(3) If Q is an orthogonal matrix (i.e. $Q^TQ = QQ^T = I$) then

$$\|A\|_2 = \|QA\|_2 = \|AQ\|_2 \; .$$

This follows since from $\|A\|_2^2 = \rho(A^TA)$ we have

$$\|QA\|_2^2 = \rho((QA)^TQA) = \rho(A^TQ^TQA) = \rho(A^TA)$$

and

$$\|AQ\|_2^2 = \rho(Q^TA^TAQ) = \rho(A^TA) \; .$$

The final equality is a consequence of $Q^T(A^TA)Q$ being similar to A^TA and thus has the same eigenvalues.

There is a similar result in the case of vectors. We have

$$\|Q\underline{x}\|_2^2 = (Q\underline{x})^TQ\underline{x} = \underline{x}^TQ^TQ\underline{x} = \underline{x}^T\underline{x} = \|\underline{x}\|_2^2.$$

Multiplying by orthogonal matrices preserves the 2−norm.

(4)

$$\|A\|_2^2 \le \|A\|_1 \, \|A\|_\infty \; .$$

This follows since

$$\begin{aligned} \|A\|_2^2 = \rho(A^TA) &\le \|A^TA\|_\infty \quad \text{(from result (1))} \\ &\le \|A^T\|_\infty \, \|A\|_\infty \quad \text{(by the sub-multiplicative property of } \|.\|_\infty) \\ &= \|A\|_1 \, \|A\|_\infty \quad (\text{since } \|A^T\|_\infty = \|A\|_1) \end{aligned}$$

3.9.1 Further remarks and the MATLAB `norm` function

The norm of a matrix is a non-negative real number indicating the size of a matrix. As result (1) above shows the size cannot be less than the spectral radius of the matrix when subordinate norms are used. That is it cannot be less than the modulus of the largest eigenvalue in magnitude. In many respects the spectral radius can be considered as specifying the size of a matrix but it must be noted that it is not a norm as nearly all of the norm axioms do not hold.

There is a `norm` function in MATLAB. In MATLAB, if `x` is a vector (row or column) then `norm(x)` is the vector 2-norm of `x`. You also get the 2-norm by typing `norm(x,2)` and you get the 1-norm and ∞-norms by typing respectively `norm(x,1)` and `norm(x,inf)`. If `A` is a square matrix then `norm(A)` is the 2-norm of `A`, i.e. the spectral norm of `A`. Like in the vector case you also obtain this by typing `norm(A,2)` and you get the 1-norm and ∞-norm by typing respectively `norm(A,1)` and `norm(A,inf)`. To get the Euclidean or Frobenius norm you type `norm(A,'fro')`.

3.10 Some key results of the section

The following are a collection of points to notes from this section on vector and matrix norms.

1. $\|\underline{x}\|_2^2 = \underline{x}^T \underline{x}$, $\|\underline{x}\|_\infty = \max_i |x_i|$, $\|\underline{x}\|_1 = \sum_1^n |x_i|$.

2. The matrix norm subordinate to a vector norm is defined by

$$\|A\| = \max_{\|x\|=1} \|A\underline{x}\|.$$

   In particular $\|A\underline{x}\| \leq \|A\| \, \|\underline{x}\|$ and $\|I\| = 1$.

3. The matrix norm axioms are as follows: $\|A\| \geq 0$ with $\|A\| = 0$ if and only if $A = 0$. For all square matrices A, B and scalars α, $\|\alpha A\| = |\alpha| \|A\|$, $\|A + B\| \leq \|A\| + \|B\|$ and $\|AB\| \leq \|A\| \, \|B\|$.

4. The spectral radius of A is defined by

$$\rho(A) := \{|\lambda| : \ \lambda \text{ is an eigenvalue of } A\}.$$

   For a general square matrix A we have $\rho(A) \leq \|A\|$. For a real symmetric and positive definite matrix A we have $\|A\|_2 = \rho(A)$.

5. Expressions for $\|A\|_\infty$, $\|A\|_1$ and $\|A\|_2$ are as follows.

$$\begin{aligned} \|A\|_\infty &= \text{max. row sum of absolute values}, \quad \|A\|_1 = \|A^T\|_\infty, \\ \|A\|_2^2 &= \rho(A^T A). \end{aligned}$$

4 Perturbations of matrices and the sensitivity of linear equations

One application of norms is in quantifying the effect of perturbations. For example if A is a non-singular matrix then how 'small' must the matrix ΔA be to ensure that $A + \Delta A$ is also non-singular? Related to this question is by how much do the solutions $\underline{x}$ and $\underline{x} + \Delta \underline{x}$ of the linear systems

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

and

$$(A + \Delta A)(\underline{x} + \Delta \underline{x}) = \underline{b} + \Delta \underline{b}$$

differ? This is an important question to ask when such systems are solved on a computer using the usual floating point arithmetic as initially entering A and $\underline{b}$ on a computer may involve perturbing them respectively by ϵA and $\epsilon \underline{b}$ where ϵ is the computer's machine precision (typically about 10^{-16} when FORTRAN double precision, double in C or MATLAB are used).

In the following we will often use

$$A + \Delta A = A(I + A^{-1}\Delta A), \quad (A + \Delta A)^{-1} = (I + A^{-1}\Delta A)^{-1}A^{-1}$$

and thus we begin by considering perturbations of the identity matrix I. Also when a matrix norm is used we will always assume that it is a norm which is subordinate to a vector norm.

4.1 The matrix $(I - E)^{-1}$ when $\|E\| < 1$

Theorem. If E is an $n \times n$ matrix and $\|E\| < 1$ then the matrix $I - E$ is non-singular and

$$(I - E)^{-1} = \sum_{k=0}^{\infty} E^k = I + E + E^2 + \cdots$$

with

$$\|(I - E)^{-1}\| \le \frac{1}{1 - \|E\|} \quad \text{and} \quad \|(I - E)^{-1} - I\| \le \frac{E}{1 - \|E\|} \; .$$

PROOF.

To prove that $I - E$ is non-singular we use proof by contradiction, i.e. we suppose that $I - E$ is singular and show that this leads to a contradiction. $I - E$ being singular means that there exist $\underline{x} \ne \underline{0}$ with

$$(I - E)\underline{x} = \underline{0}, \quad \text{i.e. } \underline{x} = E\underline{x} \; .$$

But by the matrix norm properties

$$\|\underline{x}\| = \|E\underline{x}\| \le \|E\| \, \|\underline{x}\| < \|\underline{x}\|$$

since $\|E\| < 1$ and $\|\underline{x}\| > 0$. This is a contradiction. Thus the original assumption was incorrect and $I - E$ is non-singular.

Next observe that, as in the case of scalars,

$$(I - E)(I + E + E^2 \cdots + E^{m-1}) = I - E^m \; .$$

Thus

$$I + E + \cdots + E^{m-1} = (I - E)^{-1}(I - E^m) \; .$$

Now observe that by repeatedly using the sub-multiplicative property we have

$$\|E^m\| \le \|E\|^m \to 0 \quad \text{as } m \to \infty \ .$$

Hence $E^m \to 0$ as $m \to \infty$ and

$$(I-E)^{-1} = I + E + E^2 + \cdots = \sum_{k=0}^{\infty} E^k \quad \text{and} \quad (I-E)^{-1} - I = E + E^2 + \cdots = \sum_{k=1}^{\infty} E^k \ .$$

By the triangle inequality

$$\begin{aligned} \|(I-E)^{-1}(I-E^m)\| = \|I + E + \cdots + E^{m-1}\| &\le \|I\| + \|E\| + \|E\|^2 + \cdots + \|E\|^{m-1} \\ &\le \sum_{k=0}^{\infty} \|E\|^k = \frac{1}{1-\|E\|} \ . \end{aligned}$$

Similarly

$$\|(I-E)^{-1}(I-E^m) - I\| \le \sum_{k=1}^{\infty} \|E\|^k = \frac{\|E\|}{1-\|E\|} \ .$$

Since these are true for all m we have

$$\|(I-E)^{-1}\| \le \frac{1}{1-\|E\|} \quad \text{and} \quad \|(I-E)^{-1} - I\| \le \frac{\|E\|}{1-\|E\|} \ . \qquad \square$$

4.2 Definition of the condition number

In the following analysis of the linear equations problem $A\underline{x} = \underline{b}$ the **condition number** of A is an important quantity.

Definition. If A is a non-singular matrix then the **condition number** with respect to a matrix norm $\|.\|$ is given by

$$\kappa(A) = \|A\| \, \|A^{-1}\| \ .$$

When the matrix 2-norm (sometimes called the spectral norm) is used the condition number is called the **spectral condition number**.

Since for all matrix norms $1 \le \|I\|$ we have

$$1 \le \|I\| = \|AA^{-1}\| \le \|A\| \, \|A^{-1}\| = \kappa(A)$$

with $1 = \|I\|$ for matrix norms which are subordinate to vector norms. For such norms we say that a matrix is perfectly conditioned when $\kappa(A) = 1$, well conditioned when $\kappa(A)$ is 'near' 1 and badly-conditioned or ill-conditioned when $\kappa(A)$ is 'large'.

4.3 The perturbation results

In the following the aim is to consider the problems $A\underline{x} = \underline{b}$ and $(A+\Delta A)(\underline{x}+\Delta\underline{x}) = \underline{b}+\Delta\underline{b}$. This is done by first considering simpler problems corresponding to only changing the right hand side $\underline{b}$ or to only changing the matrix A. In each case we will be concerned with the relative change in the solution and thus we start by considering the size of $\underline{x}$.

Lemma. (bounds on $\|\underline{x}\|$ in $A\underline{x} = \underline{b}$) If the matrix A is non-singular, and the vector $\underline{x}$ satisfies $A\underline{x} = \underline{b}$ then

$$\frac{\|\underline{b}\|}{\|A\|} \le \|\underline{x}\| \le \|A^{-1}\| \, \|\underline{b}\|.$$

PROOF.
This follows immediately by considering the relations $\underline{b} = A\underline{x}$ and $\underline{x} = A^{-1}\underline{b}$. Taking norms gives

$$\|\underline{b}\| = \|A\underline{x}\| \le \|A\| \, \|\underline{x}\|, \quad \text{i.e. } \|\underline{x}\| \ge \frac{\|\underline{b}\|}{\|A\|},$$

and

$$\|\underline{x}\| = \|A^{-1}\underline{b}\| \le \|A^{-1}\| \|\underline{b}\|. \qquad \square$$

Lemma. (only changing $\underline{b}$) If the matrix A is non-singular, the vector $\underline{b} \ne \underline{0}$ and vectors $\underline{x}$ and $\underline{x} + \Delta\underline{x}$ satisfy

$$A\underline{x} = \underline{b} \quad \text{and} \quad A(\underline{x} + \Delta\underline{x}) = \underline{b} + \Delta\underline{b}$$

then

$$\frac{\|\Delta\underline{x}\|}{\|\underline{x}\|} \le \kappa(A)\frac{\|\Delta\underline{b}\|}{\|\underline{b}\|} \, .$$

PROOF.
First observe that $A\underline{x} = \underline{b}$ and $A(\underline{x} + \Delta\underline{x}) = \underline{b} + \Delta\underline{b}$ implies that

$$A\Delta\underline{x} = \Delta\underline{b} \quad \text{and} \quad \Delta\underline{x} = A^{-1}\Delta\underline{b} \, .$$

Therefore

$$\|\Delta\underline{x}\| = \|A^{-1}\Delta\underline{b}\| \le \|A^{-1}\| \, \|\Delta\underline{b}\| \, .$$

Since we wish to obtain an upper bound for $\|\Delta\underline{x}\|/\|\underline{x}\|$ we require a lower bound for $\|\underline{x}\|$ and by the previous lemma we have

$$\|\underline{x}\| \ge \frac{\|\underline{b}\|}{\|A\|} \, .$$

Thus

$$\frac{\|\Delta\underline{x}\|}{\|\underline{x}\|} \le \|A\| \, \|A^{-1}\| \, \frac{\|\Delta\underline{b}\|}{\|\underline{b}\|} = \kappa(A)\frac{\|\Delta\underline{b}\|}{\|\underline{b}\|} \, . \qquad \square$$

Lemma. (only changing A) If the matrix A is non-singular, the vector $\underline{b} \ne \underline{0}$, ΔA is a matrix such that $\|A^{-1}\Delta A\| < 1$ and $\underline{x}$ and $\Delta\underline{x}$ are vectors satisfying

$$A\underline{x} = \underline{b} \quad \text{and} \quad (A + \Delta A)(\underline{x} + \Delta\underline{x}) = \underline{b}$$

then $A + \Delta A$ is non-singular and

$$\frac{\|\Delta\underline{x}\|}{\|\underline{x}\|} \le \frac{r}{1-r}, \quad \text{where} \quad r = \kappa(A)\frac{\|\Delta A\|}{\|A\|} \, .$$

PROOF.

Observe that $A+\Delta A = A(I + A^{-1}\Delta A)$ is non-singular since A is non-singular and $\|A^{-1}\Delta A\| < 1$ implies that $I + A^{-1}\Delta A$ is non-singular. (The product of two non-singular matrices is non-singular.)[6] Rearranging we have

$$\underline{x} + \Delta\underline{x} = (A+\Delta A)^{-1}\underline{b} = (I-E)^{-1}A^{-1}\underline{b} = (I-E)^{-1}\underline{x} \quad \text{where } E = -A^{-1}\Delta A,$$

and thus

$$\Delta\underline{x} = \left((I-E)^{-1} - I\right)\underline{x} \ .$$

Taking norms and using the result of the theorem on page 4.1 we have

$$\frac{\|\Delta\underline{x}\|}{\|\underline{x}\|} \le \|(I-E)^{-1} - I\| \le \frac{\|E\|}{1-\|E\|} \le \frac{r}{1-r}$$

since

$$\|E\| = \|-E\| = \|A^{-1}\Delta A\| \le \|A^{-1}\|\,\|\Delta A\| = \kappa(A)\frac{\|\Delta A\|}{\|A\|} = r \ . \qquad \square$$

Theorem. If a matrix A is non-singular, a vector $\underline{b} \ne \underline{0}$, a matrix ΔA satisfies $\|A^{-1}\Delta A\| < 1$ and vectors $\underline{x}$ and $\underline{x}+\Delta\underline{x}$ satisfy

$$A\underline{x} = \underline{b} \quad \text{and} \quad (A+\Delta A)(\underline{x}+\Delta\underline{x}) = (\underline{b}+\Delta\underline{b})$$

then

$$\frac{\|\Delta\underline{x}\|}{\|\underline{x}\|} \le \frac{\kappa(A)}{1-\kappa(A)\dfrac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|\underline{b}\|}\right) \ .$$

Proof. (This proof is not examinable.)

The proof involves the two given systems and the intermediate system

$$A\underline{y} = \underline{b} + \Delta\underline{b} \ .$$

We can use the previous two lemmas to compare $\underline{y}$ with $\underline{x}$ and $\underline{x}+\Delta\underline{x}$. From the lemmas we have

$$\frac{\|\underline{x}-\underline{y}\|}{\|\underline{x}\|} \le \kappa(A)\frac{\|\Delta\underline{b}\|}{\|\underline{b}\|}, \qquad \frac{\|\underline{x}+\Delta\underline{x}-\underline{y}\|}{\|\underline{y}\|} \le \frac{r}{1-r},$$

where as before

$$r = \frac{\kappa(A)\|\Delta A\|}{\|A\|} \ .$$

Observe that

$$\Delta\underline{x} = (\underline{x}+\Delta\underline{x}-\underline{y}) + (\underline{y}-\underline{x})$$

and thus

$$\begin{aligned}\|\Delta\underline{x}\| &\le \|\underline{x}+\Delta\underline{x}-\underline{y}\| + \|\underline{x}-\underline{y}\| \\ &\le \frac{r}{1-r}\|\underline{y}\| + \kappa(A)\frac{\|\Delta\underline{b}\|}{\|\underline{b}\|}\|\underline{x}\| \ .\end{aligned}$$

[6] From the result $\det(AB) = (\det A)(\det B)$, $\det A \ne 0$ and $\det B \ne 0$ implies that $\det(AB) \ne 0$ and as a consequence AB is non-singular.

To complete the proof we must relate $\underline{y}$ with $\underline{x}$ in order to bound $\|\underline{y}\|$ in terms of $\|\underline{x}\|$. We have

$$\underline{y} = \underline{x} + (\underline{y} - \underline{x})$$

and hence

$$\|\underline{y}\| = \|\underline{x} + (\underline{y} - \underline{x})\| \le \|\underline{x}\| + \|\underline{x} - \underline{y}\| \ .$$

Thus

$$\|\underline{y}\| \le \|\underline{x}\| + \kappa(A)\frac{\|\Delta \underline{b}\|}{\|\underline{b}\|}\|\underline{x}\| = \left(1 + \kappa(A)\frac{\|\Delta \underline{b}\|}{\|\underline{b}\|}\right)\|\underline{x}\|$$

Combining the results gives

$$\begin{aligned}
(1-r)\|\Delta \underline{x}\| &\le r\left(1 + \kappa(A)\frac{\|\Delta \underline{b}\|}{\|\underline{b}\|}\right)\|\underline{x}\| + (1-r)\kappa(A)\frac{\|\Delta \underline{b}\|}{\|\underline{b}\|}\|\underline{x}\| \\
(1-r)\frac{\|\Delta \underline{x}\|}{\|\underline{x}\|} &\le r + \kappa(A)\frac{\|\Delta \underline{b}\|}{\|\underline{b}\|} = \kappa(A)\left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta \underline{b}\|}{\|\underline{b}\|}\right) \ .
\end{aligned}$$

□

4.4 Remarks

The above results show that when $\kappa(A)$ is large we cannot expect to solve a linear system $A\underline{x} = \underline{b}$ as accurately as when $\kappa(A)$ is near 1. With usual computer floating point arithmetic with a machine precision of about $\epsilon = 10^{-16}$ we cannot expect to solve a linear system $A\underline{x} = \underline{b}$ on such a computer by <u>any</u> method when $\kappa(A)$ is of the order of $1/\epsilon = 10^{16}$ or when $\kappa(A)$ is larger than this.

To compute the condition number accurately for a given norm involves the computation of A^{-1} which requires more computation than is involved in solving $A\underline{x} = \underline{b}$. Thus in some respects the condition number is of more theoretical importance than it is a practical number to calculate. However methods for solving $A\underline{x} = \underline{b}$ can often be adapted to also give an estimate for $\kappa(A)$ without too much additional computation and in practice you only need to know roughly the size of $\kappa(A)$.

In MATLAB you can compute a condition number by typing `cond(A)`, which gives the spectral condition number of `A` when `A` is a square matrix.

4.5 Some properties of condition numbers

(1) For a real symmetric positive definite matrix A with positive eigenvalues

$$0 < \lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_1$$

the eigenvalues of $A^T A = A^2$ are

$$\lambda_n^2 \le \lambda_{n-1}^2 \le \cdots \le \lambda_1^2 = \rho(A^2)$$

and the eigenvalues of $(A^{-1})^T A^{-1} = (A^{-1})^2$ are

$$0 < \frac{1}{\lambda_1^2} \le \frac{1}{\lambda_2^2} \le \cdots \le \frac{1}{\lambda_n^2} = \rho\left((A^{-1})^2\right) \ .$$

Therefore

$$\|A\|_2 = \lambda_1 \quad \text{and} \quad \|A^{-1}\|_2 = \frac{1}{\lambda_n} \ .$$

Thus

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\text{maximum eigenvalue of } A}{\text{minimum eigenvalue of } A} \ .$$

(2) If Q is an orthogonal matrix then the result

$$\|A\|_2 = \|AQ\|_2 = \|QA\|_2$$

implies

$$\kappa_2(A) = \kappa_2(AQ) = \kappa_2(QA) \ .$$

This is a **stability** property of orthogonal matrices.

4.6 Some key results of the section

Much of this section involves detailed results. There are however a few definitions and results which can be summarised here.

1. When A^{-1} exists we write $A + \Delta A = A(I + A^{-1}\Delta)$ and when $\|E\| < 1$ the matrix $I - E$ is non-singular and we have

$$\|(I - E)^{-1}\| \le \frac{1}{1 - \|E\|}.$$

   Thus $A + \Delta A$ is non-singular when $\|A^{-1}\Delta A\| < 1$.

2. The sensitivity results depend on $\kappa(A) := \|A\| \, \|A^{-1}\| \ge 1$. Note that $\kappa(I) = 1$ and $\kappa_2(Q) = 1$ when Q is an orthogonal matrix.

3. If A is real symmetric and positive definite matrix with eigenvalues $0 < \lambda_n \le \cdots \lambda_2 \le \lambda_1$ then

$$\kappa_2(A) := \frac{\lambda_1}{\lambda_n}.$$

5 The least squares problem via the normal equations

In many situations the linear equations problem

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

cannot be solved uniquely in the usual sense. For example if there are more equations than unknowns then the system is said to be **over-determined** and in general we cannot simultaneously satisfy all of them. That is if A is an $m \times n$ matrix with $m > n$ and $\underline{b}$ is an $m \times 1$ column vector then there is in general no solution of $A\underline{x} = \underline{b}$. In such cases we consider instead how 'nearly' we can satisfy the equations. Near in this context means trying to find a vector $\underline{x}$ which satisfies

$$\|A\underline{x} - \underline{b}\| = \min_{\underline{y}} \|A\underline{y} - \underline{b}\|$$

for some vector norm $\|.\|$. The solution or solutions that you obtain will depend on which vector norm is used. By far the simplest norm to use is the 2-norm and in this case a vector $\underline{x}$ which satisfies

$$\|A\underline{x} - \underline{b}\|_2 = \min_{\underline{y}} \|A\underline{y} - \underline{b}\|_2$$

is said to be 'a least squares solution of $A\underline{x} = \underline{b}$'.

Let

$$\begin{aligned} \phi(\underline{y}) &= \|A\underline{y} - \underline{b}\|_2^2 = (A\underline{y} - \underline{b})^T(A\underline{y} - \underline{b}) \\ &= \underline{y}^T A^T A\underline{y} - 2\underline{y}^T A^T \underline{b} + \underline{b}^T \underline{b} \ . \end{aligned}$$

This non-negative quadratic in n variables has a minimum when

$$\frac{\partial \phi}{\partial y_i} = 0, \quad i = 1, 2, \cdots, n \ .$$

Since

$$\frac{\partial \underline{y}}{\partial y_i} = \frac{\partial}{\partial y_i}\begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = \underline{e}_i$$

by the product rule of differentiation we have

$$\frac{\partial \phi}{\partial y_i} = \underline{e}_i^T A^T A\underline{y} + \underline{y}^T A^T A\underline{e}_i - 2\underline{e}_i^T A^T \underline{b} = 2\underline{e}_i^T (A^T A\underline{y} - A^T \underline{b}) \ .$$

If the minimum occurs at $\underline{y} = \underline{x}$ then

$$\underline{e}_i^T (A^T A\underline{x} - A^T \underline{b}) = i\text{th component of } (A^T A\underline{x} - A^T \underline{b}) = 0 \ .$$

Since this is true for $i = 1, 2, \cdots, n$ then $\underline{x}$ satisfies

$$A^T A\underline{x} = A^T \underline{b},$$

which are the **normal equations**. That is although in general $A\underline{x} - \underline{b} \neq \underline{0}$ we do get a vector for which $A^T(A\underline{x} - \underline{b}) = \underline{0}$.

We now consider the matrix A^TA. We have

$$\underline{v}^T(A^TA)\underline{v} = \|A\underline{v}\|_2^2 \geq 0 \ .$$

If $A\underline{v} = \underline{0}$ then a linear combination of the columns of A gives the zero vector, i.e. if

$$A = (\underline{a}_1, \cdots, \underline{a}_n)$$

then

$$\underline{0} = A\underline{v} = (\underline{a}_1, \cdots, \underline{a}_n)\begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = v_1\underline{a}_1 + \cdots + v_n\underline{a}_n \ .$$

If the columns of A are linearly independent then this is only possible if $\underline{v} = \underline{0}$. Hence if the columns of A are linearly independent then the matrix A^TA is a positive definite symmetric matrix. Another way of stating this is that if

$$\begin{aligned} \operatorname{rank} A &= \text{number of linearly independent columns} \\ &= \text{number of linearly independent rows} = n \end{aligned}$$

then A^TA is a positive definite symmetric matrix.

5.1 An alternative derivation of the normal equations

There is an alternative way of showing that the vector $\underline{x}$ satisfying $A^TA\underline{x} = A^T\underline{b}$ minimises the quadratic $\phi(\underline{y})$. If $\underline{d}$ is an arbitrary vector then consider

$$\phi(\underline{x} + \underline{d}) - \phi(\underline{x}) \ .$$

If $\underline{x}$ is the global minimum of ϕ then the difference must be positive for all $\underline{d} \neq \underline{0}$. We show this directly as follows.

$$\phi(\underline{x} + \underline{d}) - \phi(\underline{x}) = (\underline{x} + \underline{d})^TA^TA(\underline{x} + \underline{d}) - 2(\underline{x} + \underline{d})^TA^T\underline{b} - \underline{x}^TA^TA\underline{x} + 2\underline{x}^TA^T\underline{b} \ .$$

To simplify note that the quadratic terms give

$$(\underline{x} + \underline{d})^TA^TA(\underline{x} + \underline{d}) - \underline{x}^TA^TA\underline{x} = 2\underline{d}^TA^TA\underline{x} + \underline{d}^TA^TA\underline{d} = 2\underline{d}^TA^T\underline{b} + \underline{d}^TA^TA\underline{d}$$

and the linear terms give

$$-2(\underline{x} + \underline{d})^TA^T\underline{b} + 2\underline{x}^TA^T\underline{b} = -2\underline{d}^TA^T\underline{b} \ .$$

Thus

$$\phi(\underline{x} + \underline{d}) - \phi(\underline{x}) = \underline{d}^TA^TA\underline{d} = \|A\underline{d}\|_2^2 \geq 0 \ .$$

5.2 The method of normal equations

Given $A \in \mathbb{R}^{m\times n}$ and $\underline{b} \in \mathbb{R}^m$ we obtain $\underline{x}$ by the following steps.

(i) Compute the lower triangular part of $C = A^TA$ and compute the vector $\underline{d} = A^T\underline{b}$.

(ii) Compute the Cholesky factorization $C = LL^T$.

(iii) Solve $LL^T\underline{x} = \underline{d}$ by solving $L\underline{y} = \underline{d}$ by forward substitution followed by $L^T\underline{x} = \underline{y}$ by backward substitution.

5.3 Sensitivity

The spectral condition number of A^TA is

$$\kappa_2(A^TA) = \|A^TA\|_2\,\|(A^TA)^{-1}\|_2 = \frac{\text{Largest eigenvalue of } A^TA}{\text{Smallest eigenvalue of } A^TA}\ .$$

We show in the next section, concerned with the QR factorization of a matrix, that we can actually solve the least squares problem by a method which leads to a system of equations which has a spectral condition number which is the square root of this quantity. This different method is hence more stable than the normal equations approach just described but it does have the disadvantage of requiring more computation.

5.4 An example using the normal equations approach

The following was part of question 3 of the 1995 MA251S paper.

By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} 1 & 1 & 3 \\ -1 & 2 & -1 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{pmatrix} \quad \text{and} \quad \underline{b} = \begin{pmatrix} 1 \\ 3 \\ 3 \\ -1 \end{pmatrix}.$$

The least squares solution satisfies the normal equations $A^TA\underline{x} = A^T\underline{b}$ and thus we must construct the matrix $C = A^TA$ and the vector $\underline{d} = A^T\underline{b}$. Since

$$\begin{aligned} (A^TA)_{ij} &= (i\text{th row of } A^T)(j\text{th column of } A) \\ &= (i\text{th column of } A)^T(j\text{th column of } A) \\ &= \text{inner product of columns } i \text{ and } j \text{ of } A. \end{aligned}$$

Similarly when we compute the Cholesky factorization $C = L\,L^T$ we have

$$\begin{aligned} (C)_{ij} = (L\,L^T)_{ij} &= (i\text{th row of } L^T)(j\text{th column of } L^T) \\ &= (i\text{th row of } L)^T(j\text{th row of } L)^T \\ &= \text{inner product of rows } i \text{ and } j \text{ of } L. \end{aligned}$$

Hence

$$C = A^TA = \begin{pmatrix} 4 & 2 & 8 \\ 2 & 10 & 7 \\ 8 & 7 & 18 \end{pmatrix} \quad \text{and} \quad \underline{d} = \begin{pmatrix} 0 \\ 12 \\ 4 \end{pmatrix}.$$

For the Cholesky factorization $C = LL^T$ we require

$$\begin{pmatrix} 4 & 2 & 8 \\ 2 & 10 & 7 \\ 8 & 7 & 18 \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix}.$$

Thus for the 1st row of L we have

$$c_{11} = 4 = l_{11}^2, \quad l_{11} = 2\ .$$

For the 2nd row of L we have

$$\begin{array}{lllll} c_{21} & = & 2 & = & l_{21}l_{11}, & l_{21}=1, \\ c_{22} & = & 10 & = & l_{21}^2+l_{22}^2 = 1+l_{22}^2, & l_{22}=3. \end{array}$$

For the 3rd row of L we have

$$\begin{array}{llllll} c_{31} & = & 8 & = & l_{31}l_{11}, & l_{31}=4, \\ c_{32} & = & 7 & = & l_{31}l_{21}+l_{32}l_{22} = 4+3l_{32}, & l_{32}=1, \\ c_{33} & = & 18 & = & l_{31}^2+l_{32}^2+l_{33}^2 = 16+1+l_{33}^2, & l_{33}=1. \end{array}$$

Thus

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 3 & 0 \\ 4 & 1 & 1 \end{pmatrix} \quad \text{and} \quad L^T = \begin{pmatrix} 2 & 1 & 4 \\ 0 & 3 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

We solve $LL^T\underline{x} = \underline{d}$ by solving $L\underline{y} = \underline{d}$ followed by $L^T\underline{x} = \underline{y}$.

$L\underline{y} = \underline{d}$ gives
$y_1 = 0$.
$3y_2 = 12$, $y_2 = 4$.
$y_3 = 4 - y_2 = 0$.

$L^T\underline{x} = \underline{y}$ gives
$x_3 = 0$.
$3x_2 = 4$, $x_2 = 4/3$.
$2x_1 = -x_2 = -4/3$, $x_1 = -2/3$.

You can confirm this in MATLAB by the instructions

```
A=[1 1 3;
  -1 2 -1;
   1 2 2;
   1 1 2];
b=[1 3 3 -1]';
x=A\b;
disp(x');
```

Note that the meaning of \ in MATLAB depends on the context. `A\b` means solve a linear system when `A` is a square matrix. When `A` has more rows than columns, as in this example, it means compute the least squares solution.

5.5 Some key results of the section

The context is $A \in \mathbb{R}^{m\times n}$ with $m \ge n$ and $\underline{b} \in \mathbb{R}^m$.

1. The least squares problem: Find $\underline{x} \in \mathbb{R}^n$, such that

$$\|A\underline{x} - \underline{b}\|_2 = \min_{\underline{y}} \|A\underline{y} - \underline{b}\|_2.$$

2. The required vector $\underline{x}$ satisfies the normal equations $A^TA\underline{x} = A^T\underline{b}$.

   The matrix A^TA is positive definite if the columns of A are linearly independent and such a matrix has a Cholesky factorization. Obtaining $\underline{x}$ by constructing the normal equations and solving these equations by constructing a Cholesky factorization is known as the method of normal equations.

6 Special orthogonal matrices, the QR factorization of a matrix and least squares

In the Gauss elimination process/LU factorization procedure the reduction of a matrix A to upper triangular form (sometimes also called right triangular form, which is the reason for using the letter R in this context) was achieved by pre-multiplying by a finite sequence of Gauss transformation matrices

$$M_k = I - \underline{m}_k\, \underline{e}_k^T, \quad \underline{m}_k = \begin{pmatrix} 0 \\ \vdots \\ m_{k+1,k} \\ \vdots \\ m_{nk} \end{pmatrix} .$$

The objective now is to show that the reduction to upper triangular form can also be achieved by a sequence of certain 'special' orthogonal matrices.

6.1 Householder matrices

(Householder matrices are named after Alston Householder who died on July 4, 1993 at the age of 89.)

$$P = P(\underline{v}) = I - 2\underline{v}\,\underline{v}^T, \quad \underline{v}^T\underline{v} = 1 .$$

If $\underline{v}$ is an $n \times 1$ vector then the normalization $\underline{v}^T\underline{v} = 1$ means that there are $n-1$ independent components.

6.2 Givens' matrices

$$G(i,j,\theta) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & s & & \\ & & -s & c & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \quad \text{where } c = \cos(\theta),\ s = \sin(\theta).$$

$G(i,j,\theta)$ differs from I only in the 4 entries shown. For given i and j there is only 1 independent component, i.e. θ.

6.3 Properties of P and G

(1)
$$P^2 = (I - 2\underline{v}\,\underline{v}^T)(I - 2\underline{v}\,\underline{v}^T) = I - 4\underline{v}\,\underline{v}^T + 4\underline{v}(\underline{v}^T\underline{v})\underline{v}^T = I,$$
since $\underline{v}^T\underline{v} = 1$. Hence $P = P^T = P^{-1}$ is symmetric and orthogonal. P has a geometric interpretation as a reflection, i.e. $\underline{x}$ and $P\underline{x}$ are mirror images in a certain hyperplane (which is defined by the vector $\underline{v}$) in n-dimensional space.

(2)
$$(G(i,j,\theta))^T G(i,j,\theta) = I, \quad \text{i.e. } G(i,j,\theta) \text{ is orthogonal.}$$

This follows since the columns of $G(i,j,\theta)$ are mutually orthonormal, i.e. they all have Euclidean length of 1 and they are all mutually orthogonal. G has a geometrical interpretation as a rotation of 2 of components in n-dimensional space.

We will not actually consider Givens' matrices much in this module.

6.4 Choice of $\underline{v}$ for reduction

If $\underline{x} = \underline{0}$ then the vector is already 'reduced'. Thus we will suppose that $\underline{x} \neq \underline{0}$. In this case the aim is to choose $\underline{v}$ so that $P(\underline{v})\underline{x}$ only has 1 non-zero component. As P is orthogonal the vectors $P\underline{x}$ and $\underline{x}$ have the same 2-norm and so suppose that

$$P\underline{x} = (I - 2\underline{v}\,\underline{v}^T)\underline{x} = \begin{pmatrix} \beta\|\underline{x}\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \beta\|\underline{x}\|_2\,\underline{e}_1, \quad \text{where } \beta = \pm 1 .$$

Thus

$$\underline{x} - 2(\underline{v}^T\underline{x})\underline{v} = \beta\|\underline{x}\|_2\,\underline{e}_1$$

and hence

$$2(\underline{v}^T\underline{x})\underline{v} = \underline{x} - \beta\|\underline{x}\|_2\,\underline{e}_1 .$$

In words we have that a scalar multiplied by $\underline{v}$ is equal to the vector on the right hand side of the above equation, that is the direction of $\underline{v}$ is determined provided the right hand side is not the zero vector. As we have a choice of whether $\beta = 1$ or $\beta = -1$ we can select to choose it so that the vector on the right hand side is non-zero by selecting from the 2 choices the one which gives the greatest 2-norm. This choice is desirable computationally as it avoids any cancellation error in the construction of $\underline{v}$. If we denote the resulting vector by $\underline{y}$ and we take

$$\underline{v} = \frac{1}{\|\underline{y}\|_2}\underline{y}$$

then we have the desired effect that

$$P(\underline{v})\underline{x} = \beta\|\underline{x}\|_2\,\underline{e}_1 .$$

We have just shown how to 'reduce' a vector by a single Householder matrix. The next task is to indicate how a sequence of such matrices can be used to reduce a matrix. The key point is to choose the matrices so as not to undo previous reduction steps. When Gauss elimination matrices were used the property that we do not undo previous reduction steps was obtained by taking the first k entries of $\underline{m}_k$ as zero. In the Householder reduction we similarly take

$$\underline{v}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ v_{kk} \\ \vdots \\ v_{nk} \end{pmatrix} = \begin{pmatrix} \underline{0} \\ \underline{\tilde{v}}_k \end{pmatrix} \quad \text{where} \quad \underline{\tilde{v}}_k = \begin{pmatrix} v_{kk} \\ \vdots \\ v_{nk}, \end{pmatrix}$$

i.e. the first $k-1$ components are 0. When this is done then the first $k-1$ rows and columns of $\underline{v}_k\,\underline{v}_k^T$ are zero and $P(\underline{v}_k)$ has the partitioned form

$$P(\underline{v}_k) = \begin{pmatrix} I_{k-1} & O \\ O & P(\tilde{\underline{v}}_k) \end{pmatrix},$$

where I_{k-1} is the $(k-1)\times(k-1)$ identity matrix and where $P(\tilde{\underline{v}}_k)$ is a smaller Householder matrix. If a matrix $A^{(k-1)}$ has zero entries in columns 1 to $k-1$ below the diagonal then the product $P(\underline{v}_k)A^{(k-1)}$ has the same entries in these columns. By choosing $\tilde{\underline{v}}_k$ to eliminate the entries below the diagonal in column k it is hence possible in the case $A \in \mathbb{R}^{m\times n}$, $m \ge n$, to generate a sequence of matrices $A^{(0)} = A$, $A^{(1)}$, $\cdots$, $A^{(n)}$ by

$$A^{(k)} = P(\underline{v}_k)A^{(k-1)}, \quad k = 1, 2, \cdots, n,$$

where $A^{(k)}$ has been reduced in columns 1 to k below the diagonal. The vector $\tilde{\underline{v}}_k$, which is part of $\underline{v}_k$, which achieves the reduction depends on the entries

$$\begin{pmatrix} a_{kk}^{(k-1)} \\ \vdots \\ a_{mk}^{(k-1)} \end{pmatrix}.$$

In the above it is assumed that if a vector is already appropriately reduced then the step is skipped and if $m = n$ then we can stop with $A^{(n-1)}$. The result of the reduction is that we have an upper triangular matrix R where

$$R = A^{(n)} = P_n \cdots P_1 A, \quad \text{where } P_k = P(\underline{v}_k)\ .$$

Thus

$$A = (P_n \cdots P_1)^{-1}R = (P_1 \cdots P_n)R = QR,$$

where

$$Q = P_1 \cdots P_n$$

is an orthogonal matrix.

6.4.1 Remarks

(1) The above reduction can always be done, i.e. there is no restriction to A being non-singular or even being a square matrix. If A has at least as many rows as columns then R having zero entries below the diagonal means that it has the partitioned form

$$R = \begin{pmatrix} R_1 \\ O \end{pmatrix},$$

where R_1 is a square upper triangular matrix.

(2) The computation of the product $(I - 2\underline{v}_k\underline{v}_k^T)A^{(k-1)}$ involves about twice as many flops as does the computation of the product $(I - 2\underline{m}_k\underline{e}_k^T)A^{(k-1)}$ in the LU factorization. Hence computing a QR factorization involves about twice as many flops as computing a LU factorization.

(3) The reduction can also be done using Givens' matrices but this involves more flops than the Householder approach. Givens' matrices are used when only a few entries are to be reduced.

(4) In MATLAB the command

```
[Q R]=qr(A);
```

takes the input matrix `A` and returns the orthogonal matrix `Q` and an upper triangular matrix `R` such that, in MATLAB notation, `A=Q*R`.

Unlike in the case of Gauss elimination, it is nearly always messy to attempt to illustrate the construction of a QR factorization by hand calculation. The reason for this is that the process of normalising a vector requires square roots and thus you cannot easily set up examples in which the numbers are manageable by hand calculations. QR calculations are intended for computers only as part of some algorithm.

With Matlab's ability to handle matrix operations the whole reduction process using Householder matrices can be achieved by the following function file.

```
function [Q,R]=myqr(A)
%  function [Q,R]=myqr(A) computes a Q*R decomposition of A.
%% On input:  A is an mxn matrix with m>=n
%% On output: Q is an orthogonal mxm matrix, R=upper triangular

%% get the size of A and set initial R and Q matrices
[m,n]=size(A);  if m<n whos A; error('A has the wrong shape'); end
R=A;
Q=eye(m,m);

%% reduce R column by column to get the upper triangular form
for k=1:n
  b=R(k:m,k); nb=norm(b); y=b;                %% b=part of the kth column

  if nb>0
  beta=1; if y(1)>0 beta=-1; end              %% beta=1 or -1
  y(1)=y(1)-beta*nb;
  v=y/norm(y);                                %% the Householder vector
  R(k:m,k:n)=R(k:m,k:n)-2*v*(v'*R(k:m,k:n));%% update R
  Q(k:m,k:m)=Q(k:m,k:m)-2*(Q(k:m,k:m)*v)*v';%% update Q
  end

end
```

---

(5) By the Householder approach it was not possible to guarantee that the diagonal entries of R are all positive. However if a Householder step gives a negative diagonal entry then the use of the appropriate Givens' matrix with $\theta = \pi$ will make the diagonal entry positive. Thus it is always possible to factorize A as QR with R having non-negative diagonal entries.

6.5 Some properties and applications

(1) **Stability**

$$\|A^T A\|_2 = \rho(A^T A) = \rho(R^T Q^T Q R) = \rho(R^T R) = \rho(R_1^T R_1) = \|R_1\|_2^2$$

where R_1 is the square upper triangular part of R. The size of R_1 in the 2-norm is the square root of the size of $\|A^TA\|_2$.

(2) **Solving linear equations**
In the case of a square matrix A and the system

$$A\underline{x} = \underline{b} \quad \text{and} \quad A = QR$$

we have

$$Q\underline{y} = \underline{b}, \quad \underline{y} = Q^T\underline{b} \quad \text{(matrix multiplication to get } \underline{y}\text{)}$$

followed by

$$R\underline{x} = \underline{y} \quad \text{(solve for } \underline{x} \text{ by back substitution).}$$

6.6 The QR approach to the least squares problem

Recall that the least squares problem involves a matrix $A \in \mathbb{R}^{m,n}$ which has more rows than columns. The method of normal equations then involved constructing the square matrix A^TA. The QR approach differs from this and avoids the construction of A^TA. Instead a QR factorization of A is first computed, i.e.

$$A = QR \quad \text{with} \quad R = \begin{pmatrix} R_1 \\ O \end{pmatrix},$$

where Q is an orthogonal matrix and R_1 is an $n \times n$ upper triangular matrix. Consider

$$A\underline{y} - \underline{b} = QR\underline{y} - \underline{b} = Q(R\underline{y} - Q^T\underline{b}).$$

From the property of orthogonal matrices in preserving the 2-norm we have

$$\|A\underline{y} - \underline{b}\|_2^2 = \|R\underline{y} - Q^T\underline{b}\|_2^2 .$$

Now

$$R\underline{y} = \begin{pmatrix} R_1 \\ O \end{pmatrix} \underline{y} = \begin{pmatrix} R_1\underline{y} \\ \underline{0} \end{pmatrix} .$$

We similarly partition $Q^T\underline{b}$, i.e.

$$Q^T\underline{b} = \begin{pmatrix} \underline{c} \\ \underline{z} \end{pmatrix} .$$

Then

$$\begin{aligned} \|R\underline{y} - Q^T\underline{b}\|_2^2 &= \text{The sum of the squares of the components,} \\ &= \|R_1\underline{y} - \underline{c}\|_2^2 + \|\underline{z}\|_2^2 \geq \|\underline{z}\|_2^2 . \end{aligned}$$

If the minimum value occurs at $\underline{y} = \underline{x}$ then $\underline{x}$ satisfies

$$R_1\underline{x} = \underline{c} .$$

To compare this with the normal equations approach observe that if $A = QR$ then

$$C = A^TA = R^TQ^TQR = R^TR = (R_1^T \ O^T) \begin{pmatrix} R_1 \\ O \end{pmatrix} = R_1^TR_1,$$

and

$$\underline{d} = A^T\underline{b} = R^TQ^T\underline{b} = (R_1^T \ O^T) \begin{pmatrix} \underline{c} \\ \underline{z} \end{pmatrix} = R_1^T\underline{c} .$$

The main difference is that in the normal equations approach $R_1 = L^T$ is obtained by computing a Cholesky factorization of $C = A^T A$. In the QR approach R_1 is part of R in the QR factorization of A. An obvious question is which is the better method? The normal equations approach requires less computation and thus this is the method usually used. However the QR approach is more stable as we will now show.

6.7 Sensitivity of least squares with the QR approach

As we have just shown that $R_1^T R_1 = A^T A$ and with the QR approach the only system to be solved is $R_1 \underline{x} = \underline{c}$, it is the condition number of R_1 that we have to consider. Now

$$\|R_1\|_2^2 = \rho(R_1^T R_1) = \rho(A^T A) = \text{The largest eigenvalue of } A^T A,$$

and

$$\|R_1^{-1}\|_2^2 = \frac{1}{\text{The smallest eigenvalue of } A^T A} .$$

Thus

$$(\kappa_2(R_1))^2 = \|R_1\|_2^2 \, \|R_1^{-1}\|_2^2 = \frac{\text{Largest eigenvalue of } A^T A}{\text{Smallest eigenvalue of } A^T A} = \kappa_2(A^T A) .$$

That is the spectral condition number of R_1 is the square root of the condition number of $A^T A = R_1^T R_1$. Hence if $\kappa_2(A^T A) = 10^{12}$ then $\kappa_2(R_1) = 10^6$. The QR approach is preferred if $A^T A$ is very badly conditioned.

6.8 Some key results of the section

The context is $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and $\underline{b} \in \mathbb{R}^m$.

1. $$P = I - 2\underline{v}\,\underline{v}^T \quad \text{with } \|\underline{v}\|_2^2 = \underline{v}^T \underline{v} = 1$$

   is a Householder matrix. It satisfies $P^2 = I$ so that $P = P^T = P^{-1}$.

2. Using an appropriate sequence of Householder matrices it is always possible to construct the factorization

   $$A = QR \quad \text{with} \quad R = \begin{pmatrix} R_1 \\ O \end{pmatrix},$$

   where Q is an orthogonal matrix and R_1 is a square upper triangular matrix.

3. The QR approeach to solving the least squares problem involves noting that

   $$\|A\underline{x} - \underline{b}\|_2^2 = \|R\underline{x} - Q^T \underline{b}\|_2^2 = \|R_1 \underline{x} - \underline{c}\|_2^2 + \|\underline{z}\|_2^2 \quad \text{where} \quad Q^T \underline{b} = \begin{pmatrix} \underline{c} \\ \underline{z} \end{pmatrix}.$$

   You just solve $R_1 \underline{x} = \underline{c}$. Since $A^T A = R^T R = R_1^T R_1$, R_1 is a Cholesky factor of $A^T A$ when $(R_1)_{ii} > 0$, $i = 1, 2, \cdots, n$.

7 Iterative methods for linear systems

7.1 Background

In numerical analysis the important topic of solving linear equations is unusual in that it is a problem for which an exact solution can be obtained in a finite number of steps when exact arithmetic is used. Methods such as Gauss elimination or its equivalent LU factorization or alternatively the QR factorization technique all have this property and they are classified as **direct methods**. In most other areas of mathematics the exact solution is never possible in a finite number of steps, e.g. nonlinear equations, most differential equations, integral equations etc., and we must instead be satisfied with an approximation which is often obtained by some kind of iteration. The exact solution to the linear equations is however rarely obtained in practice since the use of floating point arithmetic nearly always gives an approximate solution. It depends on the application as to how accurately you need to actually solve the linear equations. For example, if the linear equations arise from some kind of modelling then it is only necessary to solve the equations to about the accuracy of the model and generally this is much less than the machine precision on a computer. That is, in applications it is rarely necessary to obtain the solution to the linear equations correct to 15 significant figures. Unfortunately, with most direct methods you either obtain no solution (the method breaks down) or a solution which is accurate to approximately machine precision provided the condition number $\kappa(A) = \|A\|\,\|A^{-1}\|$ is not too large. That is the direct methods are usually unnecessarily accurate in the context of the intended application. Another drawback of direct methods is that they often require excessive storage in the case of many large sparse matrices which arise, for example, in the approximate solution of partial differential equations. These are reasons for considering alternatives to direct methods in certain cases and in particular to the consideration of **iterative methods**. As the applications which give rise to the large sparse systems are not covered in modules before level 3, iterative methods are not covered in any details in this module because to analyse the performance of the methods depends on the details of the problem being considered. Specific iterative methods can be, and have been, the topic of final year projects. We restrict attention here to briefly reviewing some standard methods and to giving just enough details to describe a procedure known as iterative refinement which is used in packages, including Matlab, to improve the accuracy of the solution obtained using Gauss elimination/LU factorization procedures implemented on a computer in floating point arithmetic.

7.2 Iterative methods of the splitting type

Many of the standard iterative methods involve a splitting of A, that is they involve representing A in the form

$$A = M - N$$

where M is non-singular and easily invertible, i.e. it is 'easy' to solve linear systems with the matrix M. There are infinitely many ways M may be chosen and each different choice leads to a different iterative method. The systems can then be written as

$$A\underline{x} = (M-N)\underline{x} = \underline{b}, \quad \text{or as} \quad M\underline{x} = N\underline{x} + \underline{b}, \quad \text{or as} \quad \underline{x} = G\underline{x} + \underline{c},$$

where

$$G = M^{-1}N \quad \text{and} \quad \underline{c} = M^{-1}\underline{b}.$$

These are different ways of characterising the exact solution $\underline{x}$. Then given any starting vector $\underline{x}_0$ we can construct the sequence

$$M\underline{x}_{k+1} = N\underline{x}_k + \underline{b}, \quad \text{i.e. } \underline{x}_{k+1} = G\underline{x}_k + \underline{c}, \quad k = 0, 1, 2, \cdots .$$

The form $M\underline{x}_{k+1} = N\underline{x}_k + \underline{b}$ corresponds to how the method is implemented (you do not need to actually generate G). The form $\underline{x}_{k+1} = G\underline{x}_k + \underline{c}$ is used in the analysis of the method. The matrix $G = M^{-1}N$ is called the iteration matrix and whether or not the sequence $(\underline{x}_0, \underline{x}_1, \underline{x}_2, \cdots)$ converges to the vector $\underline{x} = A^{-1}\underline{b}$ depends on the properties of G which we now consider.

The error in each iterate $\underline{x}_k$ is defined by

$$\underline{e}_k = \underline{x} - \underline{x}_k$$

and since $\underline{x} = G\underline{x} + \underline{c}$ and $\underline{x}_{k+1} = G\underline{x}_k + \underline{c}$ we have

$$\underline{e}_{k+1} = \underline{x} - \underline{x}_{k+1} = (G\underline{x} + \underline{c}) - (G\underline{x}_k + \underline{c}) = G(\underline{x} - \underline{x}_k) = G\underline{e}_k \; .$$

Thus the error at one step is related to the error at the previous step and from this we can express the error at the kth step in terms of the initial error. We have

$$\underline{e}_k = G\underline{e}_{k-1} = G^2\underline{e}_{k-2} = \cdots = G^k\underline{e}_0 \; .$$

Now for any of the subordinate matrix norms we have

$$\|G^k\| \le \|G\|^k \quad \text{and} \quad \|\underline{e}_k\| \le \|G\|^k \|\underline{e}_0\| \to 0 \quad \text{as } k \to \infty$$

if $\|G\| < 1$. Hence a sufficient condition for convergence of the iteration is that $\|G\| < 1$ and convergence is rapid if $\|G\|$ is small. Observe that when $\|G\| < 1$ we have convergence from all starting vectors $\underline{x}_0$. In fact the sufficient condition above can be improved and replaced by a necessary and sufficient for convergence from all starting vectors which we state in the following theorem.

Theorem. (Necessary and sufficient condition for convergence)
An iterative method $\underline{x}_{k+1} = G\underline{x}_k + \underline{c}$ converges from all initial vectors $\underline{x}_0$ if and only if

$$\rho(G) < 1$$

where $\rho(G)$ is the spectral radius of the matrix G.

A full proof of this is just outside the syllabus for this module and will not be considered here although it is worth considering some of the details so as to attempt to understand why the result is true and to understand how rapidly the iteration will converge. In the case of an iteration matrix G which has n linearly independent eigenvectors $\underline{v}_1, \cdots, \underline{v}_n$ with corresponding eigenvalues $\lambda_1, \cdots, \lambda_n$ we can express the (unknown) initial error $\underline{e}_0$ in terms of these eigenvectors to give

$$\underline{e}_0 = \sum_{i=1}^{n} \alpha_i \underline{v}_i \quad \text{giving} \quad \underline{e}_k = G^k \underline{e}_0 = \sum_{i=1}^{n} \alpha_i \lambda_i^k \underline{v}_i \; .$$

From this representation of the error the rate at which $\underline{e}_k$ tends to $\underline{0}$ depends on the rate at which the 'slowest' of $\lambda_1^k \to 0, \cdots, \lambda_n^k \to 0$, i.e. it depends on the eigenvalue nearest to 1 in modulus which is the eigenvalue with the largest modulus. That is the rate of convergence depends on the spectral radius $\rho(G)$. If $\rho(G)$ is close to 1 then the convergence will be slow and if it is near to 0 then it will be fast.

A slightly different way of describing the sequence $\underline{x}_0, \underline{x}_1, \underline{x}_2, \cdots$ and considering convergence is to consider a representation of the the exact solution $\underline{x}$ obtained by using a result given in the section on the sensitivity of linear equations. As we have

$$\underline{x} = G\underline{x} + \underline{c}, \quad \text{i.e. } (I - G)\underline{x} = \underline{c} \quad \text{and thus} \quad \underline{x} = (I - G)^{-1}\underline{c}$$

and when $\|G\| < 1$ we have

$$\underline{x} = (I + G + G^2 + \cdots)\underline{c} = \left(\sum_{i=0}^{\infty} G^i\right)\underline{c}\ .$$

This is in fact valid for $\rho(G) < 1$. When $\underline{x}_0 = \underline{0}$ we get

$$\underline{x}_1 = \underline{c}, \quad \underline{x}_2 = \underline{c} + G\underline{c}, \quad \underline{x}_3 = \underline{c} + G\underline{c} + G^2\underline{c}, \cdots$$

that is

$$\underline{x} = \left(\sum_{i=0}^{\infty} G^i\right)\underline{c} \quad \text{and} \quad \underline{x}_k = \left(\sum_{i=0}^{k-1} G^i\right)\underline{c}\ .$$

7.3 The Jacobi method and Gauss-Seidel methods

To describe these methods we split A in the form

$$A = D - L - U$$

where D is a diagonal matrix, L is strictly lower triangular and U is strictly upper triangular. Thus each entry of A is in exactly one of D, $-L$ and $-U$. The Jacobi iterative method involves taking

$$M = D \quad \text{and} \quad N = L + U \quad \text{giving} \quad G = M^{-1}N = D^{-1}(L + U)\ .$$

The Gauss-Seidel iterative method involves taking

$$M = D - L \quad \text{and} \quad N = U \quad \text{giving} \quad G = M^{-1}N = (D - L)^{-1}U\ .$$

Whether or nor these methods work for a particular matrix A depends on whether or not $\rho(G) < 1$ in each case with, as a rough guide, convergence occurring when the diagonal matrix D is 'large' compared to the off-diagonal matrices L and U. In fact it can be proved that the Jacobi method converges when A is **strictly diagonally dominant**, i.e. when

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{for } i = 1, 2, \cdots, n\ .$$

Also, it can be shown that when the matrix A is **symmetric and positive definite** the Gauss-Seidel method converges. In many practical situations it can be shown that both methods converge with the Gauss-Seidel method converging about 'twice as fast' as the Jacobi method.

7.4 Iterative refinement

An iteration of the splitting type is used in packages, including Matlab, to improve the accuracy of the solution obtained by Gauss elimination type methods. The splitting of A comes about as although we aim to construct a decomposition

$$PA = LU,$$

where P=permutation matrix, L=unit lower triangular and U=upper triangular, the use of floating point arithmetic means that we actually get matrices $\hat{P}$, $\hat{L}$, and $\hat{U}$ such that

$$\hat{P}A = \hat{L}\hat{U} - \Delta A$$

where $\|\Delta A\|$ is hopefully small compared with $\|A\|$. To help simplify the notation slightly, in our analysis we assume that $\hat{P} = P = I$ so that we have

$$\hat{L}\hat{U} = A + \Delta A, \quad \text{i.e. } A = \hat{L}\hat{U} - \Delta A \ .$$

In the latter form we have a splitting $A = M - N$ with $M = \hat{L}\hat{U}$ and $N = \Delta A$. This suggests the iteration

$$M\underline{x}_{k+1} = N\underline{x}_k + \underline{b}, \quad \text{i.e. } \hat{L}\hat{U}\underline{x}_{k+1} = \Delta A\underline{x}_k + \underline{b} \ .$$

We can rewrite this as

$$\hat{L}\hat{U}\underline{x}_{k+1} = (\hat{L}\hat{U} - A)\underline{x}_k + \underline{b} \quad \text{or as} \quad \hat{L}\hat{U}(\underline{x}_{k+1} - \underline{x}_k) = \underline{b} - A\underline{x}_k \ .$$

This is the way it is usually used. We compute the residual $\underline{r}_k = \underline{b} - A\underline{x}_k$ and then solve

$$\hat{L}\hat{U}\underline{d}_k = \underline{r}_k \quad \text{and set} \quad \underline{x}_{k+1} = \underline{x}_k + \underline{d}_k \ .$$

The solution process is done by forward and back substitution and it hence only involves $\mathcal{O}(n^2)$ flops, for a full matrix, as compared to the $\mathcal{O}(n^3)$ operations to create $\hat{L}$ and $\hat{U}$.

To analyse the progress of the iteration we need to consider the iteration matrix G which is given by

$$G = M^{-1}N = (\hat{L}\hat{U})^{-1}\Delta A = (A + \Delta A)^{-1}\Delta A = (I + A^{-1}\Delta A)^{-1}(A^{-1}\Delta A)$$

and thus provided $\|A^{-1}\Delta A\| < 1$

$$\|G\| \le \frac{\|E\|}{1 - \|E\|}, \quad \text{where} \quad E = A^{-1}\Delta A \ .$$

If $\|E\| < 1/2$ then $\|G\| < 1$ and the iteration converges. In practice $\|\Delta A\|$ is 'very small' so that $\|E\|$ is 'quite small' and thus $\|G\| \approx \|E\|$ and convergence is rapid. Only 1 or 2 iterations is usually used to get sufficient accuracy.

The point about iterative refinement is that despite obtaining an error ΔA when computing the LU factorization you can still usually obtain a vector $\hat{\underline{x}}$ for which the residual

$$\underline{r}(\hat{\underline{x}}) = \underline{b} - A\hat{\underline{x}}$$

satisfies

$$\|\underline{r}(\hat{\underline{x}})\| \approx (\text{machine precision})\|\underline{b}\| \ .$$

This is about the best you can hope for when trying to solve $A\underline{x} = \underline{b}$ on a computer.

We have now reached the point where we have just about answered the question as to what Matlab does when it implements `x=A\b;` when `A` is a full $n \times n$ matrix and `b` is a $n \times 1$ vector. If mathematically A corresponds to `A` and $\underline{b}$ corresponds to `b` then Matlab computes a factorization $PA = LU$ using row scaled partial pivoting (see section 2), it obtains $\underline{x}_0$ using forward and back substitution and it performs one or two steps of iterative refinement (as described in this section) to obtain an approximate solution $\hat{\underline{x}}$ which 'satisfies the equations to about machine precision'.

7.5 Other types of iterative methods

The Jacobi method, the Gauss-Seidel method and the iterative refinement procedure all concern methods of the splitting type. Another method of this type, which is just beyond the scope of this module, is Successive Over Relaxation (SOR) which in a sense generalises the Gauss-Seidel method. An alternative class of iterative methods involves converting the linear equations problem into an equivalent minimization problem and involves attempting to minimise a quadratic by an iterative procedure. Now, the idea of minimization has already been met in the case of least squares. Recall that we had

$$\begin{aligned} \|A\underline{x}-\underline{b}\|_2^2 &= \underline{x}^T A^T A\underline{x} - 2\underline{x}^T A^T \underline{b} + \underline{b}^T\underline{b} \\ &= \underline{x}^T C\underline{x} - 2\underline{d}^T\underline{b} + \underline{b}^T\underline{b} \quad \text{where } C = A^TA,\ \underline{d} = A^T\underline{b}\ . \end{aligned}$$

The matrix $C = A^TA$ is symmetric and positive definite (when A has full rank) and the minimum point satisfies the normal equations

$$A^TA\underline{x} = A^T\underline{b} \quad \text{i.e. } C\underline{x} = \underline{d}\ .$$

The least squares problem was done in the context of $A \in \mathbb{R}^{m,n}$ with $m \ge n$. In the case that $A \in \mathbb{R}^{n,n}$ is symmetric and positive definite we similarly have that the unique minimum of the quadratic

$$\underline{x}^TA\underline{x} - 2\underline{x}^T\underline{b}$$

is the unique solution of $A\underline{x} = \underline{b}$. The conjugate gradient method and the related preconditioned conjugate gradient method both involve minimising this quadratic and they have been implemented in Matlab since version 5.0 by the command `pcg`. Type `help pcg` to find out what to do. Since version 5.0 Matlab also contains iterative methods for non-symmetric matrices.

It is interesting to note that in the classification of methods the conjugate gradient method is both a direct method and an iterative method. It qualifies to be called a direct method as it gives the exact answer with exact arithmetic in at most n iterations where n is the size of the system but it is usually used as an iterative method with the iteration terminated before this stage is reached if a sufficiently accurate approximation is obtained with fewer than n iterations.

7.6 Some key results of the section

1. Given $A \in \mathbb{R}^{n\times n}$ there are many ways of choosing M and N with M being non-singular and easily invertible such that $A = M - N$ and we can write

$$A\underline{x} = \underline{b}, \quad M\underline{x} = N\underline{x} + \underline{b}, \quad \underline{x} = G\underline{x} + \underline{c}, \quad G = M^{-1}N,\ \underline{c} = M^{-1}\underline{b}\ .$$

2. The iteration

$$M\underline{x}_{k+1} = N\underline{x}_k + \underline{b}, \quad \text{i.e. } \underline{x}_{k+1} = G\underline{x}_k + \underline{c}, \quad k = 0, 1, 2, \cdots\ .$$

   converges for all $\underline{x}_0$ if and only if $\rho(G) < 1$.

3. Rounding error in the computer implementation of the 'LU factorization' procedure leads to computed factors $\hat{L}$ and $\hat{U}$ with

$$A = \hat{L}\hat{U} - \Delta A, \quad \text{i.e. } A = M - N \text{ with } M = \hat{L}\hat{U},\ N = \Delta A = \hat{L}\hat{U} - A.$$

   Iterative refinment uses this splitting to improve accuracy.

8 The eigenvalue problem $A\underline{v} = \lambda\underline{v}$ – Theory and estimation

8.1 Background

The standard eigenvalue problem $A\underline{v} = \lambda\underline{v}$ is one of the major problems of numerical linear algebra and there is much theory and methods associated with it. For this problem the methods are necessarily iterative since the eigenvalues are the roots of the characteristic polynomial of A and the determination of the roots of a polynomial is a nonlinear problem which cannot be solved in general in a finite number of steps. This should be contrasted with the linear equations problem in which the exact solution is obtained when exact arithmetic is used in a finite number of steps when direct methods are used. However, as has already been mentioned in the context of iterative methods for linear equations, this fundamental distinction between 'direct' and 'iterative' methods is not usually an important practical issue. In practice you are usually only concerned with reliably getting a good numerical approximation quickly.

There are many numerical methods connected with the standard eigenvalue problem and the method you should use depends on why you are trying to solve the problem. For example you may only want the eigenvalues and not the eigenvectors. You may only want the largest or dominant eigenvalue in order to have the spectral radius $\rho(A)$. You may only want the eigenvalues in a given interval or you may want all the eigenvalues, i.e. the complete spectrum, and all the associated eigenvectors. Once you have considered which of these categories your problem comes into you should consider whether there are any special properties of the matrix A which you may be able to exploit in order to most efficiently solve the problem. In this context the special property of symmetry is important as it turns out that the eigenvalue problem is 'well-conditioned' in this case which is as a consequence of the property that the eigenvectors of such a matrix can be taken to be orthogonal.

8.2 Previous parts of the module which have involved eigenvalues

Before discussing some of the theory relating to eigenvalues and eigenvectors it is perhaps worthwhile reviewing previous parts of this module in which eigenvalues have been mentioned.

The largest eigenvalue in modulus (i.e. the spectral radius) has been mentioned several times. In the section on norms we had the results

$$\|A\|_2 = \left(\rho(A^TA)\right)^{1/2} \quad \text{and} \quad \rho(A) \le \|A\| \quad \text{for all subordinate norms.}$$

In the previous section on iterative methods for linear equations we had the result that the iteration $\underline{x}_{k+1} = G\underline{x}_k + \underline{c}$ converges from all starting vectors if and only if $\rho(G) < 1$.

We have mentioned eigenvalues and eigenvectors in the case of real symmetric matrices a number of times. The main result used in this context is that the eigenvalues are real and that the eigenvectors can be taken to be orthonormal and thus for a $n \times n$ symmetric matrix A they give an orthonormal basis for $\mathbb{R}^n$. This was used to prove that $\|A\|_2^2 = \rho(A^TA)$. We have also noted that when $\underline{v} \ne \underline{0}$ is an eigenvector the eigenvalue is given by

$$\lambda = \frac{\underline{v}^TA\underline{v}}{\underline{v}^T\underline{v}} \ .$$

In the case of real symmetric positive definite matrices this enables us to immediately deduce that $\lambda > 0$, i.e. all the eigenvalues are positive. This was used in the proof of the Cholesky factorization

$A = LL^T$ when we used the property

$$\det A = \text{product of the eigenvalues} > 0 \ .$$

Also, in the context of real symmetric positive definite matrices we showed that

$$\begin{aligned} \|A\|_2 &= \text{Largest eigenvalue of } A, \\ \|A^{-1}\|_2 &= \frac{1}{\text{Smallest eigenvalue of } A}, \\ \kappa_2(A) &= \frac{\text{Largest eigenvalue of } A}{\text{Smallest eigenvalue of } A} \ . \end{aligned}$$

Finally, in an exercise on norms we had a question whose answer used the eigenvalues of A^TA. If $\mu_1, \mu_2, \cdots, \mu_n$ are the eigenvalues of A^TA then

$$\begin{aligned} \|A\|_E^2 &= \operatorname{trace}(A^TA) = \text{sum of eigenvalues of } A^TA \\ &= \mu_1 + \mu_2 + \cdots + \mu_n, \\ \|A\|_2^2 &= \max_i \mu_i \ . \end{aligned}$$

From this it follows that

$$\|A\|_2 \le \|A\|_E \le \sqrt{n}\|A\|_2 \ .$$

To summarize, eigenvalues and eigenvectors have been used in explanations (e.g. in the Cholesky factorization, in the sensitivity of linear equations and in the convergence of $\underline{x}_{k+1} = G\underline{x}_k + \underline{c}$) with the important eigenvalues generally being just the smallest and the largest which are reasons for considering methods which only determine these quantities. The Power Method and the Inverse Power Method, see later, are methods which can be used for obtaining these.

8.3 A revision of eigenvalues and eigenvectors

Before methods for estimating eigenvalues are considered (the Gerschgorin theorem and the Rayleigh quotient in these notes) and before iterative methods for attempting to accurately compute eigenvalues are considered we quickly cover some definitions and results relating to eigenvalues and eigenvectors which you should have met before.

8.3.1 The characteristic polynomial and the characteristic equation

Eigenvalues and eigenvectors are defined as follows. For a square matrix A of size $n \times n$, an eigenvalue λ (possibly complex) and an associated eigenvector $\underline{v} \ne \underline{0}$ (also possibly complex) are such that

$$A\underline{v} = \lambda\underline{v} \ .$$

To obtain a way of characterizing these quantities the equation is re-written as

$$(\lambda I - A)\underline{v} = \underline{0}$$

which implies that $\lambda I - A$ is a singular matrix which in turn implies that

$$\det(\lambda I - A) = 0 \ .$$

This is the **characteristic equation** for λ. The **characteristic polynomial** of degree n is defined by

$$P_A(t) = \det(tI - A) \ .$$

The roots of the characteristic polynomial are thus the eigenvalues of A. For the existence and number of eigenvalues we refer to the **Fundamental Theorem of Algebra** which states that a complex polynomial of degree n has exactly n roots (counting multiplicities). That is we have $\lambda_1, \lambda_2, \cdots, \lambda_n$ (in the complex plane) such that the polynomial can be factored as

$$P_A(t) = (t - \lambda_1)(t - \lambda_1) \cdots (t - \lambda_n) \ .$$

In most applications we are usually only interested in the eigenvalues of real matrices which thus corresponds to the roots of a polynomial with real coefficients. That is $P_A(t)$ is of the form

$$P_A(t) = t^n + c_{n-1}t^{n-1} + \cdots + c_1 t + c_0, \quad c_0, c_1, \cdots, c_{n-1} \in \mathbb{R} \ .$$

If we consider the polynomial as a function of the complex variable $z = r\mathrm{e}^{i\theta}$ then we have

$$z = r\mathrm{e}^{i\theta}, \quad \overline{z} = r\mathrm{e}^{-i\theta}, \quad z^k = r\mathrm{e}^{ik\theta}, \quad \text{and} \quad (\overline{z})^k = \overline{(z^k)} = r^k\mathrm{e}^{-ik\theta}$$

and from this it follows that

$$P_A(\overline{z}) = \overline{P_A(z)} \ .$$

Hence if $P_A(z) = 0$ then $P_A(\overline{z}) = 0$ and thus complex roots must occur in complex conjugate pairs.

8.3.2 Similar matrices

Once eigenvalues and eigenvectors have been defined with the eigenvalues characterized as the roots of $P_A(t)$ it is then usual to consider matrices related to A of the form

$$B = S^{-1}AS,$$

where S is an invertible matrix. If matrices A and B are related in this way then they are said to be **similar**. An important property of similar matrices is that they have the same characteristic polynomials and hence the same eigenvalues. This follows using properties of determinants. We have

$$tI - B = tI - S^{-1}AS = S^{-1}(tI - A)S,$$

and thus

$$P_B(t) = \det(tI - B) = \det S^{-1} \, \det(tI - A) \, \det S = P_A(t),$$

since $I = SS^{-1}$ implies that $1 = \det S \, \det S^{-1}$.

The general purpose methods for computing all the eigenvalues of a matrix involve creating a sequence $A_0 = A, \ A_1, A_2, \cdots$ of similar matrices which converge to a triangular or a diagonal matrix. When we have a triangular or a diagonal matrix the eigenvalues are just the diagonal entries. Techniques of this type will be briefly reviewed in the final section of this module.

8.4 Estimation of the location of the eigenvalues: using norms

We begin by repeating an earlier result linking subordinate matrix norms and the spectral radius.

The result is that for any subordinate matrix norm

$$\rho(A) \le \|A\| \ .$$

This follows immediately from the definition

$$\|A\| = \max_{\|\underline{x}\|=1} \|A\underline{x}\| \quad \text{and the relation} \quad A\underline{v} = \lambda\underline{v}, \ \underline{v} \ne \underline{0}.$$

To be useful we must use a matrix norm which can be easily computed. Using the 1-norm and ∞-norm we have

$$\rho(A) \le \min\{\|A\|_1, \|A\|_\infty\},$$

where

$$\begin{aligned} \|A\|_\infty &= \text{maximum row sum of absolute values,} \\ \|A\|_1 &= \text{maximum column sum of absolute values.} \end{aligned}$$

8.5 Estimation of the location of the eigenvalues: using Gerschgorin's theorem

We now show that the norm result can often be significantly improved by using some results of Gerschgorin which are concerned with comparing the given matrix A with the diagonal matrix $\text{diag}\{a_{11}, \cdots, a_{nn}\}$. As we will see this has the effect of replacing one disk in the complex plane in which we know that all the eigenvalues lie by n (usually smaller) disks which are contained within it. Part of the result of the theorem that follows relies on the following theorem of analysis which we will not prove.

Theorem. The eigenvalues of a matrix A are continuous functions of the entries of A.

There is a corresponding results for polynomials which is that the roots of a polynomial are a continuous function of the coefficients of the polynomial.

Theorem. (Gerschgorin's Theorem) Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ and let

$$R_i'(A) = \sum_{\substack{j=1 \\ j\ne i}}^{n} |a_{ij}|, \quad i = 1, 2, \cdots, n$$

denote the deleted absolute row sums of A. Then we have the following.

(i) All the eigenvalues of A are located in the union of the n disks

$$G(A) = \bigcup_1^n \{z \in \mathbb{C} : \quad |z - a_{ii}| \le R_i'(A)\} \ .$$

(ii) If a union of k of these disks form a connected region that is disjoint from the remaining $n-k$ disks then there are precisely k eigenvalues of A in this region.

PROOF.

(i) Let $\underline{v} \ne \underline{0}$ be an eigenvector of A with eigenvalue λ and suppose that the index p is such that $|v_p| = \|\underline{v}\|_\infty$. By considering the pth component of the vector identity $\lambda\underline{v} - A\underline{v} = \underline{0}$ we have

$$\lambda v_p - \sum_{j=1}^{n} a_{pj} v_j = 0, \quad \text{i.e. } (\lambda - a_{pp}) v_p = \sum_{j\ne p} a_{pj} v_j \ .$$

Thus by using the triangle inequality we have

$$|\lambda - a_{pp}|\,|v_p| \le \sum_{j\ne p} |a_{pj}|\,|v_j| \le \|\underline{v}\|_\infty R_p'$$

and since we have chosen the index p such that $|v_p| = \|\underline{v}\|_\infty$ we have that

$$|\lambda - a_{pp}| \le R'_p.$$

Hence λ is in the pth disk and consequently it is in $G(A)$ which is in the union of all the disks.

(ii) Let $D = \mathrm{diag}\{a_{11}, \cdots, a_{nn}\}$ and let

$$M(t) = D + t(A - D), \quad 0 \le t \le 1 \ .$$

Thus $M(t)$ is a matrix which varies continuously with t from the diagonal matrix D (when $t = 0$) to the matrix A (when $t = 1$). If $\lambda_1(t), \cdots, \lambda_n(t)$ denote the eigenvalues of $M(t)$ with $\lambda_i(0) = a_{ii}$ then $\lambda_1(t), \cdots, \lambda_n(t)$ define continuous paths in the complex plane starting respectively from the points $a_{11}, \cdots, a_{nn}$ since the eigenvalues of a matrix vary continuously with the entries of a matrix. Since the Gerschgorin disks of $M(t)$ increase in size as t increases we have

$$\lambda_i(t) \in G(M(t)) \subset G(A) \ .$$

If k of the disks are disjoint from the other $n - k$ disks, and for ease of labelling we assume that these are the first k, then we must have that

$$\lambda_i(t) \in \bigcup_1^k \{z \in \mathbb{C} : \quad |z - a_{jj}| \le R'_j(A)\}, \quad i = 1, \cdots, k, \quad 0 \le t \le 1.$$

This is because the paths are continuous, they start in this region ($\lambda_i(0) = a_{ii}$) and at all stages they are contained in $G(A)$. They cannot thus jump to the other region

$$G(A) / \bigcup_1^k \{z \in \mathbb{C} : \quad |z - a_{jj}| \le R'_j(A)\}$$

as this is only possible if the paths can be discontinuous. $\square$

Notation. The circles $\{z \in \mathbb{C} : \quad |z - a_{ii}| = R'_i\}$ are called the **Gerschgorin circles**. The interior of these circles are called the **Gerschgorin disks**.

Examples

Let A_1 and A_2 be the 3×3 matrices given below.

$$A_1 = \begin{pmatrix} 2 & 0.1 & 0.1 \\ 0.1 & 3 & 0.1 \\ 0.1 & 0.1 & 4 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 2 & 1 & 0.1 \\ 0.1 & 3 & 0.1 \\ 0.1 & 0.1 & 4 \end{pmatrix}.$$

In the case of the matrix A_1 all 3 disks are isolated and we can immediately deduce that there are eigenvalues $\lambda_1, \lambda_2, \lambda_3$ satisfying $|\lambda_1 - 4| \le 0.2$, $|\lambda_2 - 3| \le 0.2$ and $|\lambda_3 - 2| \le 0.2$. In the case of the matrix A_2 only the third disk is isolated and all we can immediately deduce is that the largest eigenvalue λ_1 satisfies $|\lambda_1 - 4| \le 0.2$. The other two eigenvalues are in the region which is the union of the other two disks.

8.5.1 Some refinements of the Gerschgorin theorem and the case of isolated disks

As well as applying Gerschgorin's theorem to A we can apply the theorem to any easily constructed matrix which has the same eigenvalues as A. For example we can apply the theorem to

$$A^T, \quad D^{-1}AD \quad \text{and} \quad D^{-1}A^TD,$$

where $D = \mathrm{diag}\{d_i\}$, $d_i > 0$, is a diagonal matrix. In the case of the diagonal similarity transformations the ij entry of AD is $a_{ij}d_j$ and the ij entry of $D^{-1}AD$ is

$$a_{ij}(d_j/d_i)$$

and thus the Gerschgorin disks of $D^{-1}AD$ are

$$\left\{ z \in \mathbb{C} : \quad |z - a_{ii}| \le \sum_{\substack{j=1 \\ j\ne i}}^{n} |a_{ij}|\frac{d_j}{d_i} \right\} .$$

The disks have the same centres as the disks of A but they have different radii.

We now consider some examples showing the application of Gerschgorin's theorem to specific matrices and additionally consider appropriate matrices of the form $D^{-1}AD$ to improve the estimation in the case of isolated disks. In the case of the two matrices

$$\begin{pmatrix} 1 & -2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 8 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 8 \end{pmatrix}$$

we get the same disks and these are shown in Figure 1. The first 2 disks overlap and hence all we can deduce is that the union of these 2 disks contains 2 eigenvalues. In fact, it can be shown (e.g. using MATLAB) that in the case of the first matrix all the eigenvalues are $1.4254685 \pm 1.932763i$ and 8.1490629 with the 1st 2 disks thus containing a complex conjugate pair of eigenvalues. The second matrix which also has these disks is symmetric and has the real eigenvalues -0.6063346, 3.4284193 and 8.1779154. The use of Gerschgorin's theorem hence does not locate very precisely the eigenvalues in the first 2 disks which have radius 2 and 3 respectively but we get a better indication about the eigenvalue in the 3rd disk. You should note that the radius of the 3rd disk is 1 with the eigenvalue being only about 0.149 and 0.178 from the centre in the 2 cases. This can be explained by considering the similar matrix

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d^{-1} \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 8 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} = \begin{pmatrix} 1 & 2 & 0 \\ 2 & 2 & d \\ 0 & 1/d & 8 \end{pmatrix} .$$

By taking $d > 1$ we expand the 2nd disk and decrease the size of the 3rd disk with the 2 disks touching when

$$4 + d = 8 - 1/d, \quad \text{i.e. } d = 4 - 1/d, \quad \text{giving } d^2 - 4d + 1 = 0 .$$

Thus

$$d = 2 + \sqrt{3} = 3.7321 .$$

Hence we can make the 3rd disk as small as 0.26795 and guarantee that we have an eigenvalue in $|x - 8| < 0.26795$.

In the case of the matrix

$$A = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix}$$

the off-diagonal entries are much smaller and we get the 3 isolated disks shown in Figure 2. Again, by considering matrices of the form $D^{-1}AD$, we can significantly reduce the size of the disks. Consider first the 3rd disk. We have

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01d \\ 0 & 0.01/d & 8 \end{pmatrix} .$$

As we will show in the next section when the Rayleigh quotient is described, the largest eigenvalue λ_l which lies in the 3rd disk must satisfy $\lambda_l > 8$. The Gerschgorin result implies that by considering the 2nd and 3rd disks we have

$$\lambda_l \le \max\{2 + 0.1 + 0.01d, 8 + 0.01/d\} .$$

By inspection, when $d = 590$ we get $\lambda_l < 8.0000169$. The largest d that you can take corresponds to when

$$2.1 + 0.01d = 8 + 0.01/d, \quad \text{giving } d^2 - 590d - 1 = 0 \quad \text{and } d = 295 + \sqrt{295^2 + 1} = 590.00169$$

and hence the estimate is hardly improved. The actual eigenvalue is 8.00001667 which compares well with the estimate that it is in the interval $(8, 8.0000169)$.

The same idea as the above can be used to improve the estimate of the smallest eigenvalue which is close to 1. We have

$$\begin{pmatrix} d^{-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix} \begin{pmatrix} d & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0.1/d & 0 \\ 0.1d & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix} .$$

Again it can be shown that the smallest eigenvalue $\lambda_s < 1$ by considering Rayleigh quotients (see later). The Gerschgorin result implies that

$$\lambda_s \ge \max\{1 - 0.1/d, 1.99 - 0.1d\} .$$

By taking $d = 10$ we get $\lambda_s \ge 0.99$ which compares well with the actual eigenvalue which is 0.9900979.

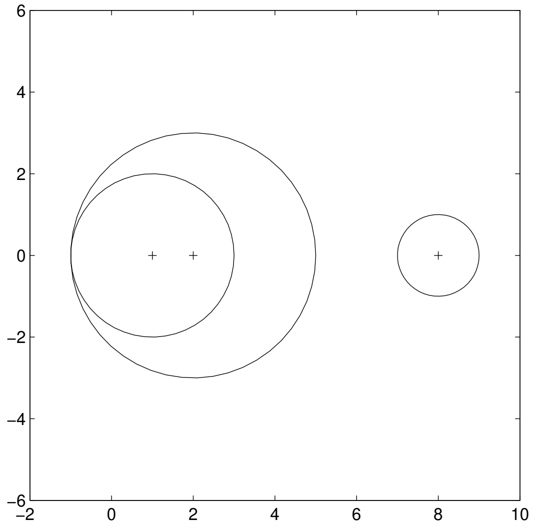


Figure 1: The 3rd disk is isolated and hence contains exactly 1 eigenvalue.

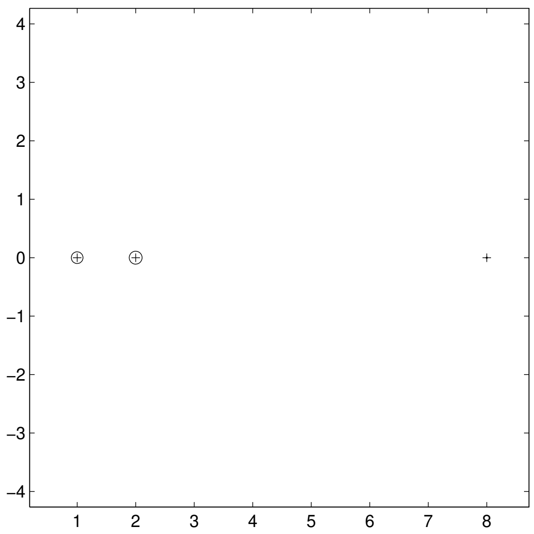


Figure 2: All 3 disks are isolated and hence contains each contains exactly 1 eigenvalue.

8.6 Estimation in the real symmetric case: using Rayleigh quotients

For a real square matrix A and a non-zero vector $\underline{x}$ this is the quantity

$$\frac{\underline{x}^T A \underline{x}}{\underline{x}^T \underline{x}} .$$

The first property to observe concerning this quantity is that if $\underline{x} \neq \underline{0}$ is an eigenvector with eigenvalue λ then

$$A\underline{x} = \lambda \underline{x} \quad \text{and} \quad \lambda = \frac{\underline{x}^T A \underline{x}}{\underline{x}^T \underline{x}} .$$

That is the Rayleigh quotient is the eigenvalue when the vector is the eigenvector. This suggests that when we have an approximate eigenvector $\underline{\hat{x}}$ the Rayleigh quotient of A with $\underline{\hat{x}}$ is a good candidate for the associated approximate eigenvalue. In the case of real symmetric matrices we will now show that in a well defined sense it is best value to take.

Consider the following quadratic in t.

$$\begin{aligned} g(t) = \|A\underline{\hat{x}} - t\underline{\hat{x}}\|_2^2 &= (A\underline{\hat{x}} - t\underline{\hat{x}})^T (A\underline{\hat{x}} - t\underline{\hat{x}}) \\ &= \underline{\hat{x}}^T A^T A \underline{\hat{x}} - 2t\underline{\hat{x}}^T A \underline{\hat{x}} + t^2 \underline{\hat{x}}^T \underline{\hat{x}} . \end{aligned}$$

The quadratic has a minimum when

$$t = \frac{\underline{\hat{x}}^T A \underline{\hat{x}}}{\underline{\hat{x}}^T \underline{\hat{x}}}, \quad \text{the Rayleigh quotient.}$$

We could have argued this in another way as follows. In the jargon of least squares, the Rayleigh quotient is the 'least squares solution of $A\underline{\hat{x}} = t\underline{\hat{x}}$'. Notice that the matrix in this case is the vector $\underline{\hat{x}}$ and the right hand side vector is $A\underline{\hat{x}}$ and the normal equations are thus

$$(\underline{\hat{x}}^T \underline{\hat{x}})t = \underline{\hat{x}}^T (A\underline{\hat{x}}).$$

In the case of real symmetric matrices we next show that the Rayleigh quotient gives a lower bound for the largest eigenvalue and an upper bound for the smallest eigenvalue. If the eigenvalues of a real symmetric matrix A are ordered so that

$$\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$$

with $\underline{u}_1, \cdots, \underline{u}_n$ denoting the corresponding orthonormal eigenvectors then, for a general vector $\underline{x}$

$$\underline{x} = \sum_1^n c_i \underline{u}_i, \quad \text{with } c_i = \underline{x}^T \underline{u}_i, \quad A\underline{x} = \sum_1^n c_i \lambda_i \underline{u}_i$$

and

$$\frac{\underline{x}^T A \underline{x}}{\underline{x}^T \underline{x}} = \frac{\sum_1^n c_i^2 \lambda_i}{\sum_1^n c_i^2} .$$

Since

$$\lambda_n \sum_1^n c_i^2 \leq \sum_1^n c_i^2 \lambda_i \leq \lambda_1 \sum_1^n c_i^2$$

we have

$$\lambda_n \le \frac{\underline{x}^T A \underline{x}}{\underline{x}^T \underline{x}} \le \lambda_1 \ .$$

The Rayleigh quotient thus always lies between the smallest and largest eigenvalues and it can thus be used to bound these quantities. We illustrate this now in the case of the largest and smallest eigenvalues of the matrix

$$A = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix}$$

considered in the Gerschgorin examples.

By taking the Rayleigh quotient of A with the base vector $\underline{e}_i$ we get

$$\underline{e}_i^T A \underline{e}_i = a_{ii} = i\text{th diagonal entry} \ .$$

Hence we immediately get that for the largest and smallest eigenvalues λ_l and λ_s respectively that

$$1 < \lambda_s \quad \text{and} \quad \lambda_l > 8 \ .$$

(We have strict inequalities here since the base vectors are not eigenvectors of A.) To improve the estimates we note that with $\underline{v} = (1, -0.1, 0)^T$ we have

$$A\underline{v} = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix} \begin{pmatrix} 1 \\ -0.1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.99 \\ -0.1 \\ -0.001 \end{pmatrix} \ .$$

Hence

$$\frac{\underline{v}^T A \underline{v}}{\underline{v}^T \underline{v}} = \frac{0.99 + 0.01}{1 + 0.01} = \frac{1}{1.01} = 0.990099..$$

Hence the previous Gerschgorin result and this result give the interval $(0.99, 0.990099)$. Recall that the actual eigenvalue is 0.9900979.

Similarly, by taking $\underline{v} = (0, 0.001, 1)^T$ we get

$$A\underline{v} = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 8 \end{pmatrix} \begin{pmatrix} 0 \\ 0.001 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.0001 \\ 0.012 \\ 8.00001 \end{pmatrix} \ .$$

Hence

$$\frac{\underline{v}^T A \underline{v}}{\underline{v}^T \underline{v}} = \frac{8.000022}{1 + 0.00001} = \frac{8.000022}{1.000001} = 8.000014..$$

giving an interval $(8.000014, 8.0000169)$ containing the actual eigenvalue 8.00001667.

8.7 Further examples

(1) We consider estimating the smallest eigenvalue of the following symmetric tri-diagonal matrix.

$$A = \begin{pmatrix} 10 & 2 & 0 & 0 \\ 2 & 7 & 1 & 0 \\ 0 & 1 & 4 & 10^{-2} \\ 0 & 0 & 10^{-2} & 1 \end{pmatrix},$$

Gerschgorin's theorem implies that the eigenvalues are in the union of the disks

$$|z-10| \le 2, \quad |z-7| \le 3, \quad |z-4| \le 1+10^{-2} \quad \text{and} \quad |z-1| \le 10^{-2}\ .$$

Since the eigenvalues of a real symmetric matrix are real the eigenvalues lie in the union of the intervals

$$8 \le x \le 12, \quad 4 \le x \le 10, \quad 3-10^{-2} \le x \le 5+10^{-2}, \quad 1-10^{-2} \le x \le 1+10^{-2}\ .$$

The last interval is not connected to the other 3 intervals and thus there is exactly 1 eigenvalue in this interval and this is the smallest eigenvalue of A.

Our next aim is to sharpen the bounds on the smallest eigenvalue. By the Rayleigh quotient result the smallest eigenvalue is less than or equal to $(\underline{v}^T A\underline{v})/(\underline{v}^T\underline{v})$ for all $\underline{v}$. If we take $\underline{v} = \underline{e}_4$, the 4th column of the 4×4 identity matrix, then the Rayleigh quotient is the $4,4$ entry of A. Since this vector is not an eigenvector we have that the smallest eigenvalue is in the interval

$$1-10^{-2} \le x < 1\ .$$

The bound can be improved by applying Gerschgorin's theorem to a matrix of the form $D^{-1}AD$ where D is a suitably chosen diagonal matrix. It is the 4th Gerschgorin disk that we wish to make smaller and thus we consider a diagonal matrix of the form $D = \mathrm{diag}\{1,1,1,d\}$. In this case

$$D^{-1}AD = \begin{pmatrix} 10 & 2 & 0 & 0 \\ 2 & 7 & 1 & 0 \\ 0 & 1 & 4 & 10^{-2}d \\ 0 & 0 & 10^{-2}/d & 1 \end{pmatrix}\ .$$

The 3rd and 4th disks are

$$|z-4| \le (1+10^{-2}d) \quad \text{and} \quad |z-1| \le 10^{-2}/d\ .$$

Observe that for $d > 1$ the 4th disk decreases in size whilst the 3rd disk increases in size. By inspection the disks are disjoint when $d = 100$ but not when $d = 200$. However when $d = 200$ the disk centred at $x = 4$ still does not include the interval $(1-10^{-2}, 1)$ and thus in this case we can deduce that the smallest eigenvalue lies in the interval

$$1-10^{-4}/2 < x < 1\ .$$

(2) We consider now the use of just Gerschgorin's theorem to estimate the smallest eigenvalue of the following non-symmetric tri-diagonal matrix.

$$A = \begin{pmatrix} 11 & 3 & 0 & 0 \\ 3 & 10 & 2 & 0 \\ 0 & 1 & 5 & 10^{-2} \\ 0 & 0 & 10^{-2} & 2 \end{pmatrix},$$

By Gerschgorin's theorem the smallest eigenvalue λ lies in the 4th disk since this disk is not connected to the other 3 disks. Thus λ satisfies $|\lambda - 2| \le 10^{-2}$.

The bound can be improved by applying Gerschgorin's theorem to a matrix of the form $D^{-1}AD$ where D is a suitably chosen diagonal matrix as in the previous example. As in the previous example it is the 4th Gerschgorin disk that we wish to make smaller and thus we take $D = \mathrm{diag}\{1, 1, 1, d\}$ which gives

$$D^{-1}AD = \begin{pmatrix} 11 & 3 & 0 & 0 \\ 3 & 10 & 2 & 0 \\ 0 & 1 & 5 & 10^{-2}d \\ 0 & 0 & 10^{-2}/d & 2 \end{pmatrix} .$$

The 3rd and 4th disks are now

$$|z-5| \le 1 + 10^{-2}d \quad \text{and} \quad |z-2| \le 10^{-2}/d \; .$$

The boundaries of these disks touch when d is such that

$$5 - (1 + 10^{-2}d) = 2 + 10^{-2}/d$$

That is

$$d^2 - 200\,d + 1 = 0 \; .$$

Solving the quadratic gives

$$d = \frac{200 \pm (200^2 - 4)^{1/2}}{2} = 100 \pm (100^2 - 1)^{1/2} \; .$$

There are two solutions since you can expand disk 4 and decrease disk 3 or you can decrease disk 4 and expand disk 3. The latter case is what we want to do here and corresponds to $d > 1$, i.e. to the + sign in the above. Thus

$$d = 100 + (100^2 - 1)^{1/2} = 100\left(1 + (1 - 0.01)^{1/2}\right)$$

and the disk in which the smallest eigenvalue lies is

$$|z-2| \le \frac{10^{-4}}{1 + (1-0.01)^{1/2}} < 6 \times 10^{-5} \; .$$

8.8 Some key results of the section

1. Revision:

   $\underline{v} \ne \underline{0}$, $A\underline{v} = \lambda\underline{v}$ is the eigenvalue/eigenvector relation.

   A and $S^{-1}AS$ are similar matrices and have the same eigenvalues.

   When A is real and symmetric the eigenvalues are real and the eigenvectors can be chosen to form an orthonormal basis for $\mathbb{R}^n$.

2. Every eigenvalue of $A = (a_{ij})$ lies in at least one Gerschgorin disk

$$\{z \in \mathbb{C} : \quad |z - a_{ii}| \le \sum_{j \ne i} |a_{ij}|\}.$$

Isolated disks contain exactly one eigenvalue. By considering $D^{-1}AD$ for an appropriate diagonal matrix D we can usually refine the estimate of the eigenvalue in this case.

3. If $\underline{x} \ne \underline{0}$ then
$$\frac{\underline{x}^T A\underline{x}}{\underline{x}^T\underline{x}}$$
is the Rayleigh quotient of A and $\underline{x}$. When $\underline{x}$ is an eigenvector the Rayleigh quotient gives the corresponding eigenvalue.

   When A is real and symmetric we have the following:
$$g(t) = \|A\underline{x} - t\underline{x}\|_2^2 \quad \text{has a global minimum at } t = \frac{\underline{x}^T A\underline{x}}{\underline{x}^T\underline{x}}.$$

   If the eigenvalues of A are $\lambda_n \le \lambda_{n-1} \le \lambda_1$ then for all $\underline{x} \ne \underline{0}$ the Rayleigh quotient bounds are
$$\lambda_n \le \frac{\underline{x}^T A\underline{x}}{\underline{x}^T\underline{x}} \le \lambda_1.$$

9 Methods for computing eigenvalues

In this module the only numerical methods that we will consider in any detail for attempting to accurately compute eigenvalues are the Power Method and the related Inverse Power Method. These determine respectively the largest eigenvalue in magnitude and the eigenvalue nearest to a given number. We will then review some of the other methods used in numerically computing eigenvalues. In this context you should note the name 'QR method' (which involves repeatedly determining QR factorizations) which is the established technique used in packages for computing all the eigenvalues and eigenvectors of a dense matrix.

9.1 The Power Method

This involves the following. Given a vector $\underline{x}_0 \neq \underline{0}$, which we can arrange to be normalised with respect to the Euclidean norm (i.e. $\|\underline{x}_0\|_2 = 1$), we generate the following sequence of unit vectors.

$$\underline{y}_k = A\underline{x}_k, \quad \underline{x}_{k+1} = \underline{y}_k/\|\underline{y}_k\|_2, \quad k = 0, 1, 2, \cdots .$$

For each k we also compute the scalar

$$\mu_k = (\underline{x}_k)^T \underline{y}_k \quad (= (\underline{x}_k)^T A\underline{x}_k = \text{Rayleigh quotient of } A \text{ with } \underline{x}_k \text{ since } \|\underline{x}_k\|_2 = 1).$$

The method hence just involves repeatedly pre-multiplying by A and normalising by dividing by the appropriate positive number. In this way we obtain vectors which are respectively parallel to $A\underline{x}_0$, $A^2\underline{x}_0$, $A^3\underline{x}_0$, $\cdots$, and have unit length. Hence the vector $\underline{x}_k$ that we obtain is the vector

$$\underline{x}_k = \frac{1}{\|A^k\underline{x}_0\|_2} A^k \underline{x}_0$$

and in the case of a symmetric matrix A the scalar μ_k is

$$\mu_k = \underline{x}_k{}^T A \underline{x}_k = \frac{1}{\|\underline{x}_0{}^T A^{2k}\underline{x}_0\|_2} \underline{x}_0{}^T A^{2k+1} \underline{x}_0 .$$

We consider now what happens as $k \to \infty$ and, not to make the description over technical, we assume that A has n linearly independent eigenvectors $\underline{v}_1, \cdots, \underline{v}_n$, which we take to be normalised with respect to the Euclidean norm, with $\lambda_1, \cdots, \lambda_n$ as the corresponding eigenvalues which we take to be ordered so that

$$|\lambda_n| \leq |\lambda_{n-1}| \leq |\lambda_2| \leq |\lambda_1| .$$

As is usual in this type of argument we express $\underline{x}_0$ in terms of the eigenvectors and proceed in a similar way to the proof of the convergence of the iteration $\underline{x}_{k+1} = G\underline{x}_k + \underline{c}$ in the case of iterative methods for linear systems. We have

$$\underline{x}_0 = \sum_1^n c_i \underline{v}_i$$

to give

$$A^k \underline{x}_0 = \sum_1^n c_i \lambda_i^k \underline{v}_i = \lambda_1^k \sum_1^n c_i (\lambda_i/\lambda_1)^k \underline{v}_i = \lambda_1^k \left(c_1 \underline{v}_1 + \sum_2^n c_i (\lambda_i/\lambda_1)^k \underline{v}_i \right) .$$

Define

$$\underline{z}_k := \frac{1}{\lambda_1^k} A^k \underline{x}_0 = c_1 \underline{v}_1 + \sum_2^n c_i (\lambda_i/\lambda_1)^k \underline{v}_i$$

which is parallel to $\underline{x}_k$. If $c_1 \neq 0$ and there is a single dominant eigenvalue, i.e. $|\lambda_2| < |\lambda_1|$ then $(\lambda_i/\lambda_1)^k \to 0$ as $k \to \infty$ for $i = 2, \cdots, n$ and

$$\underline{z}_k = \frac{1}{\lambda_1^k} A^k \underline{x}_0 \to c_1 \underline{v}_1 \quad \text{as } k \to \infty.$$

Hence if $\lambda_1 > 0$ then

$$\underline{x}_k \to \begin{cases} \underline{v}_1, & \text{if } c_1 > 0, \\ -\underline{v}_1, & \text{if } c_1 < 0. \end{cases}$$

If $\lambda_1 < 0$ then the vectors $\underline{x}_k$ will approximately alternate between the directions $\underline{v}_1$ and $-\underline{v}_1$ with the two subsequences $(\underline{x}_{2k})$ and $\left(\underline{x}_{2k+1}\right)$ both converging. In both cases if A has a single dominant eigenvalue then

$$\mu_k \to \underline{v}_1^T A \underline{v}_1 = \lambda_1 \ .$$

Thus to summarize, when there is a single dominant eigenvalue the Power Method converges giving the eigenvalue λ_1 and the corresponding eigenvector $\underline{v}_1$ provided $c_1 \neq 0$.

Remarks

(i) The rate of convergence depends on how rapidly

$$\underline{z}_k = c_1 \underline{v}_1 + \sum_2^n c_i (\lambda_i/\lambda_1)^k \underline{v}_i \to c_1 \underline{v}_1$$

which in turn depends on the ratio $|\lambda_2/\lambda_1|$. If this is small then we get rapid convergence but when it is close to 1 the convergence is slow. In applications the slow convergence case is likely the larger is the size of the problem, i.e. the larger is the value n.

(ii) For the method to converge to the eigenvalue of largest modulus it is necessary that the initial vector $\underline{x}_0$ has a component in the direction of the corresponding eigenvector, i.e. in the above $c_1 \neq 0$. In practice it is unlikely that you would know whether or not this is the case. However in practice this is not usually a drawback since even if $\underline{x}_0$ has no component in this direction then it is highly probable that the subsequent iterates $\underline{x}_1, \underline{x}_2, \cdots$ will have a component in this direction as a consequence of rounding errors in the floating point arithmetic. Although this is not guaranteed, it is a benefit of inexact arithmetic.

(iii) The normalising condition of dividing $\underline{y}_k$ by $\|\underline{y}_k\|_2$ can be replaced by any other suitable normalizing condition.

(iv) If we do not have $|\lambda_2| < |\lambda_1|$, i.e. we do not have a single dominant eigenvalue, then it is often still possible to extract from the iterates $(\underline{x}_k)$ the largest eigenvalue although this will not be considered here. The typical situations of this type are when $\lambda_2 = -\lambda_1$ and when λ_1 and λ_2 are a pair of complex conjugate eigenvalues.

9.2 The Inverse Power Method

This method involves applying the ordinary power method to a related matrix and it is described as follows. If $\lambda_1, \cdots, \lambda_n$ are the eigenvalues of A with eigenvectors $\underline{v}_1, \cdots, \underline{v}_n$ respectively then

$$A\underline{v}_i = \lambda_i \underline{v}_i \quad \text{and} \quad (A - pI)\underline{v}_i = (\lambda_i - p)\underline{v}_i \quad \text{and} \quad (A - pI)^{-1}\underline{v}_i = (\lambda_i - p)^{-1}\underline{v}_i \ .$$

Hence if the ordinary power method is applied to the matrix $(A-pI)^{-1}$ then when the method converges it gives the largest eigenvalue in modulus of the matrix $(A-pI)^{-1}$. That is it gives the largest in modulus of the n numbers

$$\mu_1 = \frac{1}{\lambda_1 - p}, \quad \mu_2 = \frac{1}{\lambda_2 - p}, \quad \cdots, \mu_n = \frac{1}{\lambda_n - p} .$$

If λ_r is the nearest eigenvalue of A to p then μ_r is the largest eigenvalue of $(A-pI)^{-1}$. In the case that p is very close to λ_r then $|\mu_r|$ is likely to be much larger than any of the other numbers $|\mu_i|$ and the convergence is rapid.

When the method is implemented an LU factorization of $A - pI$ is first computed. Then at each step

$$\underline{y}_k = (A - pI)^{-1}\underline{x}_k$$

is computed by solving

$$(A - pI)\underline{y}_k = \underline{x}_k$$

using the LU factorization.

9.3 The Rayleigh Quotient Iteration

As we have just seen, the Inverse Power Method will converge faster than the ordinary power method if the shift p is close to an eigenvalue of A although it may not improve things much if we do not have a good candidate for p. The Rayleigh Quotient Iteration overcomes this shortcoming by using the observation that if $\underline{x}_k$ is an approximate eigenvalue then the Rayleigh quotient is an approximate eigenvalue and thus this can be used as the shift. In fact each iteration gives us a better shift to use. That is we can successively improve the shift as the iteration improves instead of keeping a fixed shift as with the Inverse Power Method. The algorithm is as follows.

Given some vector $\underline{x}_0$ with $\|\underline{x}_0\|_2 = 1$ we do the following.

$p_0 = \underline{x}_0^T A \underline{x}_0$=Rayleigh quotient (the initial shift).

For $k = 0, 1, 2, \cdots$ until convergence.

> Solve $(A - p_k I)\underline{y}_k = \underline{x}_k$.
>
> $\underline{x}_{k+1} = \underline{y}_k / \|\underline{y}_k\|_2$.
>
> $p_{k+1} = \underline{x}_{k+1}^T A \underline{x}_{k+1}$=Rayleigh quotient (the next shift).

Repeat.

It can be shown that this iteration converges very rapidly. In fact it can be shown that it converges at a cubic rate which in particular means that if the error is $\mathcal{O}(\epsilon)$ at one step then it is likely to be $\mathcal{O}(\epsilon^3)$ at the next step.

9.4 Numerical Examples with the Power Method

Consider the matrices

$$A_1 = \begin{pmatrix} 1 & -2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 8 \end{pmatrix} \quad \text{and the } n \times n \text{ matrix} \quad A_2 = \begin{pmatrix} 2 & 1 & & & \\ 1 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2 & 1 \\ & & & 1 & 2 \end{pmatrix}.$$

A_1 was considered in the Gerschgorin theorem examples. Using Gerschgorin theorem and the Rayleigh quotient it was deduced that the largest eigenvalue is in the interval $|x - 8| \le 0.26795$. By using MATLAB we can perform a few iterations of the Power Method to numerically determine the largest eigenvalue more accurately. We use the following script file.

```
% set matrix A1
A1=[1 -2 0; 2 2 1; 0 1 8];

% set  starting vector
x=ones(3,1);

% perform 16 iterations
for k=1:16
   y=A1*x;
   mu=x'*y;
   x=y/norm(y,2);
   fprintf('k=%2d, mu=%18.10e\n',k,mu);
end

% compare with the values produced by the eig function
format compact
format long
eig(A1)
```

This generates the following.

```
k= 1, mu= 1.3000000000e+001
k= 2, mu= 7.3738317757e+000
k= 3, mu= 8.0058368828e+000
k= 4, mu= 8.1241321368e+000
k= 5, mu= 8.1461298554e+000
k= 6, mu= 8.1497655574e+000
k= 7, mu= 8.1495356059e+000
k= 8, mu= 8.1491630720e+000
k= 9, mu= 8.1490565480e+000
k=10, mu= 8.1490519694e+000
k=11, mu= 8.1490596349e+000
k=12, mu= 8.1490627163e+000
k=13, mu= 8.1490631289e+000
k=14, mu= 8.1490630056e+000
k=15, mu= 8.1490629266e+000
k=16, mu= 8.1490629097e+000
ans =
  1.42546854352432 + 1.93276324163023i
  1.42546854352432 - 1.93276324163023i
  8.14906291295136
```

Convergence here is reasonably fast. Modifying the script to perform the iteration with A_2 in the case $n = 100$ results in a much slower convergence.

```
% set matrix A2
n=100;
o=ones(n,1);
A2=spdiags([o,2*o,o],-1:1,n,n);

% set  starting vector
x=ones(n,1);

% perform 1000 iterations
for k=1:1000
   y=A2*x;
   mu=x'*y;
   x=y/norm(y,2);
   if(floor(k/100)*100==k)
     fprintf('k=%4d, mu=%18.10e\n',k,mu);
   end
end

% compare with the values produced by the eig function
format compact
format long
max(eig(A2))
```

```
k= 100, mu= 3.9981211344e+000
k= 200, mu= 3.9985606328e+000
k= 300, mu= 3.9987438301e+000
k= 400, mu= 3.9988449437e+000
k= 500, mu= 3.9989074590e+000
k= 600, mu= 3.9989481975e+000
k= 700, mu= 3.9989753986e+000
k= 800, mu= 3.9989937575e+000
k= 900, mu= 3.9990062040e+000
k=1000, mu= 3.9990146564e+000
ans =
   3.99903256458398
```

The μ values soon get close to the exact value but the ultimate rate of convergence is slow.

The above situation is improved by performing the Inverse Power Method. By using $p = 4$, i.e. by applying the Power Method to $(A - 4I)^{-1}$, the script and the results are as follows.

```
% set matrix A2 and matrix B for the inverse power method
n=100;
o=ones(n,1);
A2=spdiags([o,2*o,o],-1:1,n,n);
B=A2-4*speye(n);

% set  starting vector
x=ones(n,1);

% perform 8 iterations
for k=1:8
   y=B\x;
   mu=x'*y;
   muA=4+1/mu;
   x=y/norm(y,2);
   fprintf('k=%4d, mu=%24.16e, muA=%24.16e\n',k,mu,muA);
end

% compare with the values produces by the eig functions
% which is also equal to 2*(1+cos(pi/(n+1)))
format compact
format long
l1=max(eig(A2));
l2=2*(1+cos(pi/(n+1)));
fprintf('\n largest eigenvalue=%24.16e,   %24.16e\n',l1,l2);
```

```
k=   1, mu=-8.5850000000000070e+004, muA= 3.9999883517763540e+000
k=   2, mu=-1.0323273867869060e+003, muA= 3.9990313150529580e+000
k=   3, mu=-1.0336450470690480e+003, muA= 3.9990325499040160e+000
k=   4, mu=-1.0336605389540280e+003, muA= 3.9990325644035780e+000
k=   5, mu=-1.0336607293194090e+003, muA= 3.9990325645817480e+000
k=   6, mu=-1.0336607316708550e+003, muA= 3.9990325645839480e+000
k=   7, mu=-1.0336607316999190e+003, muA= 3.9990325645839760e+000
k=   8, mu=-1.0336607317002790e+003, muA= 3.9990325645839760e+000

 largest eigenvalue= 3.9990325645839780e+000,    3.9990325645839760e+000
```

This iteration converges rapidly. In fact, for this special problem, all the eigenvalues and eigenvectors can be found exactly without any need for numerical techniques. It can be shown that the eigenvalues are $\lambda_k = 2(1+\cos(k\pi/(n+1)))$, $k = 1, 2, \cdots, n$ with the largest corresponding to $k = 1$.

9.5 A survey of other methods and techniques

We consider here briefly methods and techniques used in the case of dense matrices to attempt to determine the eigenvalues and eigenvectors. Much of the details given will be descriptive without too much technical detail. The material covered here is not examinable.

First some theory. Recall that if S is any invertible matrix then A and $B = S^{-1}AS$ are similar matrices and hence have the same eigenvalues. If for a given matrix A you consider the set of all matrices which are similar to A (technically this is an equivalence class) then an obvious question is which member of this set has the 'simplest' form. If a member of the set is triangular then the diagonal entries of this matrix are the eigenvalues of A. In the context of numerically solving the eigenvalue problem it turns out that this is the most useful form. There is a result known as **Schur's theorem** (probably a result you have not met before) which states that there is always a (complex) unitary matrix U and a (complex) triangular matrix T such that

$$\bar{U}^T AU = T, \quad U^{-1} = \bar{U}^T \text{ is the unitary condition.}$$

In the above the over bar indicates the complex conjugate. A unitary matrix is slight generalization of a real orthogonal matrix. The above result applies to complex $n \times n$ matrices. In the case of real matrices, a consequence of the result is that there is an orthogonal matrix Q such that

$$Q^T AQ = \hat{T} = \text{block upper triangular .}$$

The diagonal blocks of $\hat{T}$ are of size 1×1 (a real eigenvalue) or 2×2 (which contains information from which a pair of complex conjugate eigenvalues can be found). Thus when all the eigenvalues are real we have an upper triangular matrix. Further, when A is symmetric we have this situation of all the eigenvalues being real and additionally, by transposing the above relation and using the property $A^T = A$, we find that the upper triangular matrix is a diagonal matrix. That is we have

$$Q^T AQ = \text{diag}\{\lambda_1, \cdots, \lambda_n\}$$

which is an important result which was mentioned in the first section of these notes. Note that the columns of Q are the orthonormal eigenvectors of A.

Given the above theory, the numerical methods used attempt to generate an orthogonal matrix Q such that

$$Q^T AQ = \text{diag}\{\lambda_1, \cdots, \lambda_n\}$$

when A is symmetric and

$$Q^T A Q = \text{block upper triangular}$$

when A is non-symmetric. As we have already indicated, this can only be achieved in general by an iteration. The computational cost of any procedure thus depends on the cost of each iteration and the number of iterations required. To reduce the cost of each iteration it is beneficial to first reduce a dense matrix A to a 'sparser form'.

9.5.1 Reducing to a sparser form

By pre-multiplying A by an appropriate sequence of Householder matrices $P_1, P_2, \cdots, P_{n-2}$ we can eliminate entries below the sub-diagonal in a similar way to the technique for computing a QR factorization which involves eliminating below the diagonal. We obtain

$$(P_{n-2} \cdots P_2 P_1) A = \text{upper Hessenburg}$$

where upper Hessenburg is the name of a matrix which is zero below the sub-diagonal. As we need to have a similar matrix to A we also need to post-multiply by these matrices. The order of the computations is usually to generate the matrices

$$P_k \cdots P_1 A P_1 \cdots P_k, \quad k = 1, 2, \cdots, n-2.$$

The final matrix

$$A_1 = Q_0^T A Q_0 = \begin{cases} \text{symmetric tri-diagonal when } A \text{ is symmetric} \\ \text{upper Hessenburg when } A \text{ is non-symmetric} \end{cases}$$

where $Q_0 = P_1 P_2 \cdots P_{n-2}$. In the case $n = 5$ and a symmetric matrix, the similar matrices A, $P_1 A P_1$, $P_2(P_1 A P_1) P_2$ and $P_3(P_2 P_1 A P_1 P_2) P_3$ respectively have the structure

$$\begin{pmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{pmatrix} \to \begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \end{pmatrix} \to \begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x & x & x \end{pmatrix} \to \begin{pmatrix} x & x & 0 & 0 & 0 \\ x & x & x & 0 & 0 \\ 0 & x & x & x & 0 \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \end{pmatrix}$$

In what follows we restrict attention to the case of a symmetric matrix A.

9.5.2 The QR algorithm

The Basic Algorithm

The basic QR algorithm is simple to describe but it is difficult to analyse. The algorithm is as follows.

Reduce A to A_1 which is tri-diagonal as described above.

For $k = 1, 2, \cdots$ (until convergence).

$$A_k = Q_k R_k \qquad \text{(compute the factorization)}$$
$$A_{k+1} = R_k Q_k \qquad \text{(multiply the factors in this order)}$$

Repeat.

When A is symmetric it can be shown that each of the sequence of matrices A_1, A_2, $\cdots$ is symmetric tri-diagonal with each step involving $\mathcal{O}(n)$ operations. Thus each iteration is 'cheap'. Also

$$A_{k+1} = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_K^T A_k Q_k$$

is similar to A_k. Thus all the matrices are similar and, from previous results relating to norms and orthogonal matrices, all matrices have the same 2−norm. Under fairly mild assumptions the sequence of matrices converges to a diagonal matrix or another matrix from which all the eigenvalues are obtained immediately. The basic iteration however may be slow to converge.

The QR algorithm with shifts

A simple modification to the basic algorithm produces an iteration which converges much more rapidly. The algorithm is as follows.

Reduce A to A_1 which is tri-diagonal as described above.

For $k = 1, 2, \cdots$ (until convergence).

Select a shift parameter p_k.

$A_k - p_k I = Q_k R_k$ (compute the factorization)

$A_{k+1} = R_k Q_k + p_k I$ (multiply the factors in this order and add $p_k I$.)

Repeat.

Observe that in this case

$$A_{k+1} = R_k Q_k + p_k I = Q_k^T (Q_k R_k + p_k I) Q_k = Q_k^T A_k Q_k$$

and thus all the matrices are similar. The matrices are also all symmetric tri-diagonal and have the same 2−norm. It is the careful selection of the shift parameters p_k (which we will not consider here) which gives a rapidly convergent iteration.

9.5.3 Computing selected eigenvalues

None of the methods described up to this point have involved trying to directly solve the equation

$$\det(\lambda I - A) = 0 \ .$$

However, in the symmetric tri-diagonal case there is a method based on properties of what are known as 'Sturm sequences' from which we can deduce a set of intervals with each interval containing exactly 1 eigenvalue. When A is tri-diagonal, the computation of $f(t) := \det(tI - A)$ for each value of t used in an iterative solution procedure involves only $\mathcal{O}(n)$ operations instead of $\mathcal{O}(n^3)$ in the case of a full matrix. In each interval a bracketing technique or some other root finding technique can be used to determine the solutions of $f(t) = 0$. This idea is used in some packages and it is a good technique in situations when a only 'a few' of the n eigenvalues are required. However when 'many' or 'most' of the eigenvalues are required it becomes more efficient to compute all of them using the QR algorithm.

9.5.4 The Jacobi method

Not all the methods for computing eigenvalues have been developed in the computer era. In 1845 Jacobi produced an iterative method which involved computing a sequence of similar matrices

$$A_k = G_k^T G_{k-1}^T \cdots G_1^T A G_1 \cdots G_{k-1} G_k$$

where $G_1, G_2, \cdots$ are Givens' matrices. In this way all matrices have the same 2−norm and also the same Euclidean matrix norm, i.e.

$$\|A_k\|_E^2 = \|A\|_E^2, \quad k = 1, 2, \cdots$$

Givens' matrices are selected at each stage so that when A_k is determined from A_{k-1} the sum of the squares of the off-diagonal entries for A_k is less than the corresponding quantity for A_{k-1}. The off-diagonal entries hence, on average, become smaller and smaller and under fairly mild assumptions on the way this is done we can generate a sequence of matrices which converge to a diagonal matrix. This is a simple method to implement but it is not as good as the QR method although it has gained popularity recently as it can be implemented efficiently on a parallel computer.

9.5.5 Computing eigenvectors

The power method or the related inverse power method determine one of the eigenvalues and one of the eigenvectors. To determine all the eigenvectors is straight forward for a real symmetric matrix but it can be tricky for a general matrix.

In the symmetric case and when the method used is the 'QR algorithm' the sequence of matrices are of the form

$$A_k = \tilde{Q}_k^T A \tilde{Q}_k, \quad \text{where } \tilde{Q}_k = Q_0 Q_1 Q_2 \cdots Q_{k-1} .$$

When A_k is approximately a diagonal matrix the columns of $\tilde{Q}_k$ are approximately the orthonormal eigenvectors of A.

In the QR algorithm in the case of a non-symmetric matrix the sequence of matrices only converges to a block upper triangular matrix. We hence get the eigenvalues but more work has to be done to also get the corresponding eigenvectors. Difficulties can arise here as it need not be the case that the matrix A has n linearly independent eigenvectors. To state this another way, there may not exist a matrix S such that

$$B = S^{-1} A S = \text{diagonal matrix}.$$

When this occurs we say that A is **non-diagonalizable**, **deficient** or **defective**. For example this is the case with the 2×2 matrix

$$A = \begin{pmatrix} d & 1 \\ 0 & d \end{pmatrix}$$

which has d as an eigenvalue. In this case the eigenvalue d has algebraic multiplicity 2 since

$$\det(tI - A) = (t - d)^2$$

and geometric multiplicity 1 corresponding to the span of the vector $(1, 0)^T$. This example is typical of the deficient case. For any matrix A, it can be shown that we always have that for each eigenvalue that

$$1 \le \text{geometric multiplicity} \le \text{algebraic multiplicity}$$

and the deficient situations correspond precisely to the cases when A has a multiple eigenvalue with

$$1 \le \text{geometric multiplicity} < \text{algebraic multiplicity} .$$

(Note that for real matrices, a deficient matrix is necessarily not symmetric.)

For an arbitrary matrix A, perhaps randomly generated, it is unlikely that it is deficient but in numerical analysis this precise distinction between deficient or non-deficient is replaced by being 'nearly deficient' or 'far from deficient'. When a matrix is nearly deficient the eigenvalue problem becomes more difficult to solve accurately by numerical techniques.

9.6 Some key results of the section

1. Given $\underline{x}_0 \neq \underline{0}$ the power method generates the unit vectors

$$\underline{x}_k = \frac{1}{\|A^k \underline{x}_0\|} A^k \underline{x}_0, \quad k = 1, 2, \cdots .$$

When A has a single dominant eigenvalue λ_1 we generally have $\underline{x}_k^T A \underline{x}_k \to \lambda_1$ as $k \to \infty$ with $(\underline{x}_k)$ converging to the direction of the corresponding eigenvector.

2. If A has eigenvalues $\lambda_1, \cdots, \lambda_n$ then $(A - pI)^{-1}$ has eigenvalues $(\lambda_1 - p)^{-1}, \cdots, (\lambda_n - p)^{-1}$. The inverse power method is the power method applied to $(A - pI)^{-1}$ and generates the nearest eigenvalue to the shift p.

EXERCISES ON THE BACKGROUND SECTION AND DIRECT METHODS

In the following questions 1–4 are relatively short and involve manipulation with rank one matrices. Questions 5 and 6 are entire questions from the 2001 and 2000 papers. Questions 7, 11, 8 and 9 are parts of questions from 1999, 2001, 1997 and 1994 respectively. Questions 11, 10 and 12 relate to the Cholesky factorization. There will be several further questions involving the Cholesky factorization in the least squares section to be covered later.

1. If A is a $n \times n$ matrix with eigenvalues $\lambda_1, \cdots, \lambda_n$ and corresponding eigenvectors $\underline{v}_1, \cdots, \underline{v}_n$ then show that

$$A^m \underline{v}_i = \lambda_i^m \underline{v}_i, \quad i = 1, \cdots, n \quad \text{for all positive integers } m$$

and that

$$A^{-1} \underline{v}_i = \lambda_i^{-1} \underline{v}_i \quad i = 1, \cdots, n$$

assuming that A is invertible.

---

2. Let $\underline{v}$ be a real vector of unit Euclidean length i.e. $\underline{v}^T \underline{v} = 1$, and let S be the symmetric matrix

$$S = I - \underline{v}\,\underline{v}^T \ .$$

(a) Show that $S\underline{v} = \underline{0}$. What can you say about the inverse of S?

(b) Show that $S = S^2 = S^3 = \cdots = S^n = \cdots$ for all integers n.

---

3. Let $\underline{v}$ be a real vector of unit Euclidean length i.e. $\underline{v}^T \underline{v} = 1$, and let P be the symmetric matrix

$$P = I - 2\underline{v}\,\underline{v}^T \quad \text{(this is called a Householder matrix)} \ .$$

(a) Show that $P^2 = I$, i.e. $P = P^T = P^{-1}$. The matrix is thus equal to its inverse and its transpose.

(b) For a given vector $\underline{b}$, solve for $\underline{x}$ the linear system $P\underline{x} = \underline{b}$, i.e. express the solution $\underline{x}$ in terms of $\underline{v}$ and $\underline{b}$.

---

4. The previous two questions have considered specific matrices involving the identity matrix and a rank one matrix. We now consider how to get the inverse of such a matrix (when it exists) in general.

Let $\underline{u}, \underline{v} \in \mathbb{R}^n$ be such that $1 + \underline{v}^T \underline{u} \neq 0$ and let

$$A = I + \underline{u}\,\underline{v}^T \quad \text{and} \quad B(\alpha) = I + \alpha \underline{u}\,\underline{v}^T \ .$$

By considering the matrix product $AB(\alpha)$, show that

$$A^{-1} = B(\alpha) \quad \text{when } \alpha = \frac{-1}{1 + \underline{v}^T \underline{u}} \ .$$

---

5. (The following was question 1 of the 2001 MA2051S paper.)

(i) Let $\underline{u}$ and $\underline{v}$ be vectors in $\mathbb{R}^n$ which are such that $1+\underline{v}^T\underline{u} \neq 0$. Show that

$$(I+\underline{u}\,\underline{v}^T)^{-1} = (I+\alpha\underline{u}\,\underline{v}^T) \quad \text{where} \quad \alpha = \frac{-1}{1+\underline{v}^T\underline{u}}\,.$$ **[4 MARKS]**

(ii) The $n \times n$ Gauss transformation matrix M_k is of the form

$$M_k = I - \underline{m}_k\,\underline{e}_k^{\,T},$$

where I is an $n \times n$ identity matrix, $\underline{e}_k$ is the kth column of I and $\underline{m}_k$ is an $n \times 1$ vector of the form

$$\underline{m}_k = (0,\cdots,0,m_{k+1,k},\cdots,m_{n,k})^T,$$

i.e. with the first k components as 0.

(a) Show that

$$M_k^{-1} = I + \underline{m}_k\,\underline{e}_k^{\,T}\,.$$

[1 MARK]

(b) For $2 \le k \le n-1$ explain why

$$(I+\underline{m}_1\underline{e}_1^{\,T}+\cdots+\underline{m}_{k-1}\underline{e}_{k-1}^{\,T})(I+\underline{m}_k\underline{e}_k^{\,T}) = I+\underline{m}_1\underline{e}_1^{\,T}+\cdots+\underline{m}_k\underline{e}_k^{\,T}\,.$$ **[2 MARKS]**

Hence show that

$$M_1^{-1}M_2^{-1}\cdots M_{n-1}^{-1} = I + \underline{m}_1\,\underline{e}_1^{\,T} + \underline{m}_2\,\underline{e}_2^{\,T} + \cdots + \underline{m}_{n-1}\,\underline{e}_{n-1}^{\,T}\,.$$

[2 MARKS]

(c) Consider the following factorization

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

State the corresponding factorization of the 3×3 principal submatrix of A.

[2 MARKS]

By making use of the factorization, solve the following system by forward and back substitution

$$A\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 4 \\ 0 \end{pmatrix}$$

and hence give the third column of A^{-1}.

[8 MARKS]

Give the determinant of A.

[1 MARK]

6. The following was question 1 of the 2000 MA2051S paper.

The $n \times n$ Gauss transformation matrix M_k is of the form

$$M_k = I - \underline{m}_k \, \underline{e}_k{}^T,$$

where I is an $n \times n$ identity matrix, $\underline{e}_k$ is the kth column of I and $\underline{m}_k$ is an $n \times 1$ vector of the form

$$\underline{m}_k = (0, \cdots, 0, m_{k+1,k}, \cdots, m_{n,k})^T,$$

i.e. with the first k components as 0.

(i) Show that

$$M_k^{-1} = I + \underline{m}_k \, \underline{e}_k^T \, .$$

[2 MARKS]

(ii) Show that

$$M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} = I + \underline{m}_1 \, \underline{e}_1{}^T + \underline{m}_2 \, \underline{e}_2{}^T + \cdots + \underline{m}_{n-1} \, \underline{e}_{n-1}{}^T \, .$$

[6 MARKS]

(iii) In the case $n = 4$ describe explicitly the matrix

$$L = I + \underline{m}_1 \, \underline{e}_1{}^T + \underline{m}_2 \, \underline{e}_2{}^T + \underline{m}_3 \, \underline{e}_3{}^T \, .$$

[2 MARKS]

(iv) Consider the following factorization

$$A = \begin{pmatrix} 3 & 2 & 0 & 0 \\ -3 & 2 & 2 & 0 \\ 0 & 2 & 5 & 2 \\ 0 & 0 & -2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & -1 & 3 \end{pmatrix} \begin{pmatrix} 3 & 2 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \, .$$

State the corresponding factorization of the 3×3 principal submatrix of A.

[2 MARKS]

By making use of the factorization, solve the following system by forward and back substitution

$$A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ -6 \\ 8 \\ -2 \end{pmatrix} \, .$$

[7 MARKS]

Give the determinant of A.

[1 MARK]

7. The following were part of question 1 of the 1999 MA2051S paper.

(i) If for an $n \times n$ matrix A it is possible to construct $M_1, \cdots, M_{n-1}$ such that

$$U = (M_{n-1} \cdots M_2 M_1)A$$

is upper triangular then explain why this implies that A can be decomposed into the product $A = LU$ where L is a unit lower triangular matrix.

(ii) Given that

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

use forward and back substitution to solve

$$A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \\ -5 \\ 2 \end{pmatrix} .$$

Give the determinant of A.

---

8. (The following were parts of question 1 of the 1997 MA251S paper.)

The $n \times n$ Gauss transformation matrix M_k and the $n \times n$ Householder matrix P_k are respectively of the form

$$M_k = I - \underline{m}_k \, \underline{e}_k^{\,T}, \quad \text{and} \quad P_k = I - 2\underline{v}_k \, \underline{v}_k^T,$$

where I is an $n \times n$ identity matrix, $\underline{e}_k$ is the kth column of I, $\underline{m}_k$ is an $n \times 1$ vector of the form

$$\underline{m}_k = (0, \cdots, 0, m_{k+1,k}, \cdots, m_{n,k})^T,$$

i.e with the first k components as 0, and $\underline{v}_k$ satisfies $\underline{v}_k^T \underline{v}_k = 1$.

(a) Show that

$$M_k^{-1} = I + \underline{m}_k \, \underline{e}_k^T \quad \text{and} \quad P_k^{-1} = P_k \, .$$

(b) If the vectors $\underline{v}_1, \cdots, \underline{v}_{n-1}$ are chosen so that

$$R = P_{n-1} \cdots P_2 P_1 A$$

is upper triangular then explain why this implies that A can be decomposed into the product $A = QR$ where Q is an orthogonal matrix.

(c) If a factorization $A = LU$ has been computed, with L a unit lower triangular matrix and U an upper triangular matrix, then explain briefly how this can be used to solve the linear system

$$A\underline{x} = \underline{b},$$

and explain how the factorization can be used to compute the determinant of A.

---

9. (This is the last part of question 1 of the 1994 MA351 paper. Many people found this difficult.)

   The $(n+1)\times(n+1)$ matrix B has the partitioned form

$$B = \begin{pmatrix} A & \underline{c} \\ \underline{r}^T & b_{n+1,n+1} \end{pmatrix},$$

   where A is an $n\times n$ matrix, $\underline{c}$ and $\underline{r}$ are $n\times 1$ vectors and $b_{n+1,n+1}$ is a scalar. If A is non singular and has the LU decomposition $A = LU$, which has already been computed, describe how you would obtain the LU decomposition of B.

---

10. Let A be a real symmetric positive definite matrix of size $n\times n$ and let $A = L\,L^T$ denote the Cholesky factorization, i.e. L is a real lower triangular matrix of size $n\times n$ with positive diagonal entries. Show that

$$l_{ij}^2 \le a_{ii}, \quad j = 1, 2, \cdots, i.$$

---

11. (The following was the last half of question 5 of the 2001 MA2051S paper.)

    Obtain the Cholesky decomposition of the positive definite matrix

$$A = \begin{pmatrix} 1 & 2 & -2 & -1 \\ 2 & 8 & -2 & 0 \\ -2 & -2 & 14 & 0 \\ -1 & 0 & 0 & 7 \end{pmatrix}$$

    Hence find the fourth column of A^{-1}.

---

12. (i) Obtain the Cholesky decomposition of the positive definite matrix

$$A = \begin{pmatrix} 1 & -1 & 2 & 0 \\ -1 & 2 & -5 & 1 \\ 2 & -5 & 22 & -6 \\ 0 & 1 & -6 & 3 \end{pmatrix}.$$

    Hence find $\det(A)$ and the second column of A^{-1}.

    (ii) Obtain the Cholesky decomposition of the positive definite matrix

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 2 & 8 & 6 & -2 \\ 0 & 6 & 10 & -4 \\ 0 & -2 & -4 & 6 \end{pmatrix}.$$

    Hence find $\det(A)$ and the first column of A^{-1}.

EXERCISES ON NORMS AND PERTURBATIONS

The following set of questions includes the complete question from the 2001 paper on norms together with parts of the questions set in 2000, 1999 and 1998.

1. (This is Question 2 of the 2001 MA2051S paper.)

   (i) For $n \times n$ real matrices, state the matrix norm axioms and define the matrix norm which is subordinate to a given vector norm.

   [3 MARKS]

   (ii) Define $\|\underline{x}\|_1$, $\|\underline{x}\|_2$ and $\|\underline{x}\|_\infty$ for a $n \times 1$ vector $\underline{x}$

   [1 MARK]

   Show that the matrix norm subordinate to the ∞-vector norm satisfies

   $$\|A\|_\infty = \max_i \left\{ \sum_{j=1}^{n} |a_{ij}| \right\} .$$ **[5 MARKS]**

   (iii) Define the spectral radius $\rho(A)$ of an $n \times n$ matrix A and show that

   $$\rho(A) \leq \|A\| .$$ **[3 MARKS]**

   (iv) Let A denote the $n \times n$ tri-diagonal matrix

   $$A = \begin{pmatrix} 2+d & -1 & & & \\ -1 & 2+d & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2+d & -1 \\ & & & -1 & 2+d \end{pmatrix}$$

   where $d > 0$. Show that the solution $\underline{x}$ of $A\underline{x} = \underline{b}$ satisfies

   $$\frac{\|\underline{b}\|_\infty}{4+d} \leq \|\underline{x}\|_\infty \leq \frac{\|\underline{b}\|_\infty}{d} .$$

   (In your answer you may assume the result

   $$\|(I-E)^{-1}\| \leq \frac{1}{1-\|E\|}$$

   when E is a matrix satisfying $\|E\| < 1$.)

   [6 MARKS]

   Explain, indicating any results that you use, why all the eigenvalues of A lie in the interval $[d, 4+d]$ and all the eigenvalues of A^{-1} lie in the interval $[1/(4+d), 1/d]$.

   [2 MARKS]

2. (i) (part (ii) of question 2 of the 2000 paper)

Show that if E is a square matrix and $\|E\| < 1$, for a subordinate matrix norm, then $I - E$ is non-singular. Also show that

$$\frac{1}{1 + \|E\|} \leq \|(I - E)^{-1}\| .$$

[5 MARKS]

(ii) (part (iv) of question 2 of the 2000 paper)

Let A be the symmetric $n \times n$ tri-diagonal matrix

$$A = \begin{pmatrix} 10 & 1 & & & \\ 1 & 10 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 10 & 1 \\ & & & 1 & 10 \end{pmatrix} .$$

If $\underline{x}$ satisfies $A\underline{x} = \underline{b}$ then show that

$$\frac{\|\underline{b}\|_\infty}{12} \leq \|\underline{x}\|_\infty \leq \frac{\|\underline{b}\|_\infty}{8} .$$

In your answer you may assume the inequality

$$\|(I - E)^{-1}\| \leq \frac{1}{1 - \|E\|}$$

for any matrix E for which the subordinate matrix norm satisfies $\|E\| < 1$.

[6 MARKS]

---

3. (part (iv) of question 2 of the 1999 paper)

Define the condition number $\kappa(A)$ of a non-singular matrix A.

Show that if A is non-singular and

$$A\underline{x} = \underline{b} \quad \text{and} \quad A(\underline{x} + \Delta\underline{x}) = \underline{b} + \Delta\underline{b},$$

where $\underline{b} \neq \underline{0}$, then

$$\frac{\|\Delta\underline{x}\|}{\|\underline{x}\|} \leq \kappa(A)\frac{\|\Delta\underline{b}\|}{\|\underline{b}\|}.$$

---

4. (i) (part (ii) of question 2 of the 1998 paper)

When subordinate matrix norms are used show that if $\|E\| < 1$ then the matrix $I - E$ is non-singular and that for all $m \geq 1$

$$(I - E)^{-1}(I - E^m) = (I + E + E^2 + \cdots + E^{m-1}) .$$

Using this result, or otherwise, show that

$$\frac{1}{1 + \|E\|} \leq \|(I - E)^{-1}\| \leq \frac{1}{1 - \|E\|} .$$

(ii) (part (iii) of question 2 of the 1998 paper)
Define the condition number $\kappa(A)$ of a matrix and show that if A is non-singular and the perturbed matrix $A + \Delta A$ is such that

$$r = \kappa(A)\frac{\|\Delta A\|}{\|A\|} < 1$$

then $A + \Delta A$ is also non-singular with

$$\left(\frac{1}{1+r}\right)\frac{1}{\|A\|} \leq \|(A + \Delta A)^{-1}\| \leq \left(\frac{1}{1-r}\right)\|A^{-1}\| \ .$$

---

5. It can be shown that for an $n \times n$ matrix $A = (a_{ij})$ we have

$$\begin{aligned} \|A\|_\infty &= \max_i \sum_{j=1}^n |a_{ij}|, \\ \|A\|_1 &= \max_j \sum_{i=1}^n |a_{ij}|, \\ \|A\|_2^2 &= \rho(A^T A). \end{aligned}$$

(i) Give a vector $\underline{x}$ with $\|\underline{x}\|_\infty = 1$ such that $\|A\underline{x}\|_\infty = \|A\|_\infty$.

(ii) Give a vector $\underline{x}$ with $\|\underline{x}\|_1 = 1$ such that $\|A\underline{x}\|_1 = \|A\|_1$.

(iii) Give a vector $\underline{x}$ with $\|\underline{x}\|_2 = 1$ such that $\|A\underline{x}\|_2 = \|A\|_2$.

---

6. (This is a miscellaneous set of questions relating to matrix norms. Some of the questions are, in my opinion, more difficult than the previous questions.)

(i) With the usual notation, give explicit expressions for $\|A\|_1$, $\|A\|_\infty$, and the Euclidean norm $\|A\|_E$. Show that $\|A\|_E$ is not a subordinate matrix norm.

(ii) If $\underline{v} \in \mathbb{R}^n$ and $A = \underline{v}\,\underline{v}^T$ then show that $\|A\|_E = \|A\|_2 = \underline{v}^T\underline{v} = \|\underline{v}\|_2^2$.

(iii) If $\underline{v}_1, \cdots, \underline{v}_n$ denote orthonormal eigenvectors of the matrix $A^T A$ with corresponding eigenvalues $\lambda_1, \cdots, \lambda_n$ then show the following.

(a)
$$\lambda_i = \|A\underline{v}_i\|_2^2 \ .$$

(b) For all vectors $\underline{x}$,
$$\underline{x} = \sum_{i=1}^n (\underline{x}^T\underline{v}_i)\underline{v}_i \quad \text{and} \quad \|A\underline{x}\|_2^2 = \sum_{i=1}^n (\underline{x}^T\underline{v}_i)^2\lambda_i \ .$$

(c)
$$\|A\|_2^2 = \max_{1 \leq i \leq n} \lambda_i \ .$$

(iv) Show that
$$\|A\|_2 \leq \|A\|_E \leq n^{1/2}\|A\|_2 \ .$$

(v) Show that

$$\|QA\|_2 = \|AQ\|_2 = \|A\|_2$$

where Q is an orthogonal matrix.

(vi) In the linear system

$$(B + CE)\underline{x} = \underline{b},$$

$B = (b_{ij})$ and $C = (c_{ij})$ are $n \times n$ tri-diagonal matrices with $b_{ii} = 4$, $b_{i-1,i} = b_{i+1,i} = 1$ and $c_{ii} = 1$, $c_{i-1,i} = c_{i+1,i} = 1/2$, E is an $n \times n$ matrix with $\|E\|_\infty < \alpha < 1$, and $\underline{x}$, $\underline{b}$ are $n \times 1$ vectors. By using the result

$$\|(I - E)^{-1}\| \leq \frac{1}{1 - \|E\|},$$

show that $\|B^{-1}\|_\infty \leq 1/2$ and that

$$\|\underline{x}\|_\infty \leq \|\underline{b}\|_\infty/(2 - 2\alpha).$$

EXERCISES ON LEAST SQUARES

The following set of questions include the least squares questions from 2001 and 2000 together with parts of the questions from 1996–1999.

1. The following was question 3 of the 2001 paper.

(i) Let A be an $m \times n$ real matrix, $m > n$, which has n linearly independent columns. Also let $\underline{b} \in \mathbb{R}^m$ and let

$$\phi(\underline{x}) = (\|A\underline{x} - \underline{b}\|_2)^2$$

where $\underline{x} \in \mathbb{R}^n$. Show that $\phi(x)$ can be written as

$$\phi(\underline{x}) = \underline{x}^T C \underline{x} - 2\underline{x}^T \underline{d} + \underline{b}^T \underline{b} \quad \text{where} \quad C = A^T A, \quad \text{and} \quad \underline{d} = A^T \underline{b}.$$

[2 MARKS]

Show that the vector $\underline{x}$ which minimises this quadratic satisfies the normal equations

$$C\underline{x} = \underline{d}\ .$$

[4 MARKS]

(ii) Suppose that the $m \times n$ matrix A in part (i), $m > n$, has a QR factorization

$$A = QR \quad \text{where} \quad R = \begin{pmatrix} R_1 \\ O \end{pmatrix}$$

with Q being an $m \times m$ orthogonal matrix and with R and R_1 being upper triangular matrices of size $m \times n$ and $n \times n$ respectively. Also let

$$Q^T \underline{b} = \begin{pmatrix} \underline{c} \\ \underline{z} \end{pmatrix} \quad \text{where} \quad \underline{c} \in \mathbb{R}^n,\ \underline{z} \in \mathbb{R}^{m-n}\ .$$

Show that the vector $\underline{x}$ satisfying the normal equations also satisfies

$$R_1 \underline{x} = \underline{c}.$$

[4 MARKS]

(iii) By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} -1 & -1 & 1 \\ -1 & -1 & 2 \\ 1 & 3 & 2 \\ 1 & 3 & 3 \end{pmatrix} \quad \text{and} \quad \underline{b} = \begin{pmatrix} 1 \\ 3 \\ 3 \\ -3 \end{pmatrix}\ .$$

[10 MARKS]

---

2. The following was question 3 of the 2000 paper.

(i) If A is an $m \times n$ real matrix with $m > n$ and $\underline{b}$ is an $m \times 1$ real vector then show that the vector $\underline{x}$ which minimises

$$\phi(\underline{x}) = (\|A\underline{x} - \underline{b}\|_2)^2$$

satisfies the normal equations

$$A^T A\underline{x} = A^T\underline{b}\ .$$

[6 MARKS]

Also show that for any other vector $\underline{x} + \underline{d}$ we have

$$\phi(\underline{x} + \underline{d}) - \phi(\underline{x}) = (\|A\underline{d}\|_2)^2$$

and hence explain why $\underline{x}$ is the unique minimum point of ϕ when the n columns of A are linearly independent.

[4 MARKS]

(ii) By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} -1 & -1 & -1 \\ 1 & 2 & 3 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{pmatrix} \quad \text{and} \quad \underline{b} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ -2 \end{pmatrix} .$$

[10 MARKS]

---

3. The following were part of question 3 of the 1999 paper.

(i) Let A be a real $m \times n$ matrix. If the n columns of A are linearly independent then explain why the matrix $A^T A$ is positive definite.

(ii) By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} 1 & 1 & 3 \\ -1 & 2 & -1 \\ 1 & 2 & 2 \\ 1 & 1 & 2 \end{pmatrix} \quad \text{and} \quad \underline{b} = \begin{pmatrix} 1 \\ 3 \\ 3 \\ -1 \end{pmatrix} .$$

---

4. The following was question 3 of the 1996 paper.

(i) Let $A = QR$ denote a QR factorization of A where Q is an $m \times m$ orthogonal matrix and R is an $m \times n$ matrix with the partitioned form

$$R = \begin{pmatrix} R_1 \\ O \end{pmatrix} ,$$

where R_1 is an $n \times n$ upper triangular matrix with positive diagonal entries and O is the $(m-n) \times n$ zero matrix. Explain how this factorization is related to the Cholesky factorization of $A^T A$.

(ii) By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} 4 & 0 & 0 \\ 2 & 2 & 1 \\ 2 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \underline{b} = \begin{pmatrix} -1 \\ 0 \\ 2 \\ 0 \end{pmatrix}.$$

---

5. The following were parts of questions from the 1998 and 1997 papers.

(i) By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} -2 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & -1 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 2 \end{pmatrix} \quad \text{and} \quad \underline{b} = \begin{pmatrix} -1 \\ -1 \\ 2 \\ 1 \\ 5 \end{pmatrix}.$$

(ii) By using the Cholesky method of factorization solve the normal equations in the case that

$$A = \begin{pmatrix} 1 & 1 & 2 \\ -1 & 2 & 0 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \underline{b} = \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \end{pmatrix}.$$

---

6. Let A be an $n \times n$ real matrix and suppose $\underline{v} \in \mathbb{R}^n$, $\underline{v} \neq \underline{0}$. What is the normal equations associated with the least squares problem of minimising $\|A\underline{v} - \lambda\underline{v}\|_2$ with respect to λ (i.e. a least squares problem with one unknown) and give the value of λ which minimises $\|A\underline{v} - \lambda\underline{v}\|_2$.

---

7. Let A be an $n \times n$ real symmetric matrix and for vectors $\underline{v} \in \mathbb{R}^n$ which are not zero let

$$\phi(\underline{v}) = \frac{\underline{v}^T A \underline{v}}{\underline{v}^T \underline{v}}.$$

By considering the partial derivatives of ϕ with respect to each component of $\underline{v}$ show that the local turning points of ϕ occur at eigenvectors of A.

EXERCISES ON ITERATIONS AND THE EIGENVALUE PROBLEM

1. This was question 4 of the 2001 paper.

   (i) Let $\underline{v} \neq \underline{0}$ be an eigenvector of an $n \times n$ matrix A with eigenvalue λ. Show that

   $$|\lambda - a_{kk}| \leq \sum_{\substack{j=1 \\ j\neq k}}^{n} |a_{kj}|$$

   for at least one value of k, $1 \leq k \leq n$, i.e. show that λ is in at least one of the Gerschgorin disks.

   [6 MARKS]

   What can you say about the eigenvalues when one of the Gerschgorin disks is isolated from the other $n-1$ disks?

   [1 MARK]

   (ii) Define the Rayleigh quotient $\mu_R(\underline{v})$ of an $n \times n$ symmetrix matrix A with an $n \times 1$ vector $\underline{v} \neq \underline{0}$.

   [1 MARK]

   Let

   $$\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_1$$

   denote the eigenvalues of a symmetric matrix A. By first expressing $\underline{v}$ in terms of the orthonormal eigenvectors of A show that the Rayleigh quotient satisfies the inequality

   $$\lambda_n \leq \mu_R(\underline{v}) \leq \lambda_1 \ .$$

   [4 MARKS]

   (iii) Let

   $$A = \begin{pmatrix} 1 & 0.3 & 0 & 0 \\ 0.3 & 4 & 0.01 & 0 \\ 0 & 0.01 & 6 & 1 \\ 0 & 0 & 1 & 6 \end{pmatrix}$$

   and let λ_s denote the smallest eigenvalue.

   (a) By using Gerschgorin's theorem and Rayleigh quotients explain why

   $$0.7 \leq \lambda_s < 1.$$

   [2 MARKS]

   (b) By considering a similar matrix $D^{-1}AD$ where D is a diagonal matrix of the form $D = \text{diag}\{d, 1, 1, 1\}$, $d \neq 0$, show that the smallest eigenvalue satisfies

   $$0.97 \leq \lambda_s.$$

   [3 MARKS]

(c) By considering the Rayleigh quotient of A with the vector

$$\underline{v} = (1, -0.1, 0, 0)^T,$$

or otherwise, show that the smallest eigenvalue satisfies

$$\lambda_s < \frac{0.98}{1.01}\ .$$

[3 MARKS]

---

2. This was question 5(i) of the 2001 paper.

   Let $\underline{v}_1, \cdots, \underline{v}_n$ be linearly independent eigenvectors of an $n \times n$ matrix A with eigenvalues $\lambda_1, \cdots, \lambda_n$ respectively where

$$|\lambda_n| \leq \cdots \leq |\lambda_2| < \lambda_1,$$

   i.e. with a single largest eigenvalue in magnitude which is positive. Also let a vector $\underline{x}^{(0)}$ be of the form

$$\underline{x}^{(0)} = \sum_{i=1}^{n} c_i \underline{v}_i$$

   with $c_1 > 0$.

   (a) Show that

$$\frac{1}{\lambda_1^k} A^k \underline{x}^{(0)} \to c_1 \underline{v}_1 \quad \text{as } k \to \infty.$$

[3 MARKS]

   (b) Show that if

$$\underline{x}^{(k)} = \frac{1}{\|A^k \underline{x}^{(0)}\|_2} A^k \underline{x}^{(0)} \quad \text{for } k = 1, 2, \cdots$$

   then

$$\underline{x}^{(k)} \to \frac{1}{\|\underline{v}_1\|_2} \underline{v}_1 \quad \text{and} \quad {\underline{x}^{(k)}}^T A \underline{x}^{(k)} \to \lambda_1 \quad \text{as } k \to \infty.$$

[7 MARKS]

---

3. This was question 5(iii) of the 2000 paper.

   Let

$$A = \begin{pmatrix} 10 & 1 & 0 & 0 \\ 1 & 10 & 1 & 0 \\ 0 & 1 & 10 & 1 \\ 0 & 0 & 1 & 20 \end{pmatrix}.$$

   (a) By using symmetry and Gerschgorin's theorem what can you deduce about the largest eigenvalue λ_1 of A.

[2 MARKS]

(b) By considering matrices of the form $D^{-1}AD$ where $D = \mathrm{diag}\{1,1,1,d\}$, $d \neq 0$, show that the largest eigenvalue λ_1 satisfies

$$\lambda_1 \leq 20 + \frac{1}{9}\ .$$

[4 MARKS]

(c) By considering the Rayleigh quotient bounds and the vector

$$\underline{v} = (0, 0, 0.1, 1)^T$$

show that the largest eigenvalue λ_1 satisfies

$$20 + \frac{1}{10.1} < \lambda_1\ .$$

[4 MARKS]

---

4. This was question 5(iii) of the 1999 paper.

Let A denote the symmetric matrix

$$A = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.1 & 2 & 0.01 \\ 0 & 0.01 & 3 \end{pmatrix}$$

and let λ_s denote the smallest eigenvalue of A.

(a) What can be deduced about the location of λ_s by using Gerschgorin's theorem?

(b) By considering a matrix of the form $D^{-1}AD$ where $D = \mathrm{diag}\{d, 1, 1\}$ show that the smallest eigenvalue λ_s satisfies

$$\lambda_s \geq 0.99\ .$$

(c) By considering the Rayleigh quotient of A with the vector

$$\underline{x} = \begin{pmatrix} 1 \\ -0.1 \\ 0 \end{pmatrix}$$

show that

$$\lambda_s < \frac{1}{1.01}\ .$$

---

5. The following was part of question 5 of the 1996 MA251S paper.

Define the Rayleigh quotient $\mu_R(\underline{v})$ of an $n \times n$ matrix A corresponding to an $n \times 1$ vector $\underline{v} \neq \underline{0}$. If A is a real symmetric matrix with eigenvalues $\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_2 \leq \lambda_1$ then it can be shown that

$$\lambda_n \leq \mu_R(\underline{v}) \leq \lambda_1$$

for all vectors $\underline{v}$. By using this result with a suitably chosen vector $\underline{v}$ and by using Gerschgorin's theorem show that for the $n \times n$ tri-diagonal matrix A given by

$$A = \begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix}$$

the largest eigenvalue λ_1 satisfies

$$6 - \frac{2}{n} < \lambda_1 \le 6 \ .$$

---

6. By suitable choice of the vector $\underline{v}$ and by using Gerschgorin's theorem show that the smallest eigenvalue λ_n of the matrix A in question 5 satisfies

$$2 \le \lambda_n < 2 + \frac{2}{n} \ .$$

---

7. The following was part of part (i) of question 6 of the 1992 MA351S paper.

   Show that if A is the 5×5 tri-diagonal matrix

$$A = \begin{pmatrix} \alpha+\beta & \beta & & & \\ \beta & \alpha & \beta & & \\ & \beta & \alpha & \beta & \\ & & \beta & \alpha & \beta+\epsilon \\ & & & \beta+\epsilon & \alpha+\beta \end{pmatrix}$$

   where α, β and ϵ are positive constants, then

$$\alpha + 2\beta + 0.4\epsilon \le \lambda_1 \le \alpha + 2\beta + \epsilon \ .$$

---

8. The following was question 5(ii) of the 1994 MA351 paper.

   Let

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 0 & 2 & 5 & \epsilon \\ 0 & 0 & \epsilon & 9 \end{pmatrix}$$

   with $-1 \le \epsilon \le 1$ and let $\mu_R(\underline{v})$ denote the Rayleigh quotient corresponding to a 4×1 vector $\underline{v}$. Using the Gerschgorin result applied to a matrix of the form $D^{-1}AD$, where D is a diagonal matrix of the form diag(1,1,1,d), and using the result that $\mu_R(\underline{v}) \le \rho(A)$ for all vectors $\underline{v}$, where $\rho(A)$ denotes the spectral radius of A, show that

$$9 \le \rho(A) \le 9 + \frac{\epsilon^2}{2} \ .$$

---

9. The following was question 5(ii) of the 1998 MA2051S paper.

Let $\underline{v}_1, \cdots, \underline{v}_n$ be eigenvectors of an $n \times n$ matrix A with eigenvalues $\lambda_1, \cdots, \lambda_n$ respectively where

$$|\lambda_n| \le \cdots \le |\lambda_2| < |\lambda_1|,$$

i.e. with a single largest eigenvalue in magnitude. Also let a vector $\underline{x}^{(0)}$ be of the form

$$\underline{x}^{(0)} = \sum_{i=1}^{n} c_i \underline{v}_i$$

with $c_1 \ne 0$. If the vectors $\underline{x}^{(k)}$ and $\underline{y}^{(k)}$ are defined by

$$\underline{y}^{(k)} = \frac{1}{\lambda_1^k} A^k \underline{x}^{(0)} \quad \text{and} \quad \underline{x}^{(k)} = \frac{1}{\|A^k \underline{x}^{(0)}\|_2} A^k \underline{x}^{(0)}$$

then show that

$$\frac{\underline{y}^{(k)^T} A \underline{y}^{(k)}}{\underline{y}^{(k)^T} \underline{y}^{(k)}} \to \lambda_1 \quad \text{and} \quad \frac{\underline{x}^{(k)^T} A \underline{x}^{(k)}}{\underline{x}^{(k)^T} \underline{x}^{(k)}} \to \lambda_1 \ .$$

---

10. This was question 4(ii) of the 2000 paper.

Given an $n \times 1$ vector $\underline{x}^{(0)}$ with $\|\underline{x}^{(0)}\|_2 = 1$ and given an $n \times n$ matrix A, the vectors $\underline{x}^{(k)}$ are defined by

$$\underline{x}^{(k+1)} = A\underline{x}^{(k)} / \|A\underline{x}^{(k)}\|_2$$

for $k = 0, 1, 2, \cdots$. In the case when A is symmetric with eigenvalues $\lambda_1, \cdots, \lambda_n$ and with associated orthonormal eigenvectors $\underline{v}_1, \cdots, \underline{v}_n$ respectively, show that when

$$\underline{x}^{(0)} = \sum_{i=1}^{n} c_i \underline{v}_i,$$

we have

$$\underline{x}^{(k)^T} A \underline{x}^{(k)} = \frac{\sum_{i=1}^{n} c_i^2 \lambda_i^{2k+1}}{\sum_{i=1}^{n} c_i^2 \lambda_i^{2k}} \ . \qquad \textbf{[6 MARKS]}$$

Hence show that if the eigenvalues are such that $|\lambda_n| \le \cdots \le |\lambda_2| < |\lambda_1|$ and $c_1 \ne 0$ then

$$\underline{x}^{(k)^T} A \underline{x}^{(k)} \to \lambda_1 \quad \text{as } k \to \infty. \qquad \textbf{[3 MARKS]}$$

MATLAB at Brunel

As MATLAB implements much of what is done in this module, and much more as well, I include here some notes to remind you about how to use it at Brunel.

The first point to note is that MATLAB is available on many different computer platforms. If you have you own PC then there is a student version (version 4.0) which runs under MicroSoft Windows 3.11. and there is a student version (version 5.0) which runs under MicroSoft Windows 95. You get these with the appropriate books. The version that we have at Brunel on the UNIX network is version 5.2 for computers running the Solaris operating system. To use this you need to connect to a machine running this operating system. There are two ways to do this. You can logon directly on a workstation (e.g. in room M031) or you can logon to a PC running Windows NT and then connect to a computer running Solaris by using Exceed-NT by doing the following:

1. Click on the Start icon (bottom left of the screen).
2. Select Network packages.
3. Select the X-Windows, Exceed-NT option.

When you have connected you will be in the openwindows environment with the precise windows layout depending on your files `.xinitrc` and `.twmrc` in your home directory. If you have never changed these files then you will have a default set-up. The default set-up is NOT recommended for running MATLAB without some modification as many of the important keys are disabled if you start MATLAB from the standard cmdtools window. Instead you need to run MATLAB from within an `xterm`. To make this relatively easy you should first type

```
~icstmkw/bin/ma2051s_set
source ~/.cshrc
```

The commands will make a small adjustment to your set up which will make it easier for me to assist you if you need help during the sessions. You will not need to type these again. You can then start MATLAB in a suitable `xterm` by simply typing

```
matlab_
```

(This command sets the `xterm` and runs the commands

```
use matlab
matlab
```

which you would otherwise have to type if you just create a `xterm` of your own.)

Typing `quit` or `exit` leaves MATLAB.

Some General Points about MATLAB Version 5.2

Before you start using MATLAB there are a few points that should be noted.

1. All quantities are usually considered as real matrices of appropriate dimensions. By using the reserved variables `i` or `j` (for $\sqrt{-1}$) complex matrices can be defined. Any operation which gives a complex result also introduces complex matrices.
2. Arithmetic is accurate to about Fortran `REAL*8` precision, i.e. about 16 significant figures, although the default format for output is to show only 4 significant figures.
3. Output is suppressed in commands by terminating the statement with a semicolon(;).
4. Output is enabled when an m-file is used by including the statement

```
echo on
```

This is useful when you start in locating errors. There are also a set of debugging commands to assist in locating and correcting errors. Type `help debug` to obtain further information.

5. The output of one command will often be more than a screen full. You should use the command

   ```
   more on
   ```

   to stop the output every standard screen full or type

   ```
   more(15)
   ```

   to stop the output every 15 lines. In the case of errors in a script file you can insert the command

   ```
   keyboard
   ```

   at a point in the file just before you you believe the error occurs. This then enables you to interactively test all of the quantities which are set at that point.

6. MATLAB names are case sensitive by default. All MATLAB commands are in lower case and must begin with a letter of the alphabet (`a-z`). If you create your own commands then they must begin with a letter in the range (`a-z` or `A-Z`).

7. No array or type declarations are required. Vectors and matrices become defined or possibly redefined by their usage. Also when functions are used MATLAB scans your command to determine the number of input arguments you are using and the number of output arguments you are expecting and it returns information appropriately whenever possible. That is you need not be consistent in your referencing to functions. For example if `a` is a 3×3 matrix then

   ```
   b=eig(a)
   ```

   gives a 3×1 column vector `b` consisting of the eigenvalues of `a` . However the statement

   ```
   [x,d]=eig(a)
   ```

   returns a 3×3 matrices `x` and `d` . `d` is a diagonal matrix containing the eigenvalues and the `i`*th* column of `x` gives the eigenvector corresponding to the eigenvalue `d(i,i)`.

8. All characters which occur between a `%` character and the end of line are treated as comments (provided the `%` character is not part of a character string).

9. The operations `+`, `-`, `*`, `/`, `^`, `'`, `\` are usually interpreted as matrix operations. Commands of the form `.+`, `.-`, `.*`, `./`, `.^`, `.\` are entry-wise operations.

10. If you are creating a MATLAB program which is 'reasonably long' then it is probably best to create your own m-file containing the MATLAB instructions. If `zyxw.m` is the m-file then you execute the commands by typing

    ```
    zyxw
    ```

    at the MATLAB prompt.

11. A new feature introduced since the introduction of version 5.0 (i.e. it was not available in version 4.2) is that an m-file can contain more than one matlab function. Thus if you have scripts which reference you own matlab functions then you can now put all the commands in one file. These functions will not however be 'visible' to other MATLAB programs.

12. If when using MATLAB you obtain the response

    ```
    Undefined function or variable
    ```

    then there are several possible reasons. You may have mis-typed the command and thus what you actually typed is not a variable or the name of any known m-file to MATLAB. Type `who` to get a list of the variables that you are currently using. If you are sure that you are using the correct name then you should check your path variables as it may be that the m-file exists but it is not in the `PATH`. You should also check which version of MATLAB you are using (type `ver` to check). You may be trying to use a command which is only available in the more up to date UNIX version.

13. To find out help about a command you should type

    ```
    help command_name
    ```

    where `command_name` is a builtin MATLAB command and it corresponds to an m-file with the name `command_name.m` which is in the current directory or in a directory searched by MATLAB. For example you can type

    ```
    help eig
    ```

    If you do not know the precise name of a command but you know some information such as it has something to do with eigenvalues then you can type

    ```
    lookfor eigenvalues
    ```

    This will list all the m-files whose first comment line contains the word eigenvalues.

14. In version 5.2 of MATLAB on our UNIX network, MATLAB will search for commands in the current working directory and in the directories specified by

    ```
    $MATLABPATH
    ```

    where `$MATLABPATH` refers to the environmental variable `MATLABPATH` which contains the location of the system MATLAB directories. This is set up for you when you start `matlab`. You can type `setenv` to list your environmental variables and you can add to `MATLABPATH` by, for example, the command

    ```
    setenv MATLABPATH = ($USER/2051s/matlab $MATLABPATH)
    ```

    to add the subdirectory `2051s/matlab` to the search path. With the default set-up you actually already get `$USER/matlab` in your path and thus this may not be necessary.

15. If you want a hard copy version of your session or part of your session then you can do this by typing

    ```
    diary sess014
    ```

    All text information that appears on the screen is also saved in the file called `sess014`. The command

    ```
    diary off
    ```

    turns off the diary mechanism.

16. With many packages there are a FAQ's (Frequently Asked Questions) information which is often a good place to look when you have problems with a package. The latest version of these is available on the web which you can access with the URL

    ```
    http://www.mathworks.com/support/faq/faq.shtml
    ```

Some MATLAB commands

: colon

`1:5` creates the row vector `[1 2 3 4 5]`. If `a` and `b` are in MATLAB a 10×10 matrix and a 10×1 column vector then

```
b5=b(1:5)
```

creates a column vector of length 5 consisting of the first 5 entries of `b` and

```
a5=a(1:5,1:5)
```

creates the 5×5 leading principal sub-matrix of `a`.

`for` and `end`

Used for creating loops. For example

```
for k=1:100
.......some matlab instructions
end
```

means do the instructions in the loop for `k`=1,2,$\cdots$ 100.

`if` and `break` and `end`

Used for making decisions and terminating loops early respectively. For example

```
for k=1:100
.......some matlab instructions
if(error<1e-10) break; end
end
```

means do the instructions in the loop for `k`=1, 2,$\cdots$ 100 but leave the loop if the variable `error` becomes less than 10^{-10}.

\ backslash

If `a` is a square `n`×`n` matrix and `b` is a `n`×1 column vector then the statement

```
x=a\b;
```

creates the `n`×1 column vector `x` which is the solution of the linear system, in MATLAB notation, `a*x=b`. If `a` is in the default full form then, up to rounding error, this will usually be the same as `inv(a)*b`. Note that it is much more efficient to use \ to solve the system than it is to use `inv`. This should only be used as a check when you are learning about MATLAB.

If `a` is stored in sparse form then the solution algorithm exploits this, if possible, to more efficiently solve the system. You cannot compare it with `inv(a)*b` in this case as `inv` does not work with sparsely stored matrices as it would generally give a full matrix.

If `a` has more rows than columns then `x=a\b;` gives the least squares solution.

`eig`

`eig(a)` computes the eigenvalues of `a` where `a` is a square matrix.

```
[x,d]=eig(a);
```

computes the eigenvectors `x` and eigenvalues (as a diagonal matrix) such that `a*x=x*d`.

`norm`

Computes a norm of a vector or a matrix. See page 3.10.

`cond`

`cond(a)` computes the spectral condition number of a matrix. See page 3.5.

`lu`, `qr`, and `chol`

Compute an LU type factorization, a QR type factorization and a Cholesky LL^T factorization respectively. Type `help lu`, or `help qr` or `help chol` for more details.

`inv,det,trace,rank`

For a square matrix `a`, `inv(a)` and `det(a)` compute respectively the inverse and determinant of `a`. `trace(a)` and `rank(a)` compute respectively the trace and rank of `a` for any size of matrix.

`mean,std`

For a vector `x`, `mean(x)` and `std(x)` compute respectively the mean and standard deviation of `x`.