MA519: Compuational Mathematics - Numerical Linear Algebra

Niall Madden

April 17, 2014

Abstract

These notes are for *Numerical Linear Algebra*, taught at NUI Galway in Semester 2, 2013/2014. They are largely based on material from textbooks [Ips09], [TB97], [Dem97] and [Saa03]. These notes are mainly intended for students at NUI Galway and UCD, and compliment the lecture notes. If you are interested in the topics covered, you should consult the primary references listed above.

If you have any comments, or spot any typos, please contact Niall Madden (`Niall.Madden@NUIGalway.ie`), or see the course website at http://www.maths.nuigalway.ie/~niall/MA519

Contents

1 MA519 : Introduction

This class consists of an overview of the course. The real mathematics starts in the next lecture.

Module materials, including summaries of many lectures, will be available from
http://www.maths.nuigalway.ie/~niall/MA519

These notes are *not* formal, and are unlikely to make much sense if you are not participating in classes.

1.1 Course content

We are concerned with the construction of computer algorithms for solving problems in linear algebra.

Two main types of problem are of interest:

(i) Solving linear systems ($Ax = b$).

(ii) Solving Eigenvalue problems ($Ax = \lambda x$).

Here A is a matrix: usually it is both real and square (most of what we'll study also applies directly to matrices with complex-entries).

Our primary concern is with *sparse matrices*: these are matrices which have comparatively few non-zero entries. The motivation we'll take will come from such matrices as the arise in numerical solution of differential equations, and in analysis of networks.

Starting fundamental ideas of orthogonality, norms and the Singular Value Decomposition, we introduce QR factorisation, and the notion of the condition number of a matrix.

The main ideas we'll study after that are (though not necessarily in this order)

(i) The direct solution of linear systems using Gaussian Elimination, and its variants, but par (e.g., Cholesky Factorisation, Thomas Algorithm).

(ii) Iterative methods for solving linear systems. Although we'll look at classical techniques, such as Jacobi iteration and SOR, our goal will be to understand more advanced techniques, like conjugate gradients.

(iii) Estimation of eigenvalues and eigenvectors.

We'll also look at the implementation of the methods in (say) Matlab.

1.2 Mathematical Preliminaries

You'll need a solid grounding in linear algebra (so you know the fundamentals of manipulating vectors and matrices), as well as standard ideas in algebra: the fundamental theorem of algebra, linear independence, vectors spaces, orthogonality...

A great place to start is the read (and understand and do all the exercises) in Chapter 1 of Ipsen's *Numerical Matrix Analysis: Linear Systems and Least Squares.*

1.3 Solving linear systems

Problems are of the form: *find the set of (real) numbers* $x_1, x_2, \ldots, x_n$ *such that*

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_n \end{aligned}$$

where the a_{ii} *and* b_i *are real numbers.*

It is natural to rephrase this using the language of linear algebra: *Find the vector* $\mathbf{x} = (x_1, x_2, \ldots x_n)^T$ *such that*

$$A\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where A *is an* $m \times n$ *matrix, and* $\mathbf{b}$ *is a (column) vector with* n *entries.*

In this section, we'll try to find clever ways of solving this system, both directly and iterative.

We are primarily interested in the case where $m = n$ and the matrix A is invertible. But we'll also consider what happens if, say, $m > n$ (over determined system) or $m < n$ (under determined).

1.4 The eigenvalue problem

Our other main problem is, given a square matrix A find a scalar λ and nonzero vector $\mathbf{x}$ such that $A\mathbf{x} = \lambda\mathbf{x}$.

In some case, we will want to find *all* the eigenvalues of a matrix. In other cases, we'll just want estimates for the largest and/or smallest eigenvalues.

And sometimes we'll want to compute the eigenvector associated with a certain known eigenvalue (e.g., if A is a stochastic matrix).

1.5 Other considerations

As well as studying/designing algorithms for solving particular problems, we'll consider

- When does the problem have a solution?
- How *stable* is the computation? That is, if there are errors in the data (maybe due to round-off error) are these magnified?
- How expensive is that algorithm? How much computing time or memory resources are required?

2 Matrix multiplication

This lecture follows very closely Lecture 1 of Trefethen and Bau ([TB97]), *Numerical Linear Algebra*. The key idea is to think of matrix-vector multiplication as constructing a linear combinations of the columns of the matrix.

The first 5 chapters of this book are freely available from the author's homepage:
http://www.comlab.ox.ac.uk/nick.trefethen/text.html

The summary below should be read in the context of those notes. I don't reproduce their text, just repeat the examples done in class.

2.1 Matrix vector multiplication

If $A \in \mathbb{C}^{m\times n}$ and $x \in C^n$ and b is the m-vector given by $b = Ax$, then

$$b_i = \sum_{j=1}^{N} a_{ij}x_j.$$

Example 2.1.

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

This gives

$$b = \begin{pmatrix} x_1 a + x_2 b \\ x_1 c + x_2 d \end{pmatrix} = x_1 \begin{pmatrix} a \\ c \end{pmatrix} x_2 \begin{pmatrix} b \\ d \end{pmatrix}.$$

So b is a linear combination of the columns of A with coefficients from x.

2.2 Matrix-matrix multiplication

If $A \in C^{m\times n}$, $C \in \mathbb{C}^{n\times p}$ and $B = AC$, then B is the $m \times p$ matrix given by

$$b_{ij} = \sum_{k=1}^{n} a_{ik}c_{kj}.$$

But in keeping with the ideas above, let us consider the formula for column j of B:

$$b_j = \sum_{k=1}^{n} c_{kj}a_k.$$

So column j of B is a linear combination of all the columns of A, with the coefficients taken from column j of C.

Other examples that are worth considering, include computing the inner and outer products of the vectors $(a, b, c)^T$ and $(d, e, f)^T$.

2.3 Involutory, projector, and nilpotent matrices

A particularly important case of computing computing matrix products, is computing matrix powers.

For a square matrix, A, we say that

- A is *involutory*, if $A^2 = I$.
- A is a *projector* (or is *indepotent*), if $A^2 = A$.
- A is *nilpotent*, if $A^k = 0$ for some $k > 0$

Example 2.2. Let

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

Write down A^2, A^3 and A^4.

2.4 Triangular matrices

Sometimes the structure of a matrix is preserved under matrix multiplication. An obvious example is that the product of two diagonal matrices is diagonal. A more interesting case is:

Example 2.3. Show that the product of two lower triangular matrices is itself lower triangular.

Note: this last example will crop up again. We will see that the processes of Gaussian Elimination involves repeatedly multiplying a matrix by lower triangular matrices in order to yield an upper triangular matrix. This will lead to the idea of factorising a matrix as the produce of lower and upper triangular matrices.

2.5 Exercises

Do Exercises (i)–(viii) in Section 1.7 of Ipsen's *Numerical Matrix Analysis: Linear Systems and Least Squares.*

3 Matrix inverse

3.1 Range and Rank

The *range* of a matrix is the set of vectors that can be expressed as Ax for some x. We now understand that this means that the range of a matrix is the same as the space spanned by its columns, which we call the *column space*. (The *row space* is the space spanned by the (row) vectors that for its rows).

The nullspace of a matrix is set of vectors $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{0}$. Since we think of matrix-vector multiplication as forming a linear combination of the columns of A, if the columns of A are linearly independent then the null space contains only the zero vector.

The column (row) rank of A is the dimension of the column (row) space. A slightly surprising fact that the row rank and column rank of a matrix are the same. We'll leave the proof until after we know what the SVD is. But in the mean time, we'll note that if the rank of $A \in C^{m\times n}$ is $\min(m, n)$ then it has *full rank*.

This will lead us to looking the idea of invertible matrices.

3.2 Matrix Inverses

One can consider the idea of solving the linear system $Ax = b$ as a *change of basis operation* (See [TB97, Lecture 1]).

We then went on to think about inner products and orthogonality.

. . . . . .

Recall that the rank of the matrix is the number of linearly independent rows/columns that it contains.

If $A \in \mathbb{R}^{n\times n}$ has full rank, then any vector $\mathbf{b} \in \mathbb{R}^n$ can be expressed as a linear combination of the columns of A. Or, to put it another way, the problem $A\mathbf{x} = \mathbf{b}$ has a solution. Or yet another way: there exists a matrix $A^{-1} \in \mathbb{R}^{n\times n}$ such that $\mathbf{x} = A^{-1}\mathbf{b}$.

Denoting as usual $I = (e_1|e_2|\cdots|e_n)$, given a matrix $A \in \mathbb{R}^{n\times n}$, if there exists $Z \in \mathbb{R}^{n\times n}$ such that $AZ = ZA = I$, then we say that A is *invertible* or *nonsingular*. And we write $Z = A^{-1}$.

There are several other statements that are equivalent to saying that $A \in \mathbb{C}^{m\times m}$ is invertible. These include

- $\text{range}(A) = \mathbb{R}^n$.
- $\text{null}(A) = \{\mathbf{0}\}$.
- The set of eigenvalues of A does not include zero.
- The set of singular values of A does not include zero.
- $\det(A) \neq 0$.
- ...

3.3 Solving linear equations

When trying to solve $A\mathbf{x} = \mathbf{b}$, we never, ever (well, hardly ever) first try to find A^{-1} so that we can compute $\mathbf{x} = A^{-1}\mathbf{b}$.

One of the key approaches is to find matrices L and U which are (respectively) unit lower and upper triangular matrices such that $A = LU$. We'll cover this in lectures soon (though we'll emphasise a special variant for symmetric positive definite matrices: Cholesky Factorisation).

For now we'll think of solving a linear system as a change of basis operation: we are looking for the expansion of b in the column of A.

This is not to say that there aren't many cases where we are interested finding A^{-1}. But usually only in cases where this is easy, for example if A is *unitary*...

4 Norms (Part 1)

(Largely based on [TB97, Lecture 2]).

4.1 Transpose and Conjugate Transpose

If $A \in \mathbb{R}^{m\times n}$, then its transpose, $B = A^T \in \mathbb{R}^{n\times m}$, is such that $b_{ij} = a_{ji}$. If $A \in \mathbb{C}^{m\times n}$, then its "*conjugate transpose*" a.k.a. "*hermitian conjugate*" is $B = A^\star \in \mathbb{C}^{n\times m}$ such that $b_{ij} = \bar{a}_{ji}$, where $\bar{x}$ denotes the usual complex conjugate.

Note that $(AB)^\star = A^\star B^\star$. Also, if $A \in \mathbb{R}^{m\times n}$, then $A^T = A^\star$.

If $A \in \mathbb{C}^{m\times m}$ is such that $A = A^*$, then we say that A is *Hermitian*. If $A \in \mathbb{R}^{m\times m}$ and $A = A^T$, we say that A is symmetric.

4.2 Inner products

Recall...

$$(x, y) = x^T y = \sum_{i=1}^{m} x_i y_i \text{ for } x, y \in \mathbb{R}^n,$$

$$(x, y) = x^* y = \sum_{i=1}^{m} \bar{x}_i y_i \text{ for } \text{ for } x, y \in \mathbb{C}^n,$$

where $x^* = \bar{x}^T$ is the "*conjugate transpose*", some times called the "*hermitian conjugate*". There are other Inner Products, of course....

We say x and y are *orthogonal* if $x^\star y = 0$, but we'll postpone most of the discussion on this until after we've introduced the idea of a *norm*.

4.3 Norms

Much of this section is done in the context of vectors of real numbers. As an exercise you'll determine the corresponding results over $\mathbb{C}^n$.

Definition 4.1. Let $\mathbb{R}^n$ be all the vectors of length n of real numbers. The function $\|\cdot\|$ is called a **norm** on $\mathbb{R}^n$ if, for all $x, y \in \mathbb{R}^n$

(i) $\|x\| = 0$ if and only if $x = 0$.

(ii) $\|\lambda x\| = |\lambda|\|x\|$ for any $\lambda \in \mathbb{R}$,

(iii) $\|x + y\| \leqslant \|x\| + \|y\|$ (triangle inequality).

The norm of a vector gives us some information about the "*size*" of the vector. But there are different ways of measuring the size: you could take the absolute value of the largest entry, you cold look at the "distance" for the origin, etc...

Definition 4.2. (i) The 1-*norm* is

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

(ii) The 2-*norm* (a.k.a. the *Euclidean norm*)

$$\|x\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}.$$

Note, if x is a vector in $\mathbb{R}^n$, then

$$x^T x = x_1^2 + x_2^2 + \cdots + x_n^2 = \|x\|_2^2.$$

(iii) The ∞-*norm* (also known as the max-*norm*)

$$\|x\|_\infty = \max_{i=1}^{n} |x_i|.$$

(iv) One can consider all the above to be an example of a p-*norm*

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

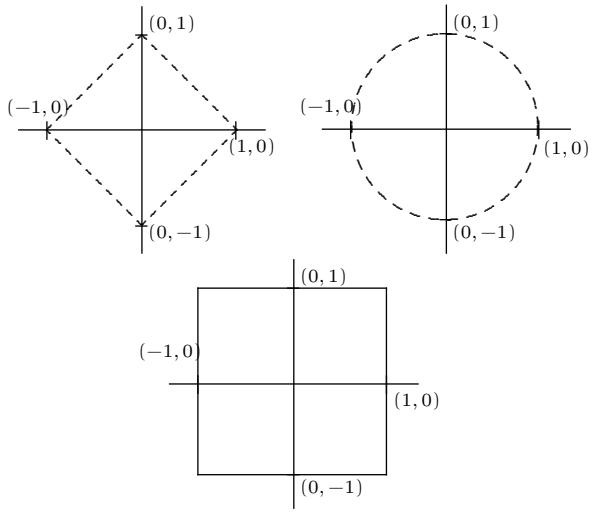


Figure 4.1: The unit vectors in $\mathbb{R}^2$: $\|x\|_1 = 1$, $\|x\|_2 = 1$, $\|x\|_\infty = 1$,

In Figure 4.1, the top left diagram shows the unit ball in $\mathbb{R}^2$ given by the 1-norm: the vectors $x = (x_1, x_2)$ in R^2 are such that $\|x\|_1 = |x_1| + |x_2| = 1$ are all found on the diamond. In the top right, the vectors have $\sqrt{x_1^2 + x_2^2} = 1$ and so are arranged in a circle. The bottom diagram gives the unit ball in $\|\cdot\|_\infty$ – the largest component of each vector is 1.

It takes a little bit of effort to show that $\|\cdot\|_2$ satisfies the triangle inequality. First we need

Lemma 4.3 (Cauchy-Schwarz)**.**

$$|x^T y| \leqslant \|x\|_2 \|y\|_2, \qquad \forall x, y \in \mathbb{R}^n.$$

Lemma 4.3 is actaully a special case of *Hölder's Inequality*:

$$|x^T y| \leqslant \|x\|_p \|y\|_q.$$

This can then be applied to show that

Lemma 4.4. *For all* $x, y \in \mathbb{R}^n$*,*

$$\|x + y\| \leqslant \|x\|_2 + \|y\|_2.$$

It follows directly that $\|\cdot\|_2$ is a norm.

5 Norms (Part 2)

5.1 Inner Products and norms

From Cauchy-Schwarz, it follows that

$$-1 \leqslant \frac{\mathbf{x}^{\mathsf{T}}\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2} \leqslant 1, \quad \text{for any } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

As we all know, this quantity can be thought of as the cosine of the angle between the vectors $\mathbf{x}$ and $\mathbf{y}$. So, for example, if $\mathbf{x}$ is *orthogonal* to $\mathbf{y}$, then $\mathbf{x}^{\mathsf{T}}\mathbf{y} = 0$.

5.2 Subordinate Matrix Norms

Definition 5.1. Given any norm $\|\cdot\|$ on $\mathbb{R}^n$, there is a *subordinate matrix norm* on $\mathbb{R}^{m\times n}$ defined by

$$\|A\| = \max_{\mathbf{x}\in\mathbb{R}^n_\star} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|, \tag{5.1}$$

where $A \in \mathbb{R}^{m\times n}$ and $\mathbb{R}^n_\star = \mathbb{R}^n/\{\mathbf{0}\}$.

The motivation for this definition is that we like to think of $A \in \mathbb{R}^{m\times n}$ as an *operator* or linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$. So rather than the norm giving us information about the "size" of the entries of a matrix, it tells us how much the matrix can change the size of a vector. However, this definition does not tell us *how* to compute a matrix norm in practice. That is what we'll look at next.

5.3 The 1-norm on $\mathbb{R}^{m\times n}$

Theorem 5.2.

$$\|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{i,j}|. \tag{5.2}$$

The proof is reasonably easy if we think of matrix-vector multiplication as forming a linear combination of the columns of A.

5.4 The max-norm on $\mathbb{R}^{n\times n}$

Theorem 5.3. *For any $A \in \mathbb{R}^{n\times n}$ the subordinate matrix norm associated with $\|\cdot\|_\infty$ on $\mathbb{R}^n$ can be computed by*

$$\|A\|_\infty = \max_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|.$$

That is, the ∞-norm of a matrix is just largest absolute-value row sum of the matrix

5.5 The 2-norm on $\mathbb{R}^{n\times n}$

It turns out that the 2-norm of a matrix is the square root of the largest eigenvalue of $B = A^{\mathsf{T}}A$. For this to make sense, we should note that all the eigenvalues of $A^{\mathsf{T}}A$ are real and be nonnegative. *Why?*

We postpone a detailed discussion until after we've met the *singular value decomposition.*

5.6 Consistency of matrix norms

It should be clear from (5.1) that, if $\|\cdot\|$ is a subordinate matrix norm, then

$$\|A\mathbf{u}\| \leqslant \|A\|\|\mathbf{u}\|,$$

for *any* $\mathbf{u} \in \mathbb{R}^n$. The analogous statement from the product of two matrices is:

Definition 5.4. A matrix norm $\|\cdot\|$ is *consistent* if

$$\|AB\| \leqslant \|A\|\|B\|, \qquad \text{for all } A, B \in \mathbb{R}^{n\times n}.$$

In some books, this is also called *sub-multiplicative.*

Theorem 5.5. *Any subordinate matrix norm is consistent.*

Not every matrix norm is norm is Consistent. See Exercise 5.8.

5.7 Some exercises on norms

Exercise 5.1. Given an norm on $\mathbb{R}^n$, denoted $\|\cdot\|$ we can define the *weighted norm* $\|\cdot\|_A$

$$\|x\|_A := \|Ax\|,$$

where A is an invertable matrix. Show that this is indeed a norm on $\mathbb{R}^n$.

If $\|\cdot\|$ is a matrix norm on $\mathbb{R}^{n\times n}$, is it true that

$$\|B\|_A := \|AB\|,$$

is a norm? Explain you answer.

Exercise 5.2. Show that for any of subordinate matrix norm, $\|I\| = 1$. (That is, the norm of the identity matrix is 1).

Exercise 5.3. Show that the eigenvalues of a hermitian matrix are purely real.

Exercise 5.4. In some books also give a definition of a norm that includes that $\|\mathbf{x}\| \geqslant 0$ for all $\mathbf{x}$. Should we add this to Definition 4.1, or can it be deduced?

Exercise 5.5. Show that $\|\cdot\|_1$ and $\|\cdot\|_\infty$ are norms.

Exercise 5.6. Prove Theorem 5.3.

Exercise 5.7. Prove Theorem 5.5.

Exercise 5.8. Suppose we define a different "max" norm of a matrix as follows:

$$\|A\|_{\widetilde{\infty}} = \max_{i,j} |a_{ij}|.$$

Show that $\|A\|_{\widetilde{\infty}}$ is a norm. Then find an example of a pair of matrices $A, B \in \mathbb{R}^{n\times n}$ such that $\|AB\|_{\widetilde{\infty}} > \|A\|_{\widetilde{\infty}}\|B\|_{\widetilde{\infty}}$.

6 Unitary matrices and matrix norms

6.1 Unitary Matrices

(See [TB97, Lecture 2].)

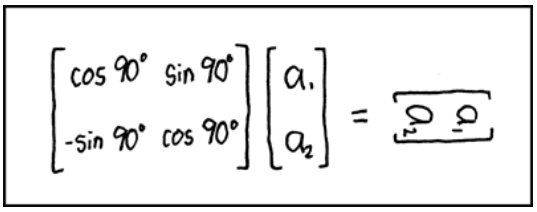
A square matrix $Q \in \mathbb{C}^{m\times m}$ is *unitary* if $Q^{-1} = Q^\star$. The columns of a unitary matrix form an *orthonormal basis* for $\mathbb{C}^m$.

Unitary matrices form an important part of many algorithms in numerical linear algebra, particularly ones for revealing eigenvalues.

They can be thought of as "change of basis operators". Suppose $Qx = b$. So if we express b as a linear combination of the columns of Q, then the coefficients are from x. Conversely, if we express x as a linear combination of the columns of $Q^\star$, then the coefficients are from b.

Other important facts include that transformation by a unitary matrix preserves the angles between vectors. It also preserves length of vectors in the 2-norm.

We'll look at some examples of unitary matrices, and then return move on to singular values.



6.2 General Matrix Norms

(See Trefethen and Bau, Lecture 3).

We'll now return to the idea of a matrix norm. We defined subordinate matrix norms in Definition 5.1. However, since the set of matrices $\mathbb{C}^{m\times n}$ forms a vector space, any function that maps such a matrix to a real number, and satisfies the conditions in Definition 4.1 is a (matrix) norm.

In Exercise 5.8, we saw an example of a silly matrix norm. A much more sensible, and important, example is the *Frobenius* norm:

$$\|A\|_F := \Big(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\Big)^{1/2}. \tag{6.1}$$

Some facts about $\|\cdot\|_F$.

- The *trace* of $A \in \mathbb{C}^{m\times m}$ is the sum of its diagonal entries:
$$\mathrm{tr}(A) = \sum_{k=1}^{m} a_{kk}.$$
This is used in the definition of the *trace inner product*: $(A, B)_{tr} = \mathrm{tr}(A^\star B)$. Moreover, $\|A\|_F = \sqrt{(A, A)_{tr}}$.
- The Frobenius norm is consistent. (See Exercise 6.2).
- If Q is unitary, then $\|QA\|_2 = \|A\|_2$ and $\|QA\|_F = \|A\|_F$. (See also Exercise 6.4).

6.3 Exercises

Exercise 6.1. From the examples of unitary matrices that we looked at in class it seems that, if Q is unitary, then

(i) $\det(Q) = 1$,

(ii) If λ is an eigenvalue of Q, then $|\lambda| = 1$.

Prove these statements, or give counter-examples to them.

Exercise 6.2. Show that the Frobenius norm is consistent. (Hint: using the definition in (6.1), and the Cauchy-Schwarz inequality).

Exercise 6.3. It is not hard to show that, if Q is unitary, then $\|Q\|_2 = 1$: just recall from our earlier discussion that $\|Qx\|_2 = \|x\|_2$. What can we say about $\|Q\|_F$?

Exercise 6.4. Suppose that Q is unitary.
Prove that $\|AQ\|_2 = \|A\|_2$.
Prove that $\|AQ\|_F = \|A\|_F$.
Is it true that $\|AQ\|_1 = \|A\|_1$?

7 The singular value decomposition

7.1 Singular Values

The singular values $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ of a matrix A can be defined in several ways:

- the σ_i^2 are the eigenvalues of $B = A^\star A$,
- σ_i is the length of the semiaxis of the hyperellipse that is the image of the unit circle under the linear transformation defined by A.
- Every matrix A has a factorisation $A = U\Sigma V^\star$ where U and V are unitary matrices, and Σ is a diagonal matrix. Then the enties of Σ are the singular values.

Please read Lecture 4 of Trefethen and Bau. T+B make the important observation in their introduction: *the image of the unit sphere under any* $m \times n$ *matrix is a hyperellipse.*

Suppose that $\{\|u_i\|\}_{i=1}^m$ is a set of orthonormal vectors, such that principal semiaxes of the hyperellipse are $\{\sigma_i u_i\}$.

These scalars $\sigma_1, \sigma_2, \ldots, \sigma_m$ are the *singular values of* A, and encapsulate many important properties of A.

7.2 Singular Value Decomposition

This exposition follows Lecture 4 of Trefethen and Bau pretty closely. We first see the SVD of $A \in \mathbb{C}^{m\times n}$ as

$$AV = \Sigma U,$$

where

- $U = (u_1|u_2|u_3|\ldots|u_m)$, the matrix that has as its columns the orthoronormal vectors in the direction of the principal semiaxes of the image of the unit sphere $S \in \mathbb{R}^n$.
- $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is the diagonal matrix containing the lengths of the principal semiaxes. We order them as

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0.$$

- Then $V = (v_1|v_2|v_3|\ldots|v_n)$ is the matrix whose columns are the preimages of the principal semiaxes, i.e., $Av_i = \sigma_i u_i$. It should be clear that the $\{v_i\}$ form an orthonormal set, so $V^{-1} = V^\star$.

7.3 Examples

We'll look at some examples in class, using a Matlab script. One of these will be based on the matrix

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

In Figure 7.1 we show S, the unit circle in $\mathbb{R}^2$. In Figure 7.3 is its image under A. In between are shown VS (left in Figure 7.2), which as you can see causes a rotation of S, but no stretching (because V is unitary, $\|x\|_2 = \|Vx\|_2$; and ΣVS (bottom), which does the stretching. Multiplication by U again rotates giving the final image.

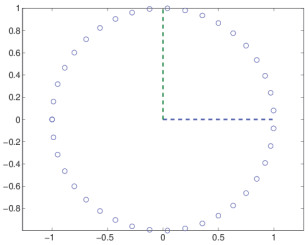


Figure 7.1: The unit circle S in $\mathbb{R}^2$

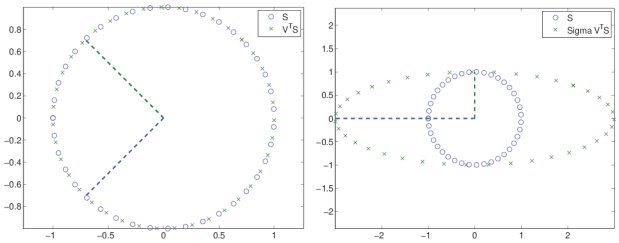


Figure 7.2: VS (left) and ΣVS

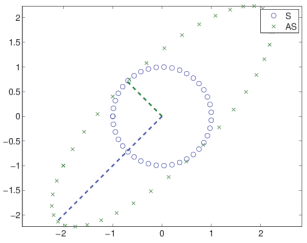


Figure 7.3: AS, the image of the unit circle

The SVD for this is

$$A = \begin{pmatrix} \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{-1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

8 One theorem about the SVD

This class started very slowly: we spent quite some time discussion (roughly) 5 different proof to the assertion that $\|AQ\|_2 = \|A\|_2$, where Q is unitary. These included

- Use that $\|A\|_2$ is the (positive) square root of the largest eigenvalue of $AA^\star$.
- Use that $\|Q\|_2 = 1$, and the consistency of matrix norms:

$$\begin{aligned}\|AQ\|_2 \leqslant \|A\|_2\|Q\|_2 = \|A\|_2 = \|AQQ^\star\|_2 \\ \leqslant \|AQ\|_2\|Q^\star\| = \|AQ\|_2.\end{aligned}$$

- $$\begin{aligned}\|AQ\|_2 &:= \max_{\|\mathbf{x}\|\neq\mathbf{0}} \frac{\|AQ\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \max_{\|\mathbf{x}\|\neq\mathbf{0}} \frac{\|AQ\mathbf{x}\|_2}{\|Q\mathbf{x}\|_2} \\ &= \max_{\|Q\mathbf{x}\|\neq\mathbf{0}} \frac{\|AQ\mathbf{x}\|_2}{\|Q\mathbf{x}\|_2} =: \|A\|_2.\end{aligned}$$

- Etc.

8.1 The (Full) SVD

The (full) SVD of $A \in \mathbb{C}^{m\times n}$ is the factorisation $A = U\Sigma V^\star$, where

$$U \in \mathbb{C}^{m\times m} \text{ is unitary,}$$

$$\Sigma \in \mathbb{R}^{m\times n} \text{ is diagonal,}$$

$$V \in \mathbb{C}^{n\times n} \text{ is unitary.}$$

Further, the diagonal entries of Σ are $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \sigma_p \geqslant 0$, where $p = \min(m, n)$.

The columns of U are called the *left singular values* of A, and the columns of V are called the *right singular values* of A.

We then started on the proof of the following result:

Theorem 8.1. *Every matrix has an SVD, and the singular values are uniquely determined. If the matrix is square and the singular values distinct, the left and right singular vectors are uniquely determined up to complex sign.*

In the class, we looked a (rushed) argument on the existence of the SVD. For the rest of the proof, have a look at Trefethen+Bau, Lecture 4.

9 Nine more theorems about the SVD

This is just a draft version of today's notes; a longer version with more exercises will be posted to the website soon...

9.1 The theorems

Theorem 9.1. *The rank of* A *is the number of nonzero singular values.*

Theorem 9.2. $\operatorname{range}(A) = \operatorname{span}(u_1, u_2, \dots, u_r)$ *and* $\operatorname{null}(A) = \operatorname{span}(v_{r+1}, \dots, v_n)$ *where* r *is the number of nonzero singular values of* A*.*

In (6.1), we defined the most important example of a matrix norm that is *not* induced by a vector norm – the *Frobenius norm*:

$$\|A\|_F := \Big(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\Big)^{1/2}.$$

Theorem 9.3. $\|A\|_2 = \sigma_1$. $\|A\|_F = \|\operatorname{diag}(\Sigma)\|_2$.

Theorem 9.4. *The singular values of* A *are the square roots of the eigenvalues of* $A^\star A$*.*

Theorem 9.5. *If* A *is hermitian, the singular values of* A *are the absolute values of the eigenvalues of* A*.*

Theorem 9.6. *If* $A \in \mathbb{C}^{m\times m}$, $|\det(A)| = \prod_{i=1}^{m} \sigma_i$.

Theorem 9.7. A *is the sum of the* r *rank-one matrices*

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^\star.$$

Theorem 9.8. *Let* A_V *be the rank-*ν *approximation to* A

$$A_\nu = \sum_{j=1}^{r} \sigma_j u_j v_j^\star.$$

Then

$$\|A - A_\nu\|_2 = \inf_{\operatorname{rank}(X)\leqslant\nu} \|A - X\|_2 = \sigma_{\nu+1},$$

where if $\nu = p = \min(m, n)$*, we define* $\sigma_{\nu+1} = 0$.

Theorem 9.9 (Schmidt)**.**

$$\|A - A_\nu\|_F = \inf_{\operatorname{rank}(X)\leqslant\nu} \|A - X\|_F = \sqrt{\sigma_{\nu+1}^2 + \sigma_{\nu+1}^2 + \cdots + \sigma_r^2}.$$

All the above results emphasise the power the power of the SVD as an important theoretical and computational tool.

Computing the SVD is an important task, but not a trivial one. There are a few different approaches, but a key idea is the QR-*factorisation*.

. . . . . .

9.2 Exercises

Exercise 9.1. Suppose you want to solve the linear system $Ax = b$, and you already have computed the SVD of A. How would you solve the linear system?

Exercise 9.2. Prove the second part of Theorem 9.2: if the SVD of A is $U\Sigma V^\star$, then the null space of A is spanned by columns $r+1, \dots, n$ of V, where where r is the number of nonzero singular values of A.

Exercise 9.3. Prove Theorem 9.9.

10 Solving Differential Equations

Today (after finishing off a left-over topic from the SVD) we'll take a detour to study numerical schemes for solving differential equations: finite differences (first) and finite element methods. Both of these approaches lead of linear systems of equations that must be solved, which is our main topic of interest.

10.1 Some basic DE theory

The particular ordinary differential equations that we are interested in are called *Boundary Value Problems (BVPs)*. We'll consider problems in one and two dimensions.

The one-dimensional problem is: *find a function* $u = u(x)$ *such that*

$$-u''(x)+a(x)u'(x)+b(x)u(x) = f(x), \quad \text{for } 0 \leqslant x \leqslant 1,$$

with $u(0) = u_0$ and $u(1) = u_1$. Here, there is no importance associated with posing the DE on the unit interval: we could easily transform to any interval. Also, it is easy to transform to a problem that satisfies homogeneous boundary conditions.

It is helpful to formulate this problem in terms of a *differential operator*. Later we'll see how to replace the differential equation with a matrix-vector problem, where the matrix approximates the differential operator. Our operator is:

$$L(u) := -u''(x) + a(x)u'(x) + b(x)(u). \tag{10.1}$$

Then the general form of our BVP is: *solve*

$$\begin{aligned} L(u) = f(x) \quad &\text{for} \quad 0 < x < 0, \\ u(0) = u_0, &u(1) = u_1. \end{aligned} \tag{10.2}$$

We make the assumptions that a, b and f are continuous and have as many continuous derivatives as we would like. Furthermore we always assume that there is a number β_0 such that $b(x) \geqslant \beta_0 > 0$.

Theorem 10.1 (Maximum Principle)**.** *Suppose* u *is a function such that* $Lu \geqslant 0$ *on* $(0,1)$ *and* $u(0) \geqslant 0$, $u(1) \geqslant 0$. *Then* $u \geqslant 0$ *for all* $x \in [0,1]$.

This result has numerous applications. For example,

Corollary 10.2. *Suppose that* $a \equiv 0$. *Define* $C = \max_{0\leqslant x\leqslant 1} |f(0)|/\beta_0$. *Then* $u(x) \leqslant C$.

Corollary 10.3. *There is at most one solution to (10.2).*

10.2 Numerical solution of BVPs

Most boundary value problems are difficult or impossible to solve analytically. So we need approximations. Two of the most popular numerical methods for this are

- Finite Difference Methods (FDMs) — where one approximates the differential operator L, based on Taylor series expansions.
- Finite Element Methods (FEMs) – where to solution space is approximated.

We'll next study FDMs, then FEMs. Then we'll study numerical schemes for solving the linear systems that they give rise to.

10.3 Exercises

Exercise 10.1. Although (10.2) is expressed in terms of arbitrary boundary conditions, without loss of generality, one can assume that the differential equations has homogeneous boundary conditions. That is, that $u(x) = 0$ at the boundaries.

Show how to find a problem which is expressed in terms of the differential operator defined in (10.1), has homogeneous boundary conditions, and with a solution that differs from this one only by a known linear function.

Exercise 10.2. Prove Corollary 10.2.

Exercise 10.3. Prove Corollary 10.3.

11 Finite difference methods (1)

Choose a set of points $\{0 = x_0 < x_1 < \cdots < x_N = 0\}$, called the *mesh*, with $h = x_i - x_{i-1} = 1/N$. Replace the derivatives in (10.2) with difference formulae. We can deduce these in several ways:

(i) Geometrically. Recall, all we are really looking for is the slope of the tangent to $f(x)$ at $x = x_i$.

(ii) By finding a polynomial that interpolates f and differentiating that.

(iii) Undetermined coefficients.

(iv) Taylor's Theorem.

For more details, see, e.g., Lecture 24 of Stewart's *Afternotes* ([Ste98]). We'll focus on the Taylor Series approach.

Theorem 11.1 (Taylor's Theorem)**.** *Suppose that* f *is continuous and* $n+1$ *times differentiable on the interval* $[a, b]$*. Then for every* $x \in [a, b]$*, we can write*

$$u(x) = u(a) + (x-a)u'(a) + \frac{(x-a)^2}{2}u''(a) + \frac{(x-a)^3}{3!}u'''(a) + \cdots + \frac{(x-a)^n}{n!}u^{(n)}(a) + R_n.$$

where the remainder R_n *is be given by*

$$R_n = \frac{(x-a)^{n+1}}{(n+1)!}u^{(n+1)}(\eta),$$

for some $\eta \in (a, b)$*. Q*

We can use this to deduce our rules. In particular:

- the backward difference scheme for u'

$$u'(x_i) = \frac{1}{h}(u(x_i) - u(x_{i-1})) + \frac{h}{2}u''(\tau), \quad (11.1)$$

- The central difference scheme for u'

$$u'(x_i) = \frac{1}{2h}(u(x_{i+1}) - u(x_{i-1})) + \mathcal{O}(h^2) \quad (11.2)$$

- The central difference scheme for u''.

$$u''(x_i) = \frac{1}{h^2}(u_{i-1} - 2u_i + u_{i+1}) + \mathcal{O}(h^2). \quad (11.3)$$

11.1 Discretisation

Let's restrict our interest, for now, to the DE:

$$-u'' + b(x)u(x) = f(x). \quad (11.4)$$

We will seek an approximation on the uniformly spaced points $x_i = ih$ where $h = 1/N$. We denote by u_i the approximation for u at $x = x_i$. The finite difference scheme gives the following system of linear equations:

$$\begin{aligned} U_0 &= 0, \\ -\Big(\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2}\Big) + b(x_i)U_i &= f(x_i), \\ i &= 1, 2, \ldots, n-1, \\ U_n &= 0. \end{aligned}$$

In keeping with our approach to studying differential equations, we'll think of the finite difference formula as defining an operator:

$$L^N U_i := -\Big(\frac{U_{i-1} - 2U_i + U_{i+1}}{h^2}\Big) + b(x_i)U_i. \quad (11.5)$$

Some of the properties of the differential operator carry over to the difference operator. In particular, we have the following analogue of Theorem 10.1.

Theorem 11.2 (Discrete Maximum Principle)**.** *Suppose that* $\{V_i\}_{i=0}^N$ *is a mesh function such that* $L^N V_i \geqslant 0$ *on* $x_1, \ldots x_{N-1}$*, and* $V_0 \geqslant 0$*,* $V_N \geqslant 0$*. Then* $V_i \geqslant 0$ *for* $i = 0, \ldots, N$.

This very useful result can be used to establish many facts, including that the solution to (11.5) is unique. Another important result is:

Corollary 11.3. *Let* $\{V_i\}_{i=0}^N$ *be any mesh function with* $V_0 = V_N = 0$*. Then*

$$|V_i| \leqslant \beta_0^{-1} \|L^h V_i\|_\infty$$

11.2 Exercises

Exercise 11.1. Suppose that we actually want to solve

$$-\varepsilon u'' + a u'(x) = f(x) \quad \text{on } (0, 1),$$

where ε and a are positive constants.
If the difference operator now includes the approximation for $u'(x_i)$ given in (11.1), show that L^N satisfies a discrete maximum principle.
If we use the approximation for $u'(x_i)$ given in (11.2), what assumptions must be made on a, ε and N for L^N to satisfy a discrete maximum principle?

12 Matlab interlude: Orthogonal Matrices, and the SVD

In this class we stepped back from our study of finite difference methods to revisit topics on orthogonal matrices, and low-rank approximation using the SVD, with applications to image processing.

The notes are available at
www.maths.nuigalway.ie/~niall/MA519/lab1/lab1.pdf

When I get the chance, I'll flesh out this page with some details about low-rank approximation.

In the mean time, here is the black and white version of the official NUI Galway logo. It is represented as a 154×500 pixel image in 616000 bytes (8 bytes per pixel).

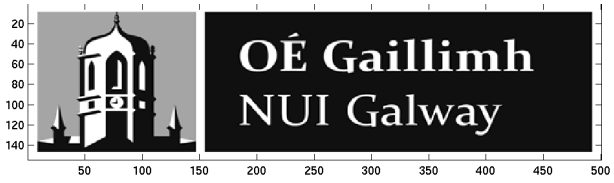


Figure 12.1: Original image of the NUI Galway logo. It has rank 125

.

Now here in Figure 12.2 is the distribution of the singular values. We can see from this that the matrix does not have full rank. In fact, the rank is 125.

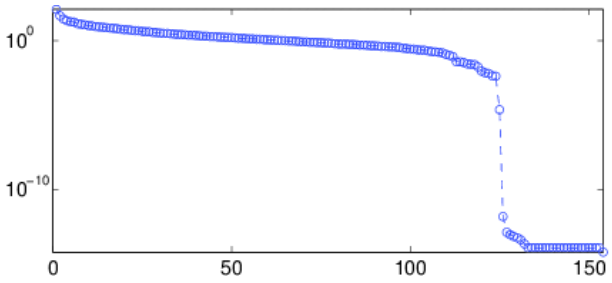


Figure 12.2: The distribution of the singular values of the image shown in Figure 12.1

Here in Figure 12.3 is the rank 2 approximation. It required just 1310 bytes, but is not have a good likeness to the original. Can you tell from looking at it that it is a rank 2 image?

In Figure 12.4 we show the rank 10 approximation. It required 6550 bytes, but the image is quite clear.

Finally, in Figure 12.5 I give that rank 40 approximation (26200 bytes). It is difficult to discerne any difference between this a the one in Figure 12.1.

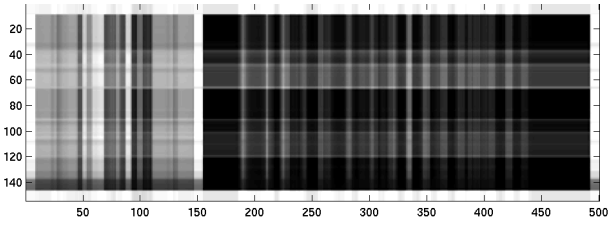


Figure 12.3: Rank 2

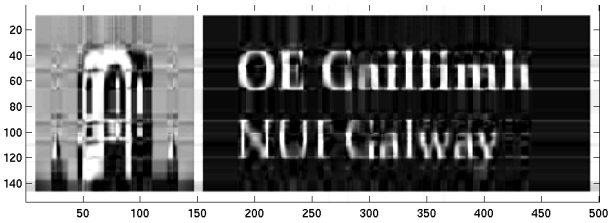


Figure 12.4: Rank 10 approximation of the image in Figure 12.1

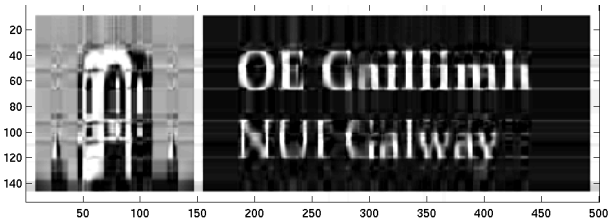


Figure 12.5: Rank 40 approximation of the image in Figure 12.1

13 Finite difference methods (2)

We actually spend most of the lecture reviewing material from Lecture 11, since it had been a week since we covered that, and proving Theorem 11.2 and Corollary 11.3.

13.1 Numerical Analysis in 1D

Let u be the solution to the differential equation (11.4), and let U be its finite difference approximation (11.5). We want to estimate the error: $\|u - U\|_\infty$.

Let δ^2 be the operator we derived in (11.3):

$$\delta^2 U_i := \frac{1}{h^2}\big(U_{i-1} - 2U_i + U_{i+1}\big),$$

and recall that we can deduce from truncated Taylor's series that

$$u''(x_i) - \delta^2 u(x_i) = \frac{h^2}{12}u^{(iv)}(\tau) \quad \text{for } \tau \in (x_{i-1}, x_{i+1}).$$

From this one can show that, for all i

$$|u''(x_i) - \delta^2 u(x_i)| \leqslant \frac{h^2}{12}M, \tag{13.1}$$

where $\|u^{(iv)}(x)\| \leqslant M$ for all x in $[0,1]$.

We will use Theorem 11.2, Corollary 11.3, and ((13.1)) to show that

Theorem 13.1. *Suppose that $u(x)$ is the solution to the problem:*

$$Lu(x) = f(x), \qquad u(0) = u(1) = 0$$

and $\|u^{(iv)}(x)\|_\infty \leqslant M$. Let U be the mesh function that solves

$$L^N U_i = f(x_i) \quad \textit{for } i = 1, 2, \ldots, N-1, \quad U_0 = U_N = 1.$$

Then

$$\max_{i=0,\ldots,N} |u(x_i) - U_i| \leqslant \frac{h^2}{12}\frac{M}{\beta_0}$$

One might think that the numerical linear algebraist would not care too much about such analyses. However, we'll see that when we use iterative methods for solving linear systems that arise from the discretisation of DEs, we need to know the discretisation error, so it can be matched by the solver error.

13.2 Practical issues

Because our problem of interest has zero boundary conditions, the finite difference method is equivalent to solving a linear system of $N-1$ equations. That is, we think of L^N as a *tridiagonal* matrix. Most of the next few lectures will be concerned with solving the associated linear system.

13.3 Exercises

Exercise 13.1. [Dem97, Lemma 6.1] Suppose we want to apply a finite difference method to the problem

$$-u''(x) = f(x) \quad \text{on } (0,1)$$

with $u(0) = u(1) = 0$, and we decide to write the linear system as

$$T\mathbf{U} = h^2\mathbf{f},$$

where $\mathbf{U} = (U_1, \ldots, U_{N-1})^T$ and $\mathbf{f} = (f(x_1), \ldots, f(x_{N-1}))$. Show that the eigenvalues of T are

$$\lambda_j = 2\big(1 - \cos(\pi j/N)\big),$$

with corresponding normalised eigenvectors

$$z_i^j = \sqrt{\frac{2}{N}}\sin(ji\pi/N).$$

Exercise 13.2. Write (11.5) as a matrix-vector equation. Show that the eigenvalues of the matrix are positive real numbers. (Note: unlike Exercise 13.1, we have that $b(x) \geqslant \beta_0 > 0$) *Hint: read up on Gerschgorin's theorems.*

14 Finite differences in 2D

14.1 A two dimensional differential equation

The next problem we wish to solve is

$$-\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u(x,y)+b(x,y)=f(x,y), \quad (14.1)$$

for all x in the unit square: $(0,1)\times(0,1)$, and with the homogeneous boundary conditions:

$$u(0,y)=u(1,y)=u(x,0)=u(x,1)=0.$$

Often we express this in terms of the Laplacian operator:

$$\Delta u := \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u.$$

For simplicity, initially we'll consider $b\equiv 0$ in (14.1), which gives the standard *Poisson* problem:

$$-\Delta u(x,y)=f(x,y) \quad \text{on } (0,1)^2.$$

14.2 Finite differences in two dimensions

[You might need to notes from class to make complete sense of this section. You should also read Section 6.3.2 of Demmel's *Applied Numeral Linear Algebra* [Dem97]].

We divide the unit square into a regular grid of $(N+1)^2$ points, with $h = x_i - x_{i-1} = y_j - y_{j-1}$. Let us denote the approximation for u at the point (x_i, v_j) by $U_{i,j}$. We can approximate the partial derivatives using the difference formula of (11.3):

$$\frac{\partial^2}{\partial x^2}u(x_i,y_j)\approx\frac{1}{h^2}(U_{i-1,j}-2U_{i,j}+U_{i+1,j}).$$

$$\frac{\partial^2}{\partial y^2}u(x_i,y_j)\approx\frac{1}{h^2}(U_{i,j-1}-2U_{i,j}+U_{i,j+1}).$$

Combining these we get

$$\begin{pmatrix} & -U_{i,j+1} & \\ -U_{i-1,j} & 4U_{i,j} & -U_{i+1,j} \\ & -U_{i,j-1} & \end{pmatrix} = h^2 f(x_i,y_j). \quad (14.2)$$

There are two ways of representing (14.2):

(a) we can think of U as a matrix of values, which is what we'll do for the rest of this lecture, or

(b) with a suitable ordering, we can think of U as a vector, which is what we'll do for most of the rest of this module.

14.3 The matrix representation of (14.2)

There are a total of $(N+1)^2$ values of $U_{i,j}$ we need to compute, each associated with the a node on the two-dimensional mesh (see diagram from class). However, we know the values at the boundary, so we only need to compute $(N-1)^2$ values:

$$U_{i,j} \quad i=1,\dots,N-1, \quad j=1,\dots,N-1.$$

Suppose we store these values as a matrix

$$U=\begin{pmatrix} U_{1,1} & U_{1,2} & \cdots & U_{1,N-1}. \\ U_{2,1} & U_{2,2} & \cdots & U_{2,N-1}. \\ \vdots & \vdots & & \vdots \\ U_{N-1,1} & U_{N-1,2} & \cdots & U_{N-1,N-1}. \end{pmatrix}$$

Let $T_N \in \mathbb{R}^{(N-1)\times(N-1)}$ matrix:

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

(Recall also Exercise 13.1.)

The we saw that we can represent the method (14.2) as

$$(T_N U + U T_N) = h^2 f(x_i, y_j). \quad (14.3)$$

We finished by observing that if, T_N has eigenvalues λ_i and λ_j with corresponding eigenvectors z_i and z_j, then $\lambda_i+\lambda_j$ is an "eigenvalue" of the system represented in (14.3), with corresponding eigenvector $V = z_i z_j^T$, in the sense that

$$T_N V + V T_N = (\lambda_i+\lambda_j)V.$$

The argument presented was taken directly from [Dem97, p. 271].

15 Properties of the 2D finite difference matrix

15.1 Recall...

We began by recapping on Lecture 14: we wish to find a numerical approximation to the solution to the Poisson Problem:

$$-\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u(x,y)=f(x,y),$$

for all $(x,y)\in(0,1)^2$, and $u=0$ on the boundary of $(0,1)^2$. We construct a mesh (i.e., a grid) of evenly spaced points $\{(x_i,y_j)\}_{i=0,j=0}^{N,N}$, where $x_i-x_{i-1}=y_j-y_{j-1}=h$.

We revisited that, if we think of the $U_{i,j}$ as being represented by an matrix, then (14.2) could be expressed as et $T_N\in\mathbb{R}^{(N-1)\times(N-1)}$ matrix:

$$(T_N U+UT_N)=h^2 f(x_i,y_j).$$

where T_N is

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}. \tag{15.1}$$

15.2 The matrix version

We now want to see how to express the method as a more standard matrix-vector equation.

For the $(N+1)\times(N+1)$ mesh we described, we are actually only interested in the $(N-1)^2$ interior points, since we know the values at the boundary. With lexicographic ordering, we represent U as

$$U=(U_1,U_2,\ldots,U_{(N-1)^2})^T,$$

where U_k corresponds to $U_{i,j}$ if $k=i+(N-1)(j-1)$. The k^{th} row of the method (14.2) becomes

$$\begin{pmatrix} & -U_{k+(N-1)} & \\ -U_{k-1} & 4U_k & -U_{k+1} \\ & -U_{k-(N-1)} & \end{pmatrix}$$

15.3 Example: N=3

See you notes from class, which included showing that if we write the system as

$$T_{3,3}U=h^2F,$$

then $T_{3,3}$ is the matrix

$$\left(\begin{array}{ccc|ccc|ccc} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{array}\right)$$

which can be written more succinctly, and more generally, as

$$\left(\begin{array}{c|c|c} T_N+2I & -I & 0 \\ \hline -I & T_N+2I & -I \\ \hline 0 & -I & T_N+2I \end{array}\right)$$

15.4 Kronecker products

Definition 15.1. If $A\in\mathbb{R}^{m\times n}$ and $B\in\mathbb{R}^{p\times q}$, then the *Kronecker product* of A and B is the $A\otimes B\in\mathbb{R}^{(mp)\times(nq)}$ given by

$$\begin{pmatrix} a_{11}B & a_{12}B & \ldots & a_{1n}B \\ a_{21}B & a_{22}B & \ldots & a_{2n}B \\ \vdots & & & \\ a_{m1}B & a_{m2}B & \ldots & a_{mn}B \end{pmatrix}$$

We then did several examples, culminating in noting that

$$T_{3,3}=I\otimes T_3+T_3\otimes I.$$

To show that this holds in general, we introduce a new operator:

Definition 15.2. If $A\in\mathbb{R}^{m\times n}$ then vec(A) is the vector in $\mathbb{R}^{mn}$ formed from stacking the columns of A. That is

$$\text{vec}(A)=(a_{11},a_{21},a_{31},\ldots,a_{m1},a_{21},\ldots,a_{mn})^T.$$

Theorem 15.3 (Demmel, Lemma 6.2)**.** *Let* $A,B,X\in\mathbb{R}^{n\times n}$*. Then*

(a) $(AX)=(I\otimes A)\text{vec}(X)$.

(b) $(XB)=(B^T\otimes I)\text{vec}(X)$.

(c) The formulation for the finite difference method

$$(T_N U+UT_N)=h^2F,$$

is equivalent to

$$T_{N,N}\text{vec}(U)=(I\otimes T_N+T_N\otimes I)\text{vec}(U)=h^2\text{vec}(F).$$

We left (a) and (b) as exercises. The proof of (c) comes directly from (a) and (b) and from noting that T_N is symmetric.

15.5 Exercise

Exercise 15.1. Prove Parts (a) and (b) of Theorem 15.3.

16 Gaussian Elimination

Last week we studied the finite difference method for finding approximate solutions to partial differential equations, which led to the construction of linear systems of equations that must be solved. For the next few weeks, we'll study how to do that. We begin with an overview of the most classic method, but our main focus will be on methods for special matrices.

16.1 Gaussian Elimination

Suppose we want to solve $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{m\times m}$ is nonsingular and $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^m$. The classical method that we all learned at school is called *Gaussian Elimination*: preform a series of elementary row operations to reduce the system to a lower triangular one, which is then easily solve with back-substitution.

Suppose the matrix is

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

The first step is to eliminate the term a_{21}. This is done by replacing A with

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22}+\mu_{21}a_{12} & a_{23}+\mu_{21}a_{13} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = A + \mu_{21}\begin{pmatrix} 0 & 0 & 0 \\ a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 \end{pmatrix}$$

where $\mu_{21} = -a_{21}/a_{11}$. Because

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

the row operation can be written as $\big(I + \mu_{21}E^{(21)}\big)A$, where $E^{(pq)} \in \mathbb{R}^{m\times m}$ has all zeros, except for $e_{pq} = 1$.

Next we eliminate the term a_{31} and then a_{32} so that, when we are done, A has been reduced to an *upper triangular matrix*.

In general each of the row operations in Gaussian Elimination can be written as

$$\big(I + \mu_{pq}E^{(pq)}\big)A \quad \text{where } 1 \leqslant q < p \leqslant m, \tag{16.1}$$

and $\big(I + \mu_{pq}E^{(pq)}\big)$ is an example of a *Unit Lower Triangular Matrix*.

16.2 Triangular Matrices

Recall from Section 2.4 that $L \in \mathbb{R}^{m\times m}$ is a *lower triangular (LT) matrix* if the only non-zero entries are on or below the main diagonal, i.e., if $l_{ij} = 0$ for $1 \leqslant i \leqslant j \leqslant m$. It is a *unit lower triangular (ULT) Matrix* is $l_{ii} = 1$.

The analogous definitions of Upper Traiangular and Unit Upper Triangular matrices is obvious.

In Example 2.3 we saw that the product of two lower (upper) triangular matrices is lower (upper) triangular. Most importantly for us, as seen in Q4 of Problem Set 1:

(a) the product of two ULT matrices is a ULT matrix.

(b) the inverse of a ULT matrix always exists, and is a ULT matrix.

16.3 Factorising A

Each elementary row operation in Gaussian Elimination (GE) involves replacing A with $(I + \mu_{rs}E^{(rs)})A$. But $(I + \mu_{rs}E^{(rs)})$ is a unit lower triangular matrix. Also, when we are finished we have an upper triangular matrix. So we can write the whole process as

$$L_k L_{k-1} L_{k-2} \dots L_2 L_1 A = U, \tag{16.2}$$

where each of the L_i is a ULT matrix. Since we know that the product of ULT matrices is itself a ULT matrix, we can write the whole process as

$$\tilde{L}A = U.$$

We also know that the inverse of a ULT matrix exists and is a ULT matrix. So we can write

$$A = LU,$$

where L is unit lower triangular and U is upper triangular. This is called the LU-*factorization* of the matrix.

16.4 Exercises

Exercise 16.1. Suppose that $B \in \mathbb{R}^{m\times m}$ is a matrix of the form

$$B = I + \boldsymbol{v}e_j^\star.$$

where $\boldsymbol{v} \in \mathbb{R}^m$ with $v_j = 0$. Show that $B^{-1} = I - \boldsymbol{v}e_j^\star$.

Exercise 16.2. We'll now use the exercise above to show that the inverses of the matrices $L_1, L_2, \dots, L_k$ on the left of (16.2) have a convenient form. If we let $l_{i,j} = x_{ij}/x_{jj}$, then we can write

$$L_j = I + \boldsymbol{l}_j\boldsymbol{e}_j,$$

where $l_j^\star = (0, 0, \dots, 0, -l_{j+1,j}, -l_{j+2,j}, \dots, -l_{m,j})$.

Show that if $A = LU$, then

$$L = \prod_{j=1}^{k} L_j^{-1},$$

can be expressed as

$$L = I + \sum_{j=1}^{k} \boldsymbol{l}_j\boldsymbol{e}_j^\star.$$

17 LU-factorisation

17.1 Formulae for L and U

In Lecture 16, we saw that applying Guassian Elimination to a linear system $A\boldsymbol{x} = \boldsymbol{b}$ is equivalent to left-multiplication of A by series unit lower triangular matrices, yielding an upper triangular matrix.

Definition 17.1. The **LU-factorization** of the matrix is a unit lower triangular matrix L and an upper triangular matrix U such that $LU = A$.

Now we'll derive a formula for L and U. Since $A = LU$, the entries of A are $a_{i,j} = (LU)_{ij} = \sum_{k=1}^{m} l_{ik}u_{kj}$. Since L and U are triangular,

$$\text{If } i \leqslant j \quad \text{then} \quad a_{i,j} = \sum_{k=1}^{i} l_{ik}u_{kj}$$

This can be written as

$$a_{i,j} = \sum_{k=1}^{i-1} l_{ik}u_{kj} + l_{ii}u_{ij}.$$

But $l_{ii} = 1$ so:

$$u_{i,j} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \quad \begin{cases} i = 1, \dots, j-1, \\ j = 2, \dots, m. \end{cases} \tag{17.3a}$$

Similarly, one can show that

$$l_{i,j} = \frac{1}{u_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}\right) \quad \begin{cases} i = 2, \dots, m, \\ j = 1, \dots, i-1. \end{cases} \tag{17.3b}$$

17.2 Solving $LU\boldsymbol{x} = \boldsymbol{b}$

We can now factorise A as $A = LU$. But we are trying to solve the problem: *find* $\boldsymbol{x} \in \mathbb{R}^m$ *such that* $A\boldsymbol{x} = \boldsymbol{b}$, *for some* $\boldsymbol{b} \in \mathbb{R}^m$. So solve

$$L\boldsymbol{y} = \boldsymbol{b} \text{ for } \boldsymbol{y} \in \mathbb{R}^n \quad \text{and then } U\boldsymbol{x} = \boldsymbol{y}.$$

Because L and U are triangular, this is easy.

Example 17.2. Use LU-factorisation to solve

$$\begin{pmatrix} -1 & 0 & 1 & 2 \\ -2 & -2 & 1 & 4 \\ -3 & -4 & -2 & 4 \\ -4 & -6 & -5 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -2 \\ -3 \\ -1 \\ 1 \end{pmatrix}$$

17.3 Pivoting

Not every matrix has an LU-factorisation. Consider, for example

$$A = \begin{pmatrix} 0 & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \quad \text{and } B = \begin{pmatrix} d & e & f \\ 0 & b & c \\ g & h & i \end{pmatrix}$$

A does not have an LU-factorisation, but B does. We can think of B as a permutation of A.

Definition 17.3. P is a *Permutation Matrix* if every entry is either 0 or 1 (i.e., it is a Boolean Matrix) and if all the row and column sums are 1.

Theorem 17.4. *For any* $A \in \mathbb{R}^{m\times m}$ *there exists a permutation matrix* P *such that* $PA = LU$.

We won't prove this in class, but you can find the argument in any good numerical analysis textbook.

17.4 The computational cost

How many computational steps (additions and multiplications) are required to compute the LU-factorisation of A? Suppose we want to compute $l_{i,j}$. From the formula (17.3b) we see that this would take $j-2$ additions, $j-1$ multiplications, 1 subtraction and 1 division: a total of $2j-2$ operations. Recall that

$$\sum_{i=1}^{k} i = \frac{1}{2}k(k+1), \text{ and } \sum_{i=1}^{k} i^2 == \frac{1}{6}k(k+1)(2k+1)$$

So the total number of operations required for computing L is

$$\sum_{i=2}^{m}\sum_{j=1}^{i-1}(2j-1) = \sum_{i=2}^{m} i^2 - 2i + 1 = \frac{1}{6}m(m+1)(2m+1) - m(m+1) \leqslant Cm^3$$

for some C. A similar (slightly smaller) number of operations is required for computing U.

17.5 Exercises

Exercise 17.1. Many textbooks and computing systems compute the factorisation $A = LDU$ where L and U are unit lower and *unit* upper triangular matrices respectively, and D is a diagonal matrix. Show that such a factorisation exists.

Exercise 17.2. Suppose that $A \in \mathbb{R}^{m\times m}$ is nonsingular and symmetric, and that every leading principle submatrix of A is nonsingular. Use Exercise 17.1 so show that A can be factorised as $A = LDL^T$.
How would this factorization be used to solve $A\boldsymbol{x} = \boldsymbol{b}$?

Exercise 17.3. Consider the matrix

$$H_3 = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}.$$

Write down the LU-factorization of H_3, and hence solve the linear system $H_3\boldsymbol{x} = b$ where $b = \frac{1}{12}(6, 2, 1)^T$.

18 Symmetric positive definiteness

These notes were revised from the ones provided in class. In particular, the exercises were added.

18.1 Overview

For the next few classes, we want to study methods for solving $Ax = b$ where A has special properties, such as being symmetric, sparse, or banded (all properties of the finite difference matrices from Lecture 15).

Suppose we wanted to solve $Ax = b$ where A is symmetric. Since $A = A^T$, we might like to think that we could use a version of LU-factorisation where $L = U^T$. However (see [Ips09, Section 3.6]), this can't be done. Consider the example

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

18.2 s.p.d. matrices

LU-factorisation is the most standard method. But many problems, such as those from finite difference methods, have very special structure that we can exploit in order to save time and memory. The first case we'll consider is if A is s.p.d.

Definition 18.1 (s.p.d.)**.** A matrix, $A \in \mathbb{R}^{m\times m}$, is *symmetric positive definite* (s.p.d.) if and only if $A = A^T$ and $x^TAx > 0$ for all vectors $x \neq 0$.

It is easy to see that such a matrix is nonsingular. If A were s.p.d. and singular, then there would exist some vector x such that $Ax = 0$, and so $x^TAx = 0$. This is not possible since $x^TAx > 0$ for all x.

Theorem 18.2 (Demmel, Prop 2.2)**.** *(a) If X^{-1} exists, A is s.p.d. $\iff$ X^TAX is s.p.d.*

(b) If A is s.p.d., then any principle submatrix of A is s.p.d.

(c) A is s.p.d. $\iff$ $A = A^T$ and all the eigenvalues of A are positive.

(d) If A is s.p.d., then $a_{ii} > 0$ for all i, and $\max_{ij} |a_{ij}| = \max_i a_{ii}$.

(e) A is s.p.d. $\iff$ there exists a unique lower triangular matrix L with positive diagonal entries such that $A = LL^T$. This is called the Cholesky factorisation of A.

In class we stepped through the proofs of (a)–(d). We'll save the proof of Theorem 18.2-(e) until the next lecture.

18.3 Exercises

Exercise 18.1. Give (at least) 3 different proofs that an s.p.d. matrix is nonsingular.

Exercise 18.2. So now we know that if A is s.p.d., then A^{-1} must exist. Must A^{-1} be s.p.d?

Exercise 18.3 (Stephen R)**.** Give (at least) 2 different proofs that the determinant of an s.p.d. matrix is strictly positive.

Exercise 18.4. We say a matrix $A \in \mathbb{C}^{m\times m}$ is *hermitian positive definite* if $z^*Az > 0$ for all $z \in \mathbb{C}^m$. Which of the five properties in Theorem 18.2 extend to *hermitian positive definite* matrices?

Exercise 18.5 (James McT)**.** Is it possible for a matrix $A \in \mathbb{R}^{m\times m}$ to be positive definite, in the sense that $x^TAx > 0$ for all x, and yet not be symmetric? (Hint: take $A \in \mathbb{R}^{2\times 2}$ to be of the form

$$A = \begin{pmatrix} 0 & b \\ c & 1 \end{pmatrix},$$

and try to find b and c such that $x^TAx > 0$).

Exercise 18.6 (Thái)**.** In Theorem 18.2-(d) we show that the maximum entry always occurs on the diagonal. Must the matrix always be diagonally dominant? Prove this, or give a counter-example.

Exercise 18.7. [Ips09, p66–67] Given a matrix A partitioned into four submatrices

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

a *Schur compliment* of A is $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$. In *Fact 3.27* of Ilse Ipsen's *Numerical Matrix Analysis* (available at http://www4.ncsu.edu/~ipsen/) it is shown that if A is s.p.d., so too is the Schur compliments.

Use this to show that, if A is s.p.d., then $|a_{ij}| \leqslant \sqrt{a_{ii}a_{jj}}$ for $i \neq j$. Show further that $|a_{ij}| \leqslant (a_{ii} + a_{jj})/2$ for $i \neq j$.

19 Cholesky Factorisation

(*As is often the case, these notes were heavily revised after the class, and so differ greatly from the version handed out in class*).

19.1 Recall: s.p.d. matrices

At the end of the last class, we had proved Parts (a)–(d) of Theorem 18.2. We'll start with a few examples of s.p.d. matrices.

The matrix

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2. \end{pmatrix}$$

is s.p.d. We can deduce this from the celebrated *Gerschgorin theorems*.

19.2 Gerschgorin's First Theorem

(See, e.g., [Dem97, Thm. 2.9], [Saa03, 4.6]). Given a matrix $A \in \mathbb{R}^{m\times m}$, the *Gerschgorin Discs* D_i are the m discs in the complex plane with centre a_{ii} and radius r_i:

$$r_i = \sum_{j=1, j\neq i}^{m} |a_{ij}|.$$

So $D_i = \{z \in \mathbb{C} : |a_{ii} - z| \leqslant r_i\}$. It can then be shown (Gerschgorin's First Theorem) that all the eigenvalues of A are contained in the union of the Gerschgorin discs. Furthermore (Gerschgorin's Second Theorem) if k of discs are disjoint (have an empty intersection) from the others, their union contains k eigenvalues. [1].

This has many applications. For example, we can now see that the eigenvalues of the matrix

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2. \end{pmatrix}$$

are contained in the interval $[1, 2]$, so they must be positive. By Theorem 18.2-(c). it must be s.p.d. See also Exercise 19.2.

We then observed that T_N from (15.1) is s.p.d. This is true since, by the Gerschgorin results, the eigenvalues are in the interval $[0, 2]$. Next we apply row reduction to show that T_N is invertable, and thus that the eigenvalues are actually in $(0, 2]$. So they are strictly positive. (The way I'd planned to do this in case was the invoke the formula in Exercise 13.1).

Example 19.1. Can we show that

$$T_{N,N} = I \otimes T_N + T_N \otimes I,$$

is s.p.d?

[1] In class, I'd initially, and incorrectly, assumed everyone knew these theorems. We'll devote a whole lecture to them some time soon

Answer: Recall from Section 14.3 (but see also [Dem97, p. 271]), that every eigenvalue of $T_{N,N}$ is the sum of two eigenvalues of T_N. Since the eigenvalues of T_N are positive, so too are those of $T_{N,N}$.

19.3 Another aside

We also had a short detour to discuss

- A is *diagonal* if its only non-zero entries are found on the main diagonal. That is $a_{ij} = 0$ if $i \neq j$.
- A is *tridiagonal* if its only non-zero entries are found on the main diagonal, the subdiagonal or the super-diagonal. That is $a_{ij} = 0$ if $|i-j| > 1$.
- A is *bidiagonal* if its only non-zero entries are found on the main diagonal, and one of the subdiagonal or the super-diagonal (but not both). That is A is triangular and tridiagonal.

19.4 Part (e) of Theorem 18.2 (finally!)

We now consider Part (e) of Theorem 18.2: $A \in \mathbb{R}^{m\times m}$ is symmetric positive definite (s.p.d.) if and only of there exists a unique lower triangular matrix L with positive diagonal entries such that $A = LL^T$. *This is called the Cholesky factorisation of* A.

It is important to note that this is an "if and only if" statement. For example, a standard way of testing if a matrix is s.p.d. is to try to compute the Cholesky factorisation, and to see if it fails.

19.5 Exercises

Exercise 19.1. [Dem97, Exer 6.2] Derive a formula for the LU factorisation of T_N that is similar to the Cholesky factorisation in (20.1).

Exercise 19.2. Recall that a matrix $A \in \mathbb{R}^{m\times m}$ is (strictly) diagonally dominant if

$$a_{ii} > \sum_{j\neq i} |a_{ij}A| \quad \text{for } i = 1, \ldots, m.$$

Show that a symmetric diagonally dominant matrix must be s.p.d.

Exercise 19.3. Show that the converse of Exercise 19.2 does not hold. That is, show (by example) that it is possible for a matrix to be s.p.d. but not diagonally dominant.

20 Computing Cholesky

20.1 Computing the Cholesky factorisation (1)

Here is an algorithm for computing the Cholesky factors of a positive definite matrix, which is taken verbatim from [Dem97, Algorithm 2.11]:

Algorithm 20.1.

for $j = 1$ to n

$$l_{jj} = \Big(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\Big)^{1/2}$$

for $i = j+1$ to n

$$l_{ij} = \Big(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\Big)/l_{jj}$$

end for

end for

As an example, lets suppose that A is T_N from (15.1), and that A is factorised as $A = LL^T$. Then L is lower bidiagonal with

$$L_{i,i} = \sqrt{\frac{i+1}{i}}, \quad \text{and} \quad L_{i,i-1} = -\sqrt{\frac{i-1}{i}}. \quad (20.1)$$

It is more useful (I think) to represent matrix algorithms, by giving the Matlab implementation. This is mainly due to the use of vector subscripting. However, care should be taken if using this code (for example, you should pre-allocate L as a sparse matrix). Here is the Matlab version of Alg. 20.1

```
for j=1:N
   L(j,j) = sqrt( A(j,j) - L(j,1:j-1)*L(j,1:j-1)');
   for i=j+1:N
      L(i,j) = (A(i,j) - ...
         L(i,1:j-1)*L(j,1:j-1)')/L(j,j);
   end
end
```

20.2 Alternative formulations

Here is another implementation of Cholesky factorisation, this one taken from [Dav06, Chap. 4], where it is called "up-looking" Cholesky. The motivation is as follows. Form a partition of $LL^T = A$ by the last row and column:

$$\begin{pmatrix} L_{11} & \\ L_{12}^T & l_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{12} \\ & l_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & a_{22} \end{pmatrix}$$

This gives three equations to solve:

(i) Solve $L_{11}L_{11}^T = A_{11}$ for L. That is, apply the Cholesky algorithm recursively.

(ii) Solve $L_{11}L_{12} = A_{12}$ for the vector L_{12} which is a forward-substitution (i.e., triangular solve).

(iii) Since $L_{12}^T L_{21} + l_{22}^2 = a_{22}$, set $l_{22} = \sqrt{a_{22} - L_{12}^T L_{21}}$.

Algorithm 20.2.

```
for j=1:N
   L(j, 1:j-1) = (L(1:j-1, 1:j-1) \ A(1:j-1,j))';
   L(j,j) = sqrt(A(j,j) - L(j,1:j-1)*L(j,1:j-1)');
end
```

Here the computation

```
(L(1:j-1, 1:j-1) \ A(1:j-1,j))';
```

means

$$\big(L_{i:j-1,i:j-1}^{-1} A_{1:j-1,j}\big)^T.$$

That is, it involves solving a triangular system of equations. Of course, we never form the inverse.

This version is *much* more efficient than Alg. 20.1 if both are implemented in Matlab as given, without any optimisations. For example, when I used these to solve a 1024×1024 linear system arising from the two-dimensional finite difference method discussed above, Alg. 20.1 took nearly 80 seconds, but Alg. 20.2 ran in under a second.

As we shall see, both these computations are very wasteful since, for this example, they are computing many entries in L that we should know are zero. Dealing with this (factorisation of sparse matrices) will be our next topic.

21 Sparse Cholesky

Post-lecture version. There are some changes to the verion given-out in class, particularly in Sections 21.4 and 21.5.

Today we want to focus on how the Cholesky Factorisation Algorithm works in the case where the matrix A is "sparse". There are many possible definitions for what it means for a matrix to the sparse, but there is one that most people agree with: *a matrix is sparse if it there is a benefit to exploiting the fact that it has many nonzero entries* (Wilkinson). If a matrix is not sparse we say it is *full*. Typical examples of sparse matrices include

- Diagonal matrices, such as the identity matrix.
- Tridiagonal matrices, such as the one-dimensional finite difference matrix, T_N.
- The two-dimensional finite difference matrix, T_{NN}.
- The matrix representing, say, a social network.

21.1 Cholesky of a tridiagonal matrix

Recall that the Cholesky factorisation of

$$T_N = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & & -1 & 2 \end{pmatrix}.$$

is given by the formula

$$L_{i,i} = \sqrt{\frac{i+1}{i}}, \quad \text{and} \quad L_{i,i-1} = -\sqrt{\frac{i-1}{i}}.$$

So we can see that Cholesky factor is bidiagonal.

In fact, it is always the case the Cholesky factor of an s.p.d. tridiagonal matrix is bidiagonal. There are many ways of showing this, but we'll use the idea the led to Alg. 20.2.

21.2 Banded matrices

A matrix is *banded* if the only non-zero entries are found within a certain distance of the main diagonal. More precisely, the matrix $m \times n$ matrix A is *banded* with *band-width* b if

$$a_{ij} = 0 \quad \text{whenever} \quad |i-j| > b.$$

Usually we care about the case where B is much less than $\min(m, n)$.

The most important example we have met so far is the two-dimensional Laplacian:

$$T_{N,N} = I \otimes T_N + T_N \otimes I.$$

This is an $m \times m$ banded matrix, with $m = N^2$ and a band-width of N. It has only (roughly) $4m$ non-zero entries, so it is quite sparse, considering that a full $m \times m$ matrix would have m^2 non-zeros.

If $A = LL^T$, would we expect that L is similarly sparse? There are three possible scenarios

Most optimistic: L will have non-zeros only where A has non-zeros. (This is not the case, unfortunately).

Least optimistic: L will be full. (This is not the case, fortunately).

Realistic: L will have many more non-zeros than A: roughly $m^{3/2}$.

We'll explain in class why this is likely to be. A precise justification is left as an exercise.

21.3 General concepts

Not every banded matrix has the simple structure of T_{NN}. The more general case is (see [Dem97, §2.7.3] for a detailed discussion):

Definition 21.1 (Banded)**.** A is a banded matrix with lower band-width b_L and upper band-width b_U if

$$a_{ij} = 0 \quad \text{whenever} \quad i > j + b_L \text{ or } i < j - b_U.$$

But we'll just focus on the simple case $b_L = b_U = b$.

Definition 21.2 (Fill-in)**.** If $A = LL^T$. Entries that appear in $L + L^T$ that correspond to zeros in A are called *fill-in*.

21.4 The arrow matrices

Two famous example are used to highlight the hazards associated with fill-in during factorisation:A

$$A_1 = \begin{pmatrix} 5 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 5 \end{pmatrix}$$

If $A_1 = L_1 L_1^T$, and $A_2 = L_2 L_2^T$, then all entries of L_1 on and below the main diagonal will be non-zero, where as L_2 will only have non-zeros in the main diagonal and in the last row. So it is pretty sparse. However, A_1 and A_2 are permutation equivalent.

21.5 Exercises

Exercise 21.1. In class, we outlined a proof that if A is tridiagonal and $A = LL^T$, then L is bidiagonal, using Alg. 20.2. Deduce your own proof.

Exercise 21.2. Show that if $A = T_{N,N}$, and $A = LL^T$, then L has the same band-width as A.

22 Towards fill-in analysis

We finish our section on Cholesky factorisation by taking a more graph theoretic view of the occurrence of fill-in. This is a large topic, and we have only a lecture to devote to it, so we will just summarise some of the main ideas, and emphasise examples, rather than theoretical rigour. The underlying questions are:

(a) Given a sparse matrix A, can we predict where the non-zero entries L occur;

(b) Can we permute the rows and columns of A to reduce fill-in.

Question (a) is motivated by the fact that, in practice, when we compute L we first need to know its *structure*. This is because, on your compute, a sparse matrix are not stored as a rectangular array of numbers but in some other format such as *triplet* or *compressed column format*. (I'll give a brief explanation of these, but they are not of great mathematical interest).

Question (b) is motivated by the "arrow matrices" that we saw in Section 21.4; due to time constraints we won't dwell on it.

For more information, please read Secvions 1.2 and Chapter 4 of [Dav06].

. . . . . .

Surprisingly, most of the class was spent explaining and discussing these ideas. We looked at

- The triplet form of sparse matrix storage
- Revisited the arrow matrices
- Discussed why, rather than permuting just the rows of a matrix, we permute the rows *and* the column.
- Discussed why the Cholesky factor of the matrix A could be so different from $\mathsf{PAP}^{\mathsf{T}}$.

23 The graph of a matrix

23.1 Graph of a matrix

Recall that a *graph* $G = (V, E)$ consists of a set of vertices (nodes) $V = \{1, \ldots, n\}$ and edges $E = \{(i, j) | i, j \in V\}$. We visualise this as nodes V with a line (edge) joining i to j if $(i, j) \in E$, with an arrow indicating that this edge is *from* i *to* j.

An *undirected* graph is one where the edges don't have arrows, but really this is just a short hand way of depicting that if there is a (directed) edge from i to j, there is also one from j to i.

We say there is a *path* from v_0 to v_n if there is a sequence of nodes $(v_0, v_1, \ldots, v_n)$ such that the edges (v_0, v_1), (v_1, v_2), dots (v_{n-1}, v_n) are all in E. We write this as $v_0 \rightsquigarrow v_n$.

Given an $m \times m$ matrix A, the *graph of the matrix* $G(A)$ is the graph with m vertices such that there is an edge (i, j) if $a_{i,j} \neq 0$.

Example 23.1. The graph of the one-dimensional Laplacian is ...

Example 23.2. Let A be the matrix

$$\begin{pmatrix} 4 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 \end{pmatrix}$$

Its graph is shown below in Figure 23.1

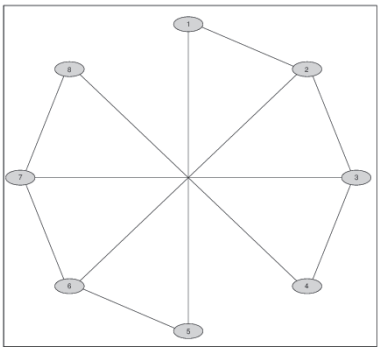


Figure 23.1: The graph of the matrix in Example 23.2

23.2 Fill-in

Recall that if $A = LL^T$, the *fill-in* refers to entries of $L + L^T$ that do not appear in A.

The graph $G(L+L^T)$ is called the *filled graph of* A.

Theorem 23.3 (Thm. 4.1 of [Dav06])**.** *The edge* (i, j) *is in the (undirected) graph* $G(L + L^T)$ *if and only if there exists a path* $i \rightsquigarrow j$ *in* $G(A)$ *where all nodes in the path, except* i *and* j *are numbered less than* $\min(i, j)$.

To see what this is telling us, consider the Cholesky factorisation of the matrix on Example 23.2. Here we just show the sparsity pattern.

$$\begin{pmatrix} * & & & & & & & \\ * & * & & & & & & \\ & * & * & & & & & \\ & & * & * & & & & \\ * & * & * & * & * & & & \\ & * & * & * & * & * & & \\ & & * & * & * & * & * & \\ & & & * & * & * & * & * \end{pmatrix}$$

23.3 Characterising fill-in for Cholesky

We need to distinguish between zeros that appear in L that are entirely due to the structure of A, and those that appear due to cancellation.

Example 23.4. If $A = I$, then all off-diagonal entries of L are zero, by necessity. However, if

$$A = \begin{pmatrix} 4 & -1 & -4 \\ -1 & 7 & 1 \\ -4 & 1 & 8 \end{pmatrix} \text{ then } L = \begin{pmatrix} 2 & & \\ -1\frac{1}{2} & \frac{3\sqrt{3}}{2} & \\ -2 & 0 & 2 \end{pmatrix}.$$

That $l_{32} = 0$ is due to cancellation.

Theorem 23.5 (Thm. 4.2 of [Dav06])**.** *If* $A = LL^T$, *and neglecting cancellation, if* $a_{ij} \neq 0$ *then* $l_{ij} \neq 0$.

Theorem 23.6 (Thm. 4.3 of [Dav06])**.** *If* $A = LL^T$, *and* $l_{ji} \neq 0$ *and* $l_{kj} \neq 0$ *where* $i < j < k$, *then* $l_{kj} \neq 0$.

23.4 Exercise

Exercise 23.1. Verify that L given in Example 23.4 is indeed the Cholesky factor of A.

24 Basic iterative methods

Today we start a new section of the course: the solution of linear systems by iterative techniques. The notes given here are not a complete record of what we did in class. For more details, see your lecture notes or [Saa03, Chap. 4].

The methods we'll look at first are classical: the Jacobi, Gauss-Seidel and SOR methods. Next week, we'll look at more modern Kirov subspace methods.

24.1 The basic idea

As usual, we want to find the vector $\mathbf{x} \in \mathbb{R}^m$, such that

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

where $A \in \mathbb{R}^{m\times m}$ and $b \in \mathbb{R}^m$. We will try to do this by constructing a sequence of vectors

$$\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots\},$$

which, we hope, will be successively better approximations to $\mathbf{x}$. For example, it might be that

$$\|\mathbf{x} - \mathbf{x}^{(k+1)}\| \leqslant \|\mathbf{x} - \mathbf{x}^{(k)}\|,$$

for some norm, or $\mathbf{x}^{(k)} \to \mathbf{x}$ as $k \to \infty$.

24.2 The residual

In both theory and practice, one of the most important quantities to work with is the *residual*:

$$\mathbf{r}^{(k)} = \mathbf{b} - A - \mathbf{x}^{(k)}.$$

Obviously, if $\mathbf{r}^{(k)} = \mathbf{0}$, we are done. So much of the focus can be on trying to minimise $\mathbf{r}^{(k)}$. Furthermore, in practical computations, we do not compute an infinite number of $\mathbf{x}^{(k)}$. Instead we would like to iterate until $\|\mathbf{x} - \mathbf{x}^{(k)}\|$ is smaller than some predetermined value. But since we can't compute $\mathbf{x} - \mathbf{x}^{(k)}\|$, we rely on making $\|\mathbf{r}^{(k)}\|$ small enough.

24.3 Jacobi's method

Suppose we want to make $\mathbf{r}_i^{(k)} = 0$ then this leads (See notes from class!) to *Jacobi's method*

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\big(b_i - \sum_{j\neq i} a_{ij}x_j^{(k)}\big). \tag{24.1}$$

Example 24.1. If we take $A = T_N$, for $N = 4$, and apply the Jacobi method to solve $A\boldsymbol{x} = h^2$, a summary of the results are as shown in Table 24.1. After 100 iterations, the error is approximately 10^{-10}.

Rather than (24.1), we shall prefer to write the method using a matrix-formulation. Let D, E and F be matrices in $\mathbb{R}^{m\times m}$ such that

- D is diagonal;
- E is strictly lower triangular (i.e., it is lower triangular, and $E_{ii} = 0$;
- F is strictly upper triangular.
- $A = D - E - F$.

Then we can write Jacobi's method as

$$\mathbf{x}^{(k+1)} = D^{-1}(E + F)\mathbf{x}^{(k)} + D^{-1}b.$$

k	$\|\mathbf{x} - \mathbf{x}^{(k)}\|_2$	$\|\mathbf{r}^{(k)}\|_2$
0	3.187e-01	1.250e-01
10	3.825e-02	1.461e-02
20	4.595e-03	1.755e-03
30	5.519e-04	2.108e-04
40	6.628e-05	2.532e-05
50	7.961e-06	3.041e-06
60	9.562e-07	3.653e-07
70	1.149e-07	4.387e-08
80	1.379e-08	5.269e-09
90	1.657e-09	6.329e-10
100	1.990e-10	7.601e-10

Table 24.1: Solving the problem in Example 24.1 using Jacobi's method

24.4 Gauss-Seidel method

As we shall see, the Gauss-Seidel method is not very efficient. A better option (*see notes from class*!) would be the Gauss-Seidel method:

$$\mathbf{x}^{(k+1)} = (D - E)^{-1}F\mathbf{x}^{(k)} + (D - E)^{-1}b.$$

If we apply this technique to the problem in Example 24.1 we get the results in Table 24.2. We see that we achieve machine precision after about 80 iterations.

k	$\|\mathbf{x} - \mathbf{x}^{(k)}\|_2$	$\|\mathbf{r}^{(k)}\|_2$
0	3.187e-01	1.250e-01
10	5.385e-03	2.473e-03
20	7.769e-05	3.568e-05
30	1.121e-06	5.148e-07
40	1.617e-08	7.426e-09
50	2.333e-10	1.071e-10
60	3.365e-12	1.545e-12
70	4.856e-14	2.231e-14
80	6.790e-16	3.310e-16

Table 24.2: Solving the problem in Example 24.1 using the Gauss-Seidel method

25 Analysis of basic iterative methods

[Post-lecture version. This has minor changes from the one given out before class, including that equation numbers have been corrected.]

In Lecture 24 we introduced the Jacobi and Gauss-Seidel methods for solving linear systems iterativly. They work as follows: to find an approximate solution to the linear system

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

we write the matrix A as $A = D - E - F$, where D is a diagonal matrix, E is strictly lower triangular, and F is strictly upper triangular. We choose an initial guess $x^{(0)}$, and then set

Jacobi:

$$\boldsymbol{x}^{(k+1)} = D^{-1}(E + F)\boldsymbol{x}^{(k)} + D^{-1}b \tag{25.1a}$$

Gauss-Seidel:

$$\boldsymbol{x}^{(k+1)} = (D - E)^{-1}F\boldsymbol{x}^{(k)} + (D - E)^{-1}b. \tag{25.1b}$$

Under certain conditions, these iterations will *converge* in the sense that $\|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \to 0$ as $k \to \infty$. That is the topic we want to study today. For more details, see [Saa03, Chap 4]).

25.1 General iteration

The Jacobi, Gauss-Seidel and Successive Overrelaxation (see Exercise 25.1) methods can be written as schemes of the form

$$\boldsymbol{x}^{(k+1)} = G\boldsymbol{x}^{(k)} + f, \tag{25.2}$$

where $G_J = I - D^{-1}A$, and $G_{GS} = I - (D - E)^{-1}A$.

Another useful formulation is express A by the splitting *A=M-N*, and the iteration as

$$\boldsymbol{x}^{(k+1)} = M^{-1}N\boldsymbol{x}^{(k)} + M^{-1}\boldsymbol{b}. \tag{25.3}$$

For example, for Jacobi's method $M = D$, and Gauss-Seidel, $M = D - E$.

Both (25.2) and (25.3) have natural interpretations in the limiting case (i.e., as $k \to \infty$).

25.2 Convergence?

We have to ask three questions about our basic iterative methods:

(a) will they converge to the correct solution?

(b) (when) will they converge at all?

(c) how quickly do they converge?

It is easy to see that, if the Jacobi and Gauss-Seidel methods converge, they do so to the solution of $A\boldsymbol{x} = \boldsymbol{b}$.

The answer Question (b) we'll introduce some notation.

Definition 25.1. The *spectrum* of a matrix A is the set of its eigenvalues. We write it $\mu(A)$. The *spectral radius* of a matrix is the modulus of its largest eigenvalue:

$$\rho(A) = \max_{\lambda \in \mu(A)} |\lambda|.$$

Theorem 25.2 (Thm 1.10 of [Saa03])**.** *The sequence* G^k, $k = 0, 1, \ldots$ *converges to zero if and only if* $\rho(G) < 1$.

We'll be a little lazy and in class, only consider the case where the matrix G is *diagonalizable*. For the general case, we should study the *Jordan canonical form*.

We can now use this to show that the sequence generated by (25.2) converges providing that $\rho(G) < 1$.

Example 25.3. If the matrix A in the system $A\boldsymbol{x} = \boldsymbol{b}$ is strictly diagonally dominant, then the associated Jacobi iteration will converge.

In general, it is not so easy to work out $\rho(G)$ for a given matrix G. However, since $\rho(G) \leqslant \|G\|$ for any consistent matrix norm (see Exercise 26.2), then we can take $\|G\|$ is a sufficient (but not necessary) condition for convergence.

. . . . . .

We can now tell when one of our methods will converge. The next question we'll deal with is *how fast* convergence occurs.

25.3 Exercises

Exercise 25.1. Find out what the *Successive Overrelaxation* (SOR) method is. Show how to write it in the form of (25.1). Furthermore, find the formula for M and N if SOR it expressed in the form (25.3).

Exercise 25.2. The on-line notes for Lecture 24 are accompanied by a Matlab programme that generated Tables 24.1 and 24.2. Write a similar Matlab programme that implements SOR.

26 Convergence

(*These notes are somewhat different from the version handed out in class – even the title changed!*

26.1 Recap: convergence

Recall that we saw that the sequence

$$\{G^0, G^1, \dots, G^k, \dots\},$$

will converge to zero if $\rho(G) < 1$.

Recall that we can write iterative schemes, such as Jacobi or Gauss-Seidel as

$$\boldsymbol{x}^{(k+1)} = G\boldsymbol{x}^{(k)} + f,$$

If $A\boldsymbol{x} = \mathbf{b}$, then the sequence generated satisfies

$$\boldsymbol{x} - \boldsymbol{x}^{(k)} = G^k(x - x^{(0)}).$$

So, if $\rho(G) < 1$, the iteration will converge (to the solution of $A\boldsymbol{x} = \mathbf{b}$) for any initial guess $\boldsymbol{x}^{(0)}$.

Example 26.1. If the matrix A in the system $A\boldsymbol{x} = \mathbf{b}$ is strictly diagonally dominant, then the associated Jacobi iteration will converge.

In general, it is not so easy to work out $\rho(G)$ for a given matrix G. However, since $\rho(G) \leqslant \|G\|$ for any consistent matrix norm (see Exercise 26.2), then we can take $\|G\|$ is a sufficient (but not necessary) condition for convergence.

26.2 The Jordan canonical form

(I'd thought I'd gloss over this topic, but have realised that it is too useful). In our "proof" of 25.2, we only considered a special case. For the general case, we need the notion of the *Jordan canonical form*.

Recall that matrices A and B are *similar* if there exists an invertible matrix X such that $A = XBX^{-1}$. It is clear that A and B have the same eigenvalues.

Definition 26.2.

1. An eigenvalue λ of A has *algebraic multiplicity* μ if it is a root of multiplicity μ of the characteristic polynomial of A.
2. If λ has multiplicity 1 it is *simple*.
3. The *geometric multiplicity*, γ, of an eigenvalue λ is the number of linearly independent eigenvalues of A.
4. An eigenvalue is *semisimple* if its geometric multiplicity is equal to its algebraic multiplicity. If it is not semisimple, it is *degenerate*.

Definition 26.3 (Jordan Canonical Form)**.** For any matrix A there are matrices X and J such that $A = X^{-1}JX$, and J has the form

$$J = \begin{pmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_p \end{pmatrix}.$$

where the J_i are block each associated with a distinct eigenvalue λ_i of A. Each of these blocks can be expressed as γ_i "Jordan" blocks where γ_i is the geometric multiplicity of λ_i. These blocks are

$$J_i = \begin{pmatrix} J_{i1} & & & \\ & J_{i2} & & \\ & & \ddots & \\ & & & J_{i\gamma_i} \end{pmatrix}.$$

where J_{ik} is the bidiagonal matrix

$$J_{ik} = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & 1 & \\ & & \ddots & \\ & & & \lambda_i \end{pmatrix}$$

26.3 Exercises

Exercise 26.1. Show that, if A is diagonally dominant, then the Gauss-Seidel iteration converges.

Exercise 26.2. Show that, if $\|\cdot\|$ is a consistent matrix norm, then $\rho(G) \leqslant \|G\|$.

27 The JCF, and applications

27.1 Recall: defective matrices

In Lecture 26 we saw that the *algebraic multiplicity* of an eigenvalues λ, is the multiplicity of the associated root in the characteristic polynomial. We denote this $\mu(\lambda)$. This eigenvalue's *geometric multiplicity*, $\gamma(\lambda)$ is the number of linearly independent eigenvectors associated with it. Always, $\gamma(\lambda) \leqslant \mu(\lambda)$ (see Exercise 27.1). If $\gamma(\lambda) < \mu(\lambda)$, the eigenvalue is *degenerate*. Moreover, the matrix is *defective*: that is, it does not have a full set of linearly independent eigenvectors.

If a matrix, $A \in \mathbb{R}^{m\times m}$ is *not* defective, then we say it is *diagonalisable*. This means that, since it has m linearly independent eigenvectors, $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, we can form the matrix $X = (\mathbf{x}_1|\cdots|\mathbf{x}_m)$, and thus write

$$A = X^{-1}\Lambda X,$$

where Λ is a diagonal matrices whose entries are the eigenvalues of A

As we saw in class, this diagonalisation – when it exists – is a useful theoretical tool. But often, of course, it does not exist.

27.2 A degenerate matrix

The simplest, and most important, example of a degenerate matrix is

$$A = \begin{pmatrix} \alpha & 1 \\ 0 & \alpha \end{pmatrix}$$

It is easy to see that its only eigenvalue is α, but that $\mu(\alpha) = 2$ and $\gamma(\alpha) = 1$. Similarly, the matrix

$$A = \begin{pmatrix} \alpha & 1 & 0 & 0 \\ 0 & \alpha & 1 & 0 \\ 0 & 0 & \alpha & 0 \\ 0 & 0 & 0 & \beta \end{pmatrix}$$

is degenerate. However, it is an easy matrix to work with: for example we can tell just by looking at it what the eigenvalues are, and what their multiplicities are.

27.3 Recall: The Jordan canonical form

The JCF is a similarity transform that reduces a matrix to a block-diagonal one. I don't reproduce it here: you can see it in the notes from Lecture 26. But we can not get a sense of what it means.

- Each of the bidiagonal matrices J_{ik} is associated with a single eigenvector of the matrix.
- Each of the matrices J_i is associated with a single eigenvalue, λ_i; and J_i has $\mu(\lambda_i)$ rows and columns.

Every matrix (even degenerate ones), can be written in the form $A = X^{-1}JX$. This has many applications, e.g., we can see immediately that if a matrix has distinct eigenvalues, then it is diagonalisable.

27.4 General convergence

Let $\mathbf{d}^{(k)} = \mathbf{x} - \mathbf{x}^{(k)}$.

Definition 27.1. The *General convergence factor* is

$$\phi = \lim_{k\to\infty} \left(\max_{x_0\in\mathbb{R}^m} \frac{\|\mathbf{d}^{(k)}\|}{\|\mathbf{d}^{(0)}\|} \right)^{1/k}.$$

We can show that $\phi = \rho(G)$. This makes use of Exercise 27.2.

27.5 Exercises

Exercise 27.1. It is a fact (which can be established with some tedium from the usual (Laplace) expansion for the determinant) that if the matrix M is partitioned as

$$M = \begin{pmatrix} A & B \\ 0 & D \end{pmatrix},$$

then $\det(M) = \det(A)\det(D)$. Use this to show that the geometric multiplicity of an eigenvalue is less than or equal to its algebraic multiplicity.

Exercise 27.2. (Tricky.) Use the Jordan canonical form to show that, for any subordinate matrix norm,

$$\lim_{k\to\infty} \|A^k\|^{1/k} = \rho(A).$$

28 Regular splittings

28.1 Preliminary results

The next topic we want to look at is the notion of a *regular splitting*. But before then, we need some basic ideas from linear algebra.

Recall that $\rho(A)$ is the spectral radius of the matrix A.

Theorem 28.1. *If $\rho(A) < 1$, then the matrix $B = I - A$ is nonsingular.*

Theorem 28.2 (Theorem 1.11 of [Saa03])**.** *The series*

$$\sum_{k=0}^{\infty} A^k$$

converges if and only if $\rho(A) < 1$, in which case $I - A$ is nonsingular, and

$$(I - A)^{-1} = \lim_{p\to\infty} \sum_{k=0}^{p} A^k$$

28.2 Nonnegative matrices

(See [Saa03, Section 1.10]).

Definition 28.3. If the matrices A and B are of the same size, when we write $A < B$, we mean $a_{ij} < b_{ij}$ for each i, j. Similarly,

$$A \leqslant B \quad \Longleftrightarrow \quad a_{ij} \leqslant b_{ij} \ \forall i, j.$$

Use O to designate the zero matrix. When we say "*the matrix A is nonnegative*", we mean $O \leqslant A$. When we say "*the matrix A is positive*", we mean $O < A$.

Definition 28.4. A matrix A is *reducible* is there exists a permutation matrix P such that PAP^T is block upper triangular.

A more intuitive, equivalent definition, is that the matrix A is irreducible if its graph is strongly connected.

The next result is classic, though we won't prove it.

Theorem 28.5 (Perron-Frobenius)**.** *If the matrix A is nonnegative, then $\lambda = \rho(A)$ is a eigenvalue of A with $\mu(\lambda) = 1$. There exists an associated eigenvector u such that $u \geqslant 0$.*

If, furthermore, A is irreducible, then $u > 0$.

Other important properties of nonnegative matrices include:

Theorem 28.6. *Let A, B and C be nonnegative matrices, with $A \leq B$. Then*

$$AC \leqslant BC \quad \text{and} \quad CA \leqslant CB.$$

Furthermore, $A^k \leqslant B^k$ for $k = 0, 1, \ldots$.

The proof is left as Exercise 28.1.

Theorem 28.7. *Suppose that $B \geqslant 0$. Then $\rho(B) < 1$ if and only if $I - B$ is nonsingular and $(I - B)^{-1} \geqslant 0$.*

28.3 Regular splitting

Definition 28.8. The pair of matrices M and N is a *regular splitting* of the matrix A if

(a) $A = M - N$.

(b) M^{-1} exists, and

(c) both M^{-1} and N are nonnegative.

Now define the iteration

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}\mathbf{b}. \tag{28.1}$$

We want to know when this will converge. We can use Theorem 28.2 to prove the following

Theorem 28.9 (Thm 4.4 of [Saa03])**.** *Let M, N be a regular splitting of A. Then $\rho(M^{-1}N) < 1$ if and only if A is nonsingular and A^{-1} is nonnegative.*

So we now can establish when an iteration of the form (28.1) converges. These apply directly to the Jacobi and Gauss-Seidel methods. See Exercise 28.2

28.4 Exercises

Exercise 28.1. Prove Theorem 28.6.

Exercise 28.2. Recall from (25.1) that the Jacobi method can be written as

$$\mathbf{x}^{(k+1)} = D^{-1}(E + F)\mathbf{x}^{(k)} + D^{-1}\mathbf{b},$$

and the Gauss-Seidel as

$$\mathbf{x}^{(k+1)} = (D - E)^{-1}F\mathbf{x}^{(k)} + (D - E)^{-1}\mathbf{b}.$$

When conditions do you need on A for these to be regular splittings?

29 Non-stationary methods

29.1 Motivation, via preconditioners

In recent lectures, we studied the Gauss-Seidel and Jacobi (and, in an exercises, SOR) methods for solving linear systems of the form

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

We call a matrix M a *preconditioner* for A if $M^{-1}A \approx I$. Then, instead of solving $A\boldsymbol{x} = \mathbf{b}$, we solve

$$M^{-1}A\boldsymbol{x} = M^{-1}\mathbf{b}.$$

If indeed $M^{-1}A \approx I$, this should be a relatively easy task. For the Jacobi method, we took $M = \mathrm{diag}(A)$, and then we iterated as

$$x^{(k+1)} = M^{-1}(M - A)x^{(k)} + M^{-1}\mathbf{b}.$$

This can be rearranged as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + M^{-1}(\mathbf{b} - A\boldsymbol{x}^{(k)}). \tag{29.1}$$

Several observations come from this

(a) Recall that the *residual* is $\mathbf{r}^{(k)} = \mathbf{b} - A\boldsymbol{x}^{(k)}$. So we can write (29.1) as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + M^{-1}\mathbf{r}^{(k)}. \tag{29.2}$$

(b) Suppose that, in fact, $\boldsymbol{x}^{(k)} = \boldsymbol{x}$. Since the iteration should always give that $\|\boldsymbol{x} - x^{(k+1)}\| \leqslant \|\boldsymbol{x} - x^{(k)}\|$, then we need $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. This is achieved when $\|\mathbf{r}^{(k)}\| = 0$. Since this is not achievable (realistically), when instead want to minimise $\|M^{-1}\mathbf{r}^{(k)}\|$.

(c) Different methods give different choices of M. Why not allow the freedom for this value to change at each iteration, in order to maximise the error reduction at each step?

The topic of choosing a preconditioner, M, is a deep one, and not really part of this course (it will be part of the follow-on workshop in May). But here we use it as a motivation.

29.2 Orthomin(1)

See [Gre97, Chap. 2]. As a variation on (29.1), let's consider methods of the form

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k(\mathbf{b} - A\boldsymbol{x}^{(k)}), \tag{29.3}$$

and where the goal is to choose the scalar α_k in the best way possible. As discussed above, this could be by trying to minimise the 2-norm of the residual, $\mathbf{r}^{(k+1)}$. With a little work (take notes!) this can be done by picking

$$\alpha_k = \frac{(\mathbf{r}^{(k)}, A\mathbf{r}^{(k)})}{(A\mathbf{r}^{(k)}, A\mathbf{r}^{(k)})}. \tag{29.4}$$

This method is known (among other things) as *Orthomin(1).*

29.3 Steepest Descent

Suppose that A is s.p.d. Then $\|\boldsymbol{x}\|_A := \sqrt{\boldsymbol{x}^T A\boldsymbol{x}}$ is a norm. We could choose α_k in (29.3) to minimise $\boldsymbol{e}^{(k+1)} = \boldsymbol{x} - \boldsymbol{x}^{(k)}$ in this norm. This leads to

$$\alpha_k = \frac{(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{r}^{(k)}, A\mathbf{r}^{(k)})}, \tag{29.5}$$

which is called the method of *steepest descent.* See Exercise 29.1

29.4 Exercises

Exercise 29.1. Suppose that we want to choose α_k in (29.3) to minimise $\|\boldsymbol{e}^{(k+1)}\|_A$. Recalling that $\boldsymbol{e}^{(k+1)} = \boldsymbol{e}^{(k)} - \alpha_k\mathbf{r}^{(k)}$, show that this leads to the formula in (29.5). *Hint: see [Gre97, §1.3.2].*

30 Convergence of Orthomin(1)

30.1 Numerical range

(See [Gre97, §1.3.6]).

Definition 30.1 (Numerical range and radius)**.** The numerical range (a.k.a., *field of values*) of a matrix $A \in \mathbb{C}^{m\times m}$ is the set of complex numbers

$$\mathcal{F}(A) = \left\{ \frac{y^\star A y}{y^\star y} \ : \ y \in \mathbb{C}^m, y \neq 0 \right\}.$$

The *numerical radius* $\eta(A)$, is defined as

$$\eta(A) := \max\{|z| \ : \ z \in \mathcal{F}(A)\}.$$

Properties of the numerical range include that

$$\mathcal{F}(A + \alpha I) = \mathcal{F}(A) + \alpha, \quad \text{and} \quad \mathcal{F}(\alpha A) = \alpha \mathcal{F}(A).$$

30.2 Orthomin(1), again

Recall that the Orthomin(1) method (29.3)+(29.4) comes from setting

$$x^{(k+1)} = x^{(k)} + a_k(b - Ax^{(k)}), \tag{30.1a}$$

where

$$a_k = \frac{(r^{(k)}, Ar^{(k)})}{(Ar^{(k)}, Ar^{(k)})}. \tag{30.1b}$$

So (see [Gre97, §1.3.2]), since

$$r^{(k+1)} = r^{(k)} - a_k A r^{(k)},$$

we can think of the method as setting $r^{(k+1)}$ as $r^{(k)}$ minus its projection on to $Ar^{(k)}$.

Theorem 30.2 (Thm. 2.2.1 of [Gre97])**.** *If Orthomin(1) method* (30.1) *generates the sequence* $\{x^{(0)}, x^{(1)}, \dots\}$*, and* $r^{(0)}$*,* $r^{(1)}$*, ... are the associated residuals, then* $\|r^{(k+1)}\|_2 < \|r^{(k)}\|_2$ *for any* $x^{(0)}$*, if and only if* $0 \notin \mathcal{F}(A)$*.*

However, in practice, it is not enough to know that the residual decreases from step k to step $k+1$, we need to know by how much it decreases. This is given by:

Theorem 30.3 (Thm. 2.2.2 of [Gre97])**.** *The iteration* (30.1) *converges to* $A^{-1}b$ *for any initial guess* $x^{(0)}$ *if* $0 \notin \mathcal{F}(A)$*, and*

$$\|r^{(k+1)}\|_2 \leqslant \|r^{(k)}\| \sqrt{1 - d^2/\|A\|_2^2},$$

where d *is the distance from the origin to* $\mathcal{F}(A)$*.*

Note: as observed during class (thanks, Olga!) the proof discussed was not complete: we also need that $d \leqslant \|A\|$*. This was addressed at the start of Lecture 31.*

30.3 Exercises

Exercise 30.1. Recall Definition 30.1.

(a) Show that if $A \in \mathbb{R}^{m\times m}$, then $\mathcal{F}(A) \subset \mathbb{R}$.

(b) Show that if $A \in \mathbb{C}^{m\times m}$, and $A^\star = A$, then $\mathcal{F}(A) \subset \mathbb{R}$.

Exercise 30.2. (a) Show that if λ is an eigenvalue of A, then $\lambda \in \mathcal{F}(A)$.

(b) Suppose that Q is unitary. Show that $\mathcal{F}(Q^\star A Q) = \mathcal{F}(A)$.

31 Orthomin(2)

Sorry: no typed notes for this lecture at this time...

32 Conjugate Gradient Method (CG)

(*These notes where heavily revised after the class*).
In the final section of this course, we will look at one of the most popular iterative methods for solving linear systems of equations when the system matrix is s.p.d.

32.1 Example

The CG method is more complicated to state and analyse than other iterative methods we have studied, such has the Gauss-Seidel method. So, even before we state the method, we begin by demonstrating that it is worth the effort: it is *much* faster than Gauss-Seidel.

Suppose we apply the Gauss-Seidel method and CG to the linear system that arises from solving the Laplace equation on a 16×16 grid. So A has 256 rows and columns. In Figure 32.1 below we show the residual reduction for both methods. It is clear that CG is converging *much* faster that Gauss-Seidel.

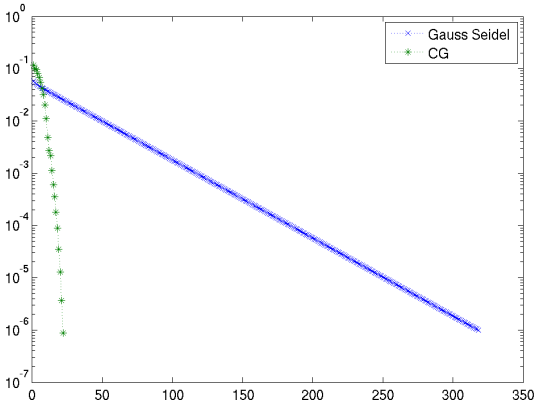


Figure 32.1: Gauss-Seidel and CG residuals

Looking at just the residual reduction in CG, as shown in Figure 32.2, one could be convinced that, unlike Gauss-Seidel, we might actually achieve the exact solution in a finite number of steps.

32.2 Conjugates

We say that two vectors, $\mathbf{x}$ and $\mathbf{y}$, are *conjugate*, with respect to the matrix A, if $x^T A y = 0$. When A is s.p.d., the bilinear form

$$(\mathbf{x}, \mathbf{y})_A := (A\mathbf{x}, \mathbf{y}) = x^T A y,$$

is an inner product. So, being *conjugate with respect to* A is the same as being orthogonal with respect to the inner product $(\cdot,\cdot)_A$. So the expression "A-orthogonal" is also used for conjugate w.r.t. A.

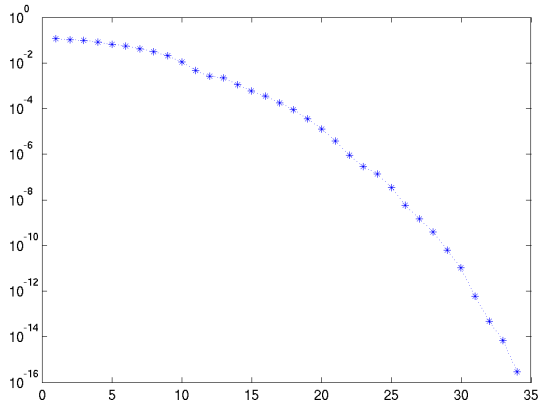


Figure 32.2: CG residuals

32.3 The CG algorithm

Choose , $\mathbf{x}^{(0)}$. Set $\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$ and $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$. For $k = 1, 2, \ldots,$ compute: $A\mathbf{p}^{(k-1)}$ and then set

$$a_{k-1} = \frac{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}{(\mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)})}. \tag{32.1}$$

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + a_{k-1}\mathbf{p}^{(k-1)}. \tag{32.2}$$

$$\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - a_{k-1}A\mathbf{p}^{(k-1)} \tag{32.3}$$

$$b_{k-1} = \frac{(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}. \tag{32.4}$$

$$\mathbf{p}^{(k)} = \mathbf{r}^{(k)} + b_{k-1}\mathbf{p}^{(k-1)} \tag{32.5}$$

Important: There are other possible expressions for a_k and b_k. See Exercise 32.1.

32.4 Analysis (Initial ideas)

Usually, we define the residual, $\mathbf{r}^{(k)}$ as $\mathbf{r}^{(k)} := b - Ax^{(k)}$. However, we seem to be using a different definition in (32.3). So we spent a new minutes in class, showing that these formulations are equivalent.

Next we observed that the method is well-defined. It would only fail to be well-defined in the case there the denominator in (32.1) or (32.4) was zero. However, since A is s.p.d., this can only happen if $\mathbf{r}^{(k-1)} = 0$, in which case $\mathbf{x} = \mathbf{x}^{(k+1)}$, and we are done.

32.5 Exercises

Exercise 32.1. Show that the following expressions are equivalent to those in (32.1) and (32.4)

$$a_{k-1} = \frac{(\mathbf{r}^{(k-1)}, \mathbf{p}^{(k-1)})}{(\mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)})}, \tag{32.6}$$

$$b_{k-1} = -\frac{(\mathbf{r}^{(k)}, A\mathbf{p}^{(k-1)})}{(\mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)})} \tag{32.7}$$

Exercise 32.2. Write a Matlab implementation of the CG algorithm. Use it to generate the diagrams in Figure 32.1.

33 Analysis of CG

(**DRAFT**)

We'll devote this class to proving the following result. Here, by "iterates" we mean the $\boldsymbol{x}^{(k)}$; the errors are $\boldsymbol{e}^{(k)} = A^{-1}b - \boldsymbol{x}^{(k)}$; and the residuals are $\mathbf{r}^{(k)} = b - A\boldsymbol{x}^{(k)}$. Note that $\boldsymbol{e}^{(k)} = A^{-1}\mathbf{r}^{(k)}$.

Theorem 33.1 (Thm 2.3.2 of [Gre97])**.** *Suppose that A is s.p.d. and the CG algorithm generates the sequence $\{\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \dots\}$. Then*

(a) The iterates, errors and residuals are all well-defined.

(b) The vectors $\mathbf{r}^{(k)}$, $\boldsymbol{e}^{(k)}$ and $\mathbf{p}^{(k)}$ satisfy

$$(\mathbf{r}^{(k+1)}, \mathbf{r}^{(j)}) = 0, \tag{33.1}$$

$$(\boldsymbol{e}^{(k+1)}, A\mathbf{p}^{(j)}) = 0, \tag{33.2}$$

$$(\mathbf{p}^{(k+1)}, A\mathbf{p}^{(j)}) = 0, \tag{33.3}$$

for all $j \leqslant k$.

(c) CG generates the exact solution after m iterations.

(d) Of all the vectors in the affine space

$$\boldsymbol{e}^{(0)} + \operatorname{span}\{A\boldsymbol{e}^{(0)}, A^2\boldsymbol{e}^{(0)}, ; A^{k+1}\boldsymbol{e}^{(0)}\},$$

$\boldsymbol{e}^{(k+1)}$ has the smallest A-norm.

34 Krylov subspaces, and the optimality of CG

(**DRAFT**: in particular, more exercises will be added in time.)

In Theorem 33.1 we proved some important properties of the CG algorithm. We didn't get to Part (d). Although it easily follows from Parts (a)–(c), we are going to take a different approach.

34.1 Krylov subspaces

(Parts of this presentation are borrowed from Wikipedia!)

Given $\mathbf{b}$, a matrix A and vector $\mathbf{b}$, the *Krylov subspace* generated by A and $\mathbf{b}$ is

$$\mathcal{K}_n = \operatorname{span}\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \dots, A^{n-1}\mathbf{b}\}. \tag{34.1}$$

It's importance originates from the fact that, due the Cayley-Hamilton Theorem, one can express the inverse of a matrix as a linear combinations of its powers. The Cayley-Hamilton theorem states that, if $p(\lambda)$ is the characteristic polynomial of A, i.e., $p(\lambda) = \det(\lambda I - A)$, then $p(A) = 0$.

To be clear, we don't mean that we substitute A for λ in the expression $\lambda I - A$. We mean we write out the polynomial

$$p(\lambda) = \lambda^m + c_{m-1}\lambda^{m-1} + \cdots + c_1\lambda + (-1)^m \det(A).$$

Then we substitute A for λ in the above polynomial. Then $p(A) = 0$ can be rearranged to get:

$$A(A^{m-1} + c_{m-1}A^{m-2} + \cdots + c_1) = -(-1)^m \det(A) I.$$

Dividing by $(-1)^{m-1}\det(A)$ gives a formula for A^{-1}. This means, that the solution, $\mathbf{x}$, to $A\mathbf{x} = \mathbf{b}$ can be written as a linear combination of the vectors in $\mathcal{K}_m$.

34.2 CG, with $\mathbf{x}^{(0)} = \mathbf{0}$

If we initiate the CG algorithm with $\mathbf{x}^{(0)} = \mathbf{0}$. So the algorithm becomes: set $\mathbf{r}^{(0)} = \mathbf{p}^{(0)} = \mathbf{b}$.
For $k = 1, 2, \dots,$ set

$$\begin{aligned}
\mathbf{x}^{(k)} &= \mathbf{x}^{(k-1)} + a_{k-1}\mathbf{p}^{(k-1)}.\\
\mathbf{r}^{(k)} &= \mathbf{r}^{(k-1)} - a_{k-1}A\mathbf{p}^{(k-1)}\\
\mathbf{p}^{(k)} &= \mathbf{r}^{(k)} + b_{k-1}\mathbf{p}^{(k-1)}
\end{aligned}$$

where

$$a_{k-1} = \frac{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}{(\mathbf{p}^{(k-1)}, A\mathbf{p}^{(k-1)})}, \quad b_{k-1} = \frac{(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)})}.$$

Clearly, $\mathbf{r}^{(0)} \in \mathcal{K}_0$, and $\mathbf{p}^{(0)} \in K_0$. Also, since $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + a_0\mathbf{p}^{(0)}$, and a_0 is just a scalar, we get that $\mathbf{x}^{(1)} \in \mathcal{K}_0$ too.

Next, using that $\mathbf{r}^{(1)} = \mathbf{r}^{(0)} - a_0 A\mathbf{p}^{(0)}$, we see that $\mathbf{r}^{(1)} \in \mathcal{K}_1$. Proceeding inductively, we get that

$$\begin{aligned}
\mathcal{K}_n &= \operatorname{span}\{\mathbf{x}^{(1)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}\\
&= \operatorname{span}\{\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, \dots, \mathbf{r}^{(n-1)}\}\\
&= \operatorname{span}\{\mathbf{p}^{(0)}, \mathbf{p}^{(1)}, \dots, \mathbf{p}^{(n-1)}\}.
\end{aligned}$$

Note that, for example, $\mathbf{r}^{(n)}$ is orthogonal to $\mathcal{K}_n$.

34.3 Optimality of CG

The presentation of this section is borrowed from [TB97, Lecture 38].

Theorem 34.1 (Thm. 38.2 of [TB97])**.** *Suppose that A is s.p.d. and the CG algorithm generates the sequence $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots\}$. If $\mathbf{r}^{(n-1)} \neq 0$, then $\mathbf{x}^{(n)}$ is the unique vector $\mathcal{K}_n$ for which $\|\mathbf{e}^{(n)}\|_A$ is minimised. Furthermore, $\|\mathbf{e}^{(n)}\|_A \leqslant \|\mathbf{e}^{(n-1)}\|_A$.*

34.4 Exercises

Exercise 34.1. Show that any $x \in \mathcal{K}_n$ can be expressed as $p(A)\mathbf{b}$, where p is some polynomial of degree at most $n - 1$.

THE END.

References

[Dav06] Timothy A. Davis. *Direct methods for sparse linear systems*, volume 2 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006.

[Dem97] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[Gre97] Anne Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[Ips09] Ilse C. F. Ipsen. *Numerical matrix analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

[Saa03] Yousef Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2003.

[Ste98] G. W. Stewart. *Afternotes goes to graduate school*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[TB97] Lloyd N. Trefethen and David Bau, III. *Numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.