MA2715 Advanced Calculus and Numerical Methods

Lecture Notes by M.K. Warby in 2019/0
Dept. of Mathematics, Brunel Univ., UK

Email: Mike.Warby@brunel.ac.uk
URL: http://people.brunel.ac.uk/~icstmkw/ma2715/

MA2715 and MA2895 – the week breakdown

The lectures MA2715 run for weeks 17-27 with seminars starting in week 18.

The related assessment block MA2895 will run with teaching events for weeks 17–27 involving computer labs in which Matlab will be used.

Assessment dates and assessment information

Some of the assessment is under the code MA2895_CB, which is a 10 credit module, which has the title "Numerical Analysis Project" and for this the breakdown is as follows:

- Class test worth 30%: Planned for week 22.
- Assignment worth 70%: Deadline is likely to be the start of week 28. The assignment tasks are likely to be given out in the next three to five weeks with some of the topics related to the material of MA2715.

We will confirm later as to the precise arrangement of the test planned for week 22 and also the deadline for the assignment.

There will also be questions on the material taught on all parts of MA2715_SB in the 20 credits assessment block MA2815_CN which has the title "Advanced Mathematics II and Numerical Methods". This exam will be in the April/May 2020 exam period.

Recommended reading and my sources

There is no essential text to obtain for this module although there are many texts which cover at least most of the material and among the sources for the notes that I will generate are the following books.

1. Kendall E. Atkinson. *Elementary numerical analysis.* Wiley, 3rd edition, 2004.
   `QA297.A83`
2. Richard L. Burden and J. Douglas Faires. *Numerical analysis.* Brooks/Cole, 7th edition, 2001.
   `QA297.B87`. The Brunel library has later editions with the 9th edition being published in 2011.
3. Glyn James. *Advanced Modern Engineering Mathematics.* Pearson Prentice Hall, Fourth edition, 2010.
   `TA330.A48`. ISBN: 9780130454256.

   (Chapter 7 of this book contains material about Fourier series which is planned to be the last chapter covered in `MA2715`.)

Books about Matlab related to `MA2895`

The following is repeated in the handouts for the Matlab sessions and for completeness I include it here as well. There is no core Matlab book that you have to buy for the module although I do myself own several books and among those that I consult sometimes are the following.

1. Timothy A. Davis. *MATLAB primer.* CRC Press, eighth edition, 2011.
   `QA297.D38`.
2. Brian D. Hahn and D. T. D. T. Valentine. *Essential MATLAB for engineers and scientists.* Academic Press, 5th edition, 2013.
   `QA297.H345`.
3. D. J. Higham and Nicholas J. Higham. *MATLAB guide.* Society for Industrial and Applied Mathematics, 2nd edition, 2005.
   `QA297.H525`

   (I also have the 3rd edition published in Feb 2017 but this is not yet in the Brunel library.)

Please note that Matlab itself has an extensive help system.

Chapter 1

Vectors and matrices: notation, revision and norms

1.1 Notation

In this module the following notation will be used.

$\mathbb{R}$ denotes the set of real numbers.

$\mathbb{R}^n$ denotes the set of vectors of length n with real entries.

$\mathbb{R}^{m,n}$ denotes the set of matrices with m rows and n columns with real entries.

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

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

If $\underline{x}$ is a column vector then $\underline{x}^T = (x_1, \dots, x_n)$ is a **row vector**. Here the superscript T denotes transpose. Column vectors and row vectors are special cases of a matrix and in the notation used we have reduced how much we write by just using 1 subscript for the entries. When both $m > 1$ and $n > 1$ we have a **matrix** and we use the notation $A = (a_{ij}) \in \mathbb{R}^{m,n}$ as shorthand for a real matrix with m rows and n columns with the i, j entry being a_{ij}, i.e.

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

When we refer to the size of a matrix the terms m-by-n or $m \times n$ will sometimes be used. In this module we mostly just consider the case when $m = n$, i.e. we mostly just consider square matrices.

Other notation which will frequently be used are the following.

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

The $n \times n$ **identity matrix** is denoted by

$$I = (\delta_{ij}) = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix},$$

where

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

The identity matrix is a square diagonal matrix with each diagonal entry being equal to 1. When just I is written the size should be clear from the context but if the size is not immediately clear then we will write I_n when the size is $n \times n$.

The **standard base vectors** for $\mathbb{R}^n$ are denoted by $\underline{e}_1, \ldots, \underline{e}_n$ which correspond respectively to the columns of I. Thus when $n = 3$ we have

$$I = I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \underline{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \underline{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \underline{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

1.2 The vector space $\mathbb{R}^n$

$\mathbb{R}^n$ is an example of a vector space and the vectors $\underline{e}_1, \ldots, \underline{e}_n$ give the standard basis for this space and further these base vectors are orthonormal. (Vector spaces were discussed at the start of level 2 as part of the module `MA2721`.) We consider next the definitions of these terms using the column vector notation used in this module.

The **inner product** of vectors $\underline{x}$ and $\underline{y}$ is defined by

$$\underline{x}^T \underline{y} = (x_1, x_2, \ldots, x_n) \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

In books and other sources you may see this operation written as $(\underline{x}, \underline{y})$ or $< \underline{x}, \underline{y} >$ or as $\underline{x} \cdot \underline{y}$ but in these notes the matrix product notation of a row vector times a column vector will be used.

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

$$\underline{x}^T \underline{y} = 0.$$

A vector $\underline{x}$ has **unit length** if

$$\underline{x}^T \underline{x} = x_1^2 + \cdots + x_n^2 = 1.$$

(As we will see in section 1.5, for vectors $\underline{x}$ of any length the quantity $\underline{x}^T\underline{x}$ is also referred to as the square of the 2-norm of $\underline{x}$.)

Vectors $\underline{v}_1, \ldots, \underline{v}_n$ are **orthonormal** if the vectors each have unit length and they are orthogonal to each other, i.e.

$$\underline{v}_i^T\underline{v}_i = 1, \quad i = 1, \ldots, n \qquad \text{and} \quad \underline{v}_i^T\underline{v}_j = 0, \text{ when } i \neq j.$$

Vectors $\underline{v}_1, \ldots, \underline{v}_n$ are **linearly independent** if

$$\alpha_1\underline{v}_1 + \cdots + \alpha_n\underline{v}_n = \underline{0}$$

is only true when $\alpha_1 = \cdots = \alpha_n = 0$. It follows almost immediately that if vectors are orthonormal then they are linearly independent. The converse of linearly independent is linearly dependent. Vectors $\underline{v}_1, \ldots, \underline{v}_n$ are **linearly dependent** if there exists $\underline{\alpha} = (\alpha_i) \neq \underline{0}$ such that

$$\alpha_1\underline{v}_1 + \cdots + \alpha_n\underline{v}_n = \underline{0}.$$

1.3 The vector $A\underline{x}$ and the solution of $A\underline{x} = \underline{b}$

In the context of things done in MA2715 note that when we have a $n \times n$ matrix A and a $n \times 1$ column vector $\underline{x}$ the product gives another column vector, i.e.

$$\underline{b} = A\underline{x} \in \mathbb{R}^n.$$

This can be represented in a number of ways. With $\underline{b} = (b_i)$ the matrix multiplication means that

$$b_i = (i\text{th row of } A)\underline{x} = \sum_{j=1}^{n} a_{ij}x_j, \quad i = 1, \ldots, n.$$

Instead of considering things entry-by-entry we can consider representing A column -by-column, i.e.

$$A = (\underline{a}_1, \ldots, \underline{a}_n) \quad \text{where } \underline{a}_j = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{nj} \end{pmatrix} = j\text{th column of } A.$$

It is valid here to write

$$\begin{aligned} A\underline{x} &= (\underline{a}_1, \ldots, \underline{a}_n)\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \\ &= \underline{a}_1x_1 + \cdots + \underline{a}_nx_n \\ &= x_1\underline{a}_1 + \cdots + x_n\underline{a}_n. \end{aligned}$$

Provided the matrix sizes are compatible matrix multiplication works in "block form" to explain how the second line follows from the first line. The final line follows because $\underline{a}_j\,x_j = x_j\underline{a}_j$ as a consequence of what we mean by a scalar times a vector. The point of writing the expression in this form is that $A\underline{x}$ is a linear combination of the n columns of A. When the columns of A are linearly dependent there is a vector $\underline{x} \neq \underline{0}$ such that

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

and this vector is an eigenvector of A with 0 as the eigenvalue. When the columns of A are linearly dependent the matrix is singular, the determinant $\det A = 0$, the matrix does not have an inverse matrix (i.e. it is not invertible) and we cannot uniquely solve a linear system

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

In chapter 2 we consider how computer packages solve $A\underline{x} = \underline{b}$ and we hence need to determine when there is a unique solution and when there is not, i.e. as part of the computation we need to be able to decide if the matrix A is a singular matrix or not. In your previous study you will have done many examples with small matrices (e.g. $n = 2$ or $n = 3$) where you attempt to obtain a solution or you detect that there is no unique solution and you possibly also determine a general solution if it exists. For larger matrices with computations done on a computer with floating point arithmetic used (which usually has rounding errors) we cannot generally be so precise in deciding that a matrix is singular but instead be can just determine that a matrix is "close to being singular". This is described briefly in section 1.5 when vector and matrix norms and the matrix condition number is introduced.

1.4 Eigenvalues and eigenvectors

Eigenvalues and eigenvectors will appear a few times in this module and we give here briefly a revision of the definition of these terms together with some of the basic properties.

Let A denote a real $n \times n$ matrix. A vector $\underline{v} \neq \underline{0}$ is an **eigenvector** of A with **eigenvalue** λ if

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

This can be equivalently written as

$$A\underline{v} - \lambda\underline{v} = \underline{0} \quad \text{or} \quad (A - \lambda I)\underline{v} = \underline{0}.$$

As $\underline{v} \neq \underline{0}$ this means that we have a non-trivial solution of $(A - \lambda I)\underline{v} = \underline{0}$ and this implies that $A - \lambda I$ is singular and its determinant is 0. If we let

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

then this defines the **characteristic polynomial** of A, and the eigenvalues are the roots of this polynomial. As a minor point, the characteristic polynomial is often defined as $\det(tI - A)$, e.g. this is the case in Matlab, which only differs from the previous definition by a factor of -1 when n is odd and both versions have the same roots which are the solutions of

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

which is known as the **characteristic equation**. There is a result known as the **fundamental theorem of algebra** which states that such a polynomial has n roots $\lambda_1, \ldots, \lambda_n$ in the complex plane such that

$$P_A(t) = (\lambda_1 - t)(\lambda_2 - t)\cdots(\lambda_n - t)$$

which is a polynomial of degree n. The set of eigenvalues $\{\lambda_1, \dots, \lambda_n\}$ is known as the **spectrum** of A. The **spectral radius** of a matrix is defined by

$$\rho(A) = \max\{|\lambda_1|, \dots, |\lambda_n|\}.$$

As a final comment here, as A is a real matrix the polynomial $p_A(t)$ has real coefficients and as a consequence any non-real roots of $p_A(t)$ must occur in complex conjugate pairs.

1.4.1 Examples

1. Let
$$A = I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$
Then
$$p_A(t) = \det(I_2 - tI_2) = \begin{vmatrix} 1-t & 0 \\ 0 & 1-t \end{vmatrix} = (1-t)^2.$$
The eigenvalues are $\lambda_1 = \lambda_2 = 1$, i.e. we have a repeated eigenvalue. Every non-zero vector in $\mathbb{R}^2$ is an eigenvector.

2. Let
$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$
Then
$$p_A(t) = \begin{vmatrix} 2-t & 1 \\ 1 & 2-t \end{vmatrix} = (2-t)^2 - 1 = (1-t)(3-t).$$
The eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 3$. To get an eigenvector corresponding to $\lambda_1 = 1$ we need to obtain a non-zero solution to
$$(A - \lambda_1 I)\underline{x} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$
A solution is $\underline{x}^T = (1, -1)$.

   Similarly to get an eigenvector corresponding to $\lambda_2 = 3$ we need to obtain a non-zero solution to
$$(A - \lambda_2 I)\underline{x} = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$
A solution is $\underline{x}^T = (1, 1)$.

   Observe that the eigenvector corresponding λ_1 is orthogonal to the eigenvector corresponding to λ_2 which is a consequence of the matrix being real and symmetric. The spectral radius of this matrix is $\max\{1, 3\} = 3$.

3. Let
$$A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$
Then
$$p_A(t) = \begin{vmatrix} -t & -1 \\ 1 & -t \end{vmatrix} = t^2 + 1 = (i-t)(-i-t).$$

The eigenvalues are the complex conjugate pair $\lambda_1 = i$ and $\lambda_2 = -i$. The eigenvectors are also complex. The spectral radius of this matrix is 1. The matrix is a rotation matrix corresponding to anti-clockwise rotation by angle $\pi/2$ about the origin.

4. Let
$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}.$$
Then
$$p_A(t) = \begin{vmatrix} -t & 1 \\ 0 & -t \end{vmatrix} = t^2.$$
The eigenvalues are $\lambda_1 = \lambda_2 = 0$. To obtain an eigenvector we need to obtain a non-zero solution of
$$(A - 0I)\underline{x} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$
The only information that these equations give is that $x_2 = 0$ and thus the only direction which is an eigenvector is $\underline{x}^T = (1, 0)$. This is an example of a **deficient matrix** in that it has an eigenvalue of algebraic multiplicity equal to 2 (this is from considering the characteristic polynomial) but the dimension of the eigen-subspace corresponding to this eigenvalue is only 1. The dimension of the eigen-subspace corresponding to an eigenvalue is known as the geometric multiplicity.

   Note that in this example the spectral radius is 0 but the matrix is not the zero matrix although A^2 is the zero matrix. This example shows that for general matrices the spectral radius cannot be used to define a norm for square matrices with norms considered soon in this chapter.

Later in the module we will need eigenvalues and eigenvectors when we consider the solution of systems of differential equations of the form $\underline{u}' = A\underline{u}$ where $\underline{u} = \underline{u}(x) \in \mathbb{R}^n$ and A is a $n \times n$ constant matrix.

1.4.2 A summary of key points about eigenvalues and eigenvectors

Let $\underline{v}_i \neq 0$ denote an eigenvector associated with the eigenvalue λ_i, i.e.
$$A\underline{v}_i = \lambda_i \underline{v}_i, \quad i = 1, 2, \ldots, n.$$
The following are some important results about eigenvalues which are needed in this module.

1. An $n \times n$ matrix A is non-singular if and only if $\lambda_i \neq 0$ for $i = 1, 2, \ldots, n$.

2. If $\underline{v}_1, \ldots, \underline{v}_n$ are linearly independent then they give a basis for $\mathbb{R}^n$. In matrix form we have

$$\begin{aligned} A(\underline{v}_1, \ldots, \underline{v}_n) &= (A\underline{v}_1, \ldots, A\underline{v}_n) \\ &= (\lambda_1\underline{v}_1, \ldots, \lambda_n\underline{v}_n) \\ &= (\underline{v}_1, \ldots, \underline{v}_n)\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}. \end{aligned}$$

   If we let $V = (\underline{v}_1, \ldots, \underline{v}_n)$ be the matrix with the columns as the eigenvectors and we let $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ then in matrix form we have

$$AV = VD.$$

   As we are assuming that the columns of V are linearly independent it follows that V has an inverse and we can write

$$V^{-1}AV = D \quad \text{and} \quad A = VDV^{-1}.$$

   The first result is often referred to as diagonalising the matrix and the second version gives a representation of the matrix in terms of the eigenvalues and eigenvectors. Using this representation we can immediately get powers of A as

$$A^2 = (VDV^{-1})(VDV^{-1}) = VD^2V^{-1}$$

   and more generally

$$A^k = VD^kV^{-1}, \quad k = 1, 2, \ldots.$$

   When all the eigenvalues are non-zero we also have

$$A^{-1} = VD^{-1}V^{-1}.$$

3. If $\lambda_1, \ldots, \lambda_n$ are distinct then it can be shown that the eigenvectors $\underline{v}_1, \ldots, \underline{v}_n$ are linearly independent and the conditions in the previous item hold.

4. If A is a real symmetric matrix then we can diagonalise A using an orthogonal matrix, i.e. there exists an orthogonal matrix Q such that

$$Q^TAQ = D, \quad A = QDQ^T. \tag{1.4.1}$$

   This is a special case of the situation above with $V = Q$ as the inverse of an orthogonal matrix is its transpose, i.e. $Q^{-1} = Q^T$.

1.5 Vector and matrix norms

For a given object (vectors and matrices in our case) a norm is a way of giving a ‘size’ to the object. For example the modulus of a real or complex number defines a norm and the Euclidean length of a vector in three dimensional geometry defines a norm. With a way of giving a size to objects like vectors and matrices we can then consider when a vector $\underline{x}$ is close to another vector $\underline{y}$ by considering the size of $\underline{x} - \underline{y}$ and we can consider when a matrix A is close to another matrix B by considering the size of $A - B$. In the context of solving $A\underline{x} = \underline{b}$ situations like these arise when we have an approximate solution $\hat{\underline{x}}$ and we need to estimate or compute the size of the error $\underline{x} - \hat{\underline{x}}$ and the size of the residual term $\underline{b} - A\hat{\underline{x}}$.

In the n dimensional space $\mathbb{R}^n$ there are many different norms which can be defined and we will consider the most popular ones shortly. In the applications that are considered in this module it does not usually matter too much which norm is used as in a certain sense all norms in $\mathbb{R}^n$ are equivalent which roughly means that if $\|\underline{x}\|$ is very small or very big in one norm then it is also very small or very big respectively in another norm provided n is not too large.

1.5.1 Vector norms – the axioms

We begin abstractly by giving the **vector norm axioms**.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is a vector norm if the following conditions are satisfied:

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

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

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

If f is a norm then we use the notation

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

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

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

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

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

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

1.5.2 Commonly used vector norms

The most commonly used vector norms are the 2-norm, the $\infty-$norm and the $1-$ norm and for $\underline{x} \in \mathbb{R}^n$ these are defined as follows.

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

is the **2-norm** or **Euclidean norm**. (When we just refer to the length of a vector it is usually understood that this means the two norm.) In usage the next most popular norm is

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

which is the **∞-norm**. The **1-norm** is

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

There is a function called `norm()` in Matlab which computes these norms. For example, if you put

```
x=[1, -2, 2]
n2=norm(x)
n2b=norm(x, 2)
ninf=norm(x, inf)
n1=norm(x, 1)
```

Then the output is

```
x =
   1  -2   2
n2 =  3
n2b =  3
ninf =  2
n1 =  5
```

`norm(x)` and `norm(x, 2)` both give the 2-norm whilst in the other cases a second argument of `inf` or `1` has to be given.

As a final comment on the definition of the 2-norm, if we need to consider vectors with non-real entries then we need to change slightly what is given above to be such that

$$\|\underline{x}\|_2^2 = |x_1|^2 + \cdots + |x_n|^2 = \sum_{i=1}^n |x_i|^2 = \bar{\underline{x}}^T \underline{x}.$$

Here the notation $\bar{\underline{x}} = (\bar{x}_i)$, means that each entry is the complex conjugate of the corresponding entry of $\underline{x} = (x_i)$. In a later chapter there will be examples involving real matrices which have complex eigenvalues and eigenvectors but there will not be too many other examples in this module when we have vectors with non-real entries.

1.5.3 Matrix norms induced by vector norms

Let A denote a $n \times n$ matrix. Given any vector norm for vectors in $\mathbb{R}^n$ we can define the matrix norm of A induced by the vector norm as

$$\|A\| = \max\left\{\|A\underline{x}\| : \ \|\underline{x}\| = 1\right\}. \tag{1.5.1}$$

It is straightforward to show that a similar set of axioms to the vector case also hold here. More precisely we have the following.

(i) $\|A\| \ge 0$ for all $A \in \mathbb{R}^{n,n}$ with $\|A\| = 0$ if and only if $A = 0$. ($A = 0$ means A=zero matrix.) This is known as the non-negative condition.

(ii) $\|\alpha A\| = |\alpha|\,\|A\|$ for all scalars $\alpha \in \mathbb{R}$ and all $A \in \mathbb{R}^{n,n}$. This is known as the linearity condition.

(iii) $\|A + B\| \le \|A\| + \|B\|$ for all $A, B \in \mathbb{R}^{n,n}$. This is known as the triangle inequality.

We can add to these properties if we consider matrix-vector multiplication and matrix-matrix multiplication.

In the first case, an immediate consequence of the definition of the matrix norm is that if we consider a vector $\underline{x} \ne \underline{0}$ then

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

is a vector of norm 1 in the norm being used and

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

so that

$$\|A\underline{x}\| = \|\underline{x}\|\,\|A\underline{y}\| \le \|\underline{x}\|\,\|A\|.$$

Thus for all vectors $\underline{x} \in \mathbb{R}^n$ we have

$$\|A\underline{x}\| \le \|A\|\,\|\underline{x}\| \tag{1.5.2}$$

and for at least one direction there is a vector $\underline{x}$ with $\|A\underline{x}\| = \|A\|\,\|\underline{x}\|$.

In the case of matrix-matrix multiplication we have that if A and B are both $n \times n$ matrices then the products AB and BA are also $n \times n$ matrices. If $\underline{x}$ is such that $\|\underline{x}\| = 1$ then $AB\underline{x} = A(B\underline{x})$ and thus

$$\begin{aligned}\|AB\underline{x}\| &\le \|A\|\,\|B\underline{x}\|, \qquad \text{using (1.5.2) in the case of the matrix } A,\\ &\le \|A\|\,\|B\|\,\|\underline{x}\|, \qquad \text{using (1.5.2) in the case of the matrix } B,\\ &= \|A\|\,\|B\|, \qquad \text{as the vector } \underline{x} \text{ has norm 1.}\end{aligned}$$

Thus

$$\|AB\| = \max\{\|AB\underline{x}\| :\ \|\underline{x}\| = 1\} \le \|A\|\,\|B\|.$$

Similarly

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

These indicate that we can bound the norm of the product of two matrices in terms of the norms of the matrices involved and in particular note that it follows that

$$\|A^k\| \le \|A\|^k, \quad k = 1, 2, \ldots.$$

1.5.4 The matrix norms $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$

Each of the common vector norms generates an induced matrix norm and it turns out that it is not too difficult to get explicit expressions for these although we omit the details of the derivations in these notes although some these are considered in the exercise sheet questions. We use the notation $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$ for the matrix norms induced by the vector 1-norm, 2-norm and ∞-norm respectively. The results are as follows.

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

where, as on page 1-5,

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

These results show that it is straightforward to compute $\|A\|_\infty$ and $\|A\|_1$ but that the matrix 2-norm involves computing the dominant eigenvalue of $A^T A$, i.e. the largest in magnitude eigenvalue of $A^T A$, which can usually only be done numerically and it involves much more work than the other two cases.

As an example, consider the following 3×3 matrix which is on the wiki-page about matrix norms.

$$A = \begin{pmatrix} 3 & 5 & 7 \\ 2 & 6 & 4 \\ 0 & 2 & 8 \end{pmatrix}.$$

The row sums of the absolute values are 15, 12 and 10 and thus $\|A\|_\infty = 15$. The column sums of the absolute values are 5, 13 and 19 and thus $\|A\|_1 = 19$. We can use Matlab to work out the 2-norm and this can be done by the following commands

```
A=[3, 5, 7;  2, 6, 4;  0, 2, 8]
ninf=norm(A, inf)
n1=norm(A, 1)
n2=norm(A)
```

which generate the following output

```
A =
   3   5   7
   2   6   4
   0   2   8
ninf =  15
n1 =  19
n2 =  13.686
```

1.6 The spectral radius and matrix norms

There are some connections between the spectral radius $\rho(A)$ of a matrix A and any of the induced matrix norms $\|A\|$.

Firstly, if $\underline{v} \neq \underline{0}$ is an eigenvector of A with eigenvalue λ, i.e. $A\underline{v} = \lambda\underline{v}$ then any scaling of $\underline{v}$ is also an eigenvector and thus there is a eigenvector corresponding to λ such that $\|\underline{v}\| = 1$. In this case

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

This shows that an induced matrix norm is a bound on the magnitude of all the eigenvalues are thus in particular

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

As we can easily compute $\|A\|_1$ and $\|A\|_\infty$ we get that

$$\rho(A) \leq \min\left\{\|A\|_1,\ \|A\|_\infty\right\}.$$

Secondly, if the matrix is real and symmetric, i.e. $A \in \mathbb{R}^{n\times n}$ and $A^T = A$, then $\|A\|_1 = \|A\|_\infty$ and

$$\|A\|_2^2 = \rho(A^TA) = \rho(A^2).$$

When we have a real symmetric matrix all the eigenvalues are real and for any matrix

$$A\underline{v} = \lambda\underline{v} \quad \text{implies that} \quad A^2\underline{v} = A(\lambda\underline{v}) = \lambda^2\underline{v}.$$

The eigenvalues of A^2 are the square of the eigenvalues of A and thus in the real symmetric case

$$\|A\|_2 = \rho(A) = \max\left\{|\lambda_i| :\ \lambda_i \text{ is an eigenvalue of } A\right\}.$$

As we still have bounds on $\rho(A)$, by using $\|A\|_1 = \|A\|_\infty$ it follows that in the real symmetric case

$$\|A\|_2 = \rho(A) \leq \|A\|_1 = \|A\|_\infty.$$

1.7 The matrix condition number

When A is a $n \times n$ matrix which is non-singular it has an inverse A^{-1} and the matrix condition number is defined as

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

When A is singular we say that $\kappa(A) = \infty$. From the result that $\|AB\| \leq \|A\|\,\|B\|$, that $I = A^{-1}A$ and that $\|I\| = 1$ it follows that for all square matrices

$$1 \leq \kappa(A) \leq \infty.$$

This quantity can be expressed in terms of eigenvalues when we have a real symmetric matrix as when all the eigenvalues of A are non-zero the relation

$$A\underline{v} = \lambda\underline{v} \quad \text{rearranges to} \quad A^{-1}\underline{v} = \left(\frac{1}{\lambda}\right)\underline{v}.$$

Thus if the eigenvalues of a non-singular matrix A are $\lambda_1, \ldots, \lambda_n$ then the eigenvalues of A^{-1} are $\lambda_1^{-1}, \ldots, \lambda_n^{-1}$. If for convenience we suppose that the labelling is such that

$$0 < |\lambda_n| \le \cdots \le |\lambda_1|$$

then

$$0 < \frac{1}{|\lambda_1|} \le \cdots \le \frac{1}{|\lambda_n|}$$

and with the 2-norm and a symmetric matrix

$$\|A\|_2 = |\lambda_1|, \quad \|A^{-1}\|_2 = \frac{1}{|\lambda_n|}$$

and the condition number using the 2-norm is

$$\kappa(A) = \frac{|\lambda_1|}{|\lambda_n|}.$$

The significance of the condition number to this module is that a large condition number indicates that the matrix is close to a singular matrix and we cannot expect to accurately solve a linear system $A\underline{x} = \underline{b}$ by any means. With some further analysis it can be shown that $1/\kappa(A)$ gives the relative distance of the matrix A to a singular matrix, i.e. for all singular matrices C of the same shape as A we have

$$\frac{\|A - C\|}{\|A\|} \ge \frac{1}{\kappa(A)}.$$

and there exists a singular matrix B such that

$$\frac{\|A - B\|}{\|A\|} = \frac{1}{\kappa(A)}.$$

In the case of the 2-norm the nearest singular matrix B to the non-singular matrix A can be quite easily described by using what is known as the singular valued decomposition of A which is a topic beyond this module. In the particular case that A is real and symmetric the situation is slightly easier by noting that we have the representation of A in terms of eigenvalues and eigenvectors given in (1.4.1)

$$\begin{aligned} A &= QDQ^T \\ &= \lambda_1 \underline{v}_1 \, \underline{v}_1^T + \cdots + \lambda_{n-1} \underline{v}_{n-1} \, \underline{v}_{n-1}^T + \lambda_n \underline{v}_n \, \underline{v}_n^T. \end{aligned}$$

If we replace the smallest eigenvalue in magnitude (which is λ_n) by 0 then we get a singular matrix. If we denote this singular matrix by B then it has the representation

$$B = \lambda_1 \underline{v}_1 \, \underline{v}_1^T + \cdots + \lambda_{n-1} \underline{v}_{n-1} \, \underline{v}_{n-1}^T$$

and it can be shown that this is the nearest singular matrix to the matrix A with

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

Chapter 2

Direct methods for solving $A\underline{x} = \underline{b}$

2.1 Introduction

In this chapter we consider **Gauss elimination** type methods for a solving a system of linear equations

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

where A is an $n \times n$ matrix and $\underline{b}$ is a $n \times 1$ column vector. Methods of this type are used in Matlab when you give the command

```
x=A\b;
```

when `A` and `b` have the appropriate shape in the Matlab workspace. In fact when you put `A\b` in Matlab the software attempts to classify the type of matrix involved and then selects the "best" technique to use to reliably solve the problem in an efficient way. Different methods are used if for example the matrix is symmetric or if it is tri-diagonal as compared to what is done if A is a full matrix with no special properties. (The symmetric case also needs a property known as positive definite.) In this chapter we just consider this general case and consider in general the Gauss elimination procedure involving systematically reducing a general system to an equivalent system involving a triangular matrix. A key result of the chapter is in showing that the Gauss elimination process is equivalent to factorizing the matrix A in terms of triangular matrices, i.e. in its basic form (when it works) we show that we get the factorization

$$A = LU,$$

where L is a **unit lower triangular matrix** and U is an **upper triangular matrix**. In the case that $n = 4$ this means that we have a factorization of the form

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}.$$

An upper triangular matrix is a matrix for which all the entries below the diagonal are 0, a lower triangular matrix is a matrix for which all the entries above the diagonal are 0 and a unit lower triangular matrix is a lower triangular matrix with the diagonal entries all being equal to 1. For Gauss elimination to work in its basic form needs A to have special properties. For a more stable way of factorizing a matrix packages instead create a factorization of the form

$$PA = LU,$$

where P is permutation matrix. In this form the matrix PA is obtained from A by rearranging the rows. In Matlab you can get the matrices generated by using the command

```
[L, U, P]=lu(A);
```

Gauss elimination and implementations involving LU factorization are examples of **direct methods** for solving $A\underline{x} = \underline{b}$ where the term direct method means that the exact solution is obtained in a finite number of steps provided exact arithmetic is used. However it should be realised that to do computations quickly on a computer floating point arithmetic is used and this has rounding error with calculation typically done with close to 16 decimal digit accuracy. Thus in practice we only get an approximate solution although hopefully the method can generate an approximate solution which is close to machine accuracy.

2.2 Forward and backward substitution algorithms

We start by considering systems with triangular matrices and we start with two examples which can be done by hand calculations.

Consider the upper triangular system

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

The last equation only involves x_4 and hence we get this immediately. Then using x_4 we can use the previous equation to get x_3 and continue in a similar way to get x_2 and then x_1. For the details

$$\begin{aligned} 4x_4 &= 4, \quad \text{giving } x_4 = 1, \\ 3x_3 &= -2 - x_4, \quad \text{giving } x_3 = -1, \\ 2x_2 &= 3 - x_3, \quad \text{giving } x_2 = 2, \\ x_1 &= -x_3, \quad \text{giving } x_1 = -2. \end{aligned}$$

Next consider the lower triangular system

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 2 \\ 0 \end{pmatrix}.$$

The first equation only involves x_1 and hence we get this immediately. Then using x_1 we can use the second equation to get x_2 and continue in a similar way to get x_3 and then x_4. For the details

$$\begin{aligned}
x_1 &= 2,\\
x_2 &= 3 - x_1, \quad \text{giving } x_2 = 1,\\
x_3 &= 2 - x_1 - x_2, \quad \text{giving } x_3 = -1,\\
x_4 &= -x_1 - x_2 - x_3, \quad \text{giving } x_4 = -2.
\end{aligned}$$

The procedure used in the case of an upper triangular matrix is known as **backward substitution** and the procedure used in the case of the lower triangular system is known as **forward substitution**. We now consider how to do this generally for $n \times n$ triangular matrices. For a general upper triangular system $U\underline{x} = \underline{b}$, i.e.

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

backward substitution can be described as follows.

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

For a general lower triangular system $L\underline{x} = \underline{b}$, i.e.

$$\begin{array}{ccccccccc}
l_{11}x_1 & & & & & & & = & b_1\\
l_{21}x_1 & + & l_{22}x_2 & & & & & = & b_2\\
\vdots & & \vdots & & & & & & \\
l_{n1}x_1 & + & \cdots & + & l_{n,n-1}x_{n-1} & + & l_{n,n}x_n & = & b_n
\end{array}$$

forward substitution can be described as follows.

$$\begin{aligned}
x_1 &= b_1/l_{11}\\
x_i &= \left(b_i - \sum_{k=1}^{i-1} l_{ik}x_k\right)/l_{ii}, \quad i = 2, \ldots, n\ .
\end{aligned}$$

In both cases the procedure requires that the diagonal entries are non-zero as we divide by these entries.

In algorithms implemented on a computer using floating point arithmetic an operation involving 1 multiplication and 1 addition/subtraction is sometimes referred to as a **flop** although it seems now more common to refer to this as 2 flop operations as 2 operations are involved and this is what will be done here. The computational cost of an algorithm is often expressed in terms of how many such operations are needed to solve a problem. The speed of a computer is also often expressed in terms of how many of such operations can be done in 1 second (the terms **flops** is used in this case). In the case of the number of operations involved in the backward or forward substitution this can be determined without too much effort and in the case of forward substitution the computation of

$$b_i - \sum_{k=1}^{i-1} l_{ik} x_k$$

involves $2(i-1)$ flop operations and this must be done for $i = 2, \ldots, n$. In total this gives

$$2(1 + 2 + 3 + \cdots + (n-1)) = n(n-1).$$

When n is large there is a much smaller number of divisions and we say that in total there is an the order of n^2 operations which is often written as $\mathcal{O}(n^2)$. The operation count with backward substitution is the same. Thus in each case the triangular matrices involve $\mathcal{O}(n^2/2)$ entries and the number of operations needed to solve the systems is $\mathcal{O}(n^2)$.

2.3 Solving a system $LU\underline{x} = \underline{b}$

Before we consider how to obtain a factorization $A = LU$ we consider briefly how this helps us to solve a system of equations

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

i.e. we start by assuming that we have a lower triangular matrices L and an upper triangular U such that $A = LU$. All we do is to let $\underline{y} = U\underline{x}$ and first solve a system $L\underline{y} = \underline{b}$ and once we have $\underline{y}$ we solve a second system $U\underline{x} = \underline{y}$ to get $\underline{x}$. To express this as an algorithm we have the following.

Solve $L\underline{y} = \underline{b}$ by forward substitution.

Solve $U\underline{x} = \underline{y}$ by backward substitution.

The number of operations involved is $\mathcal{O}(2n^2)$. This is much less operations than the number of operations needed to get the triangular matrices L and U which we show needs $\mathcal{O}(2n^3/3)$ operations as we describe in the next section.

2.4 Reduction to triangular form by Gauss elimination

In Gauss elimination you eliminate unknowns by subtracting appropriate multiples of one equation from the other equations and in terms of the coefficients this involves subtracting multiples of one row from other rows. In the case that $n = 4$ the sequence of matrices generated by basic Gauss elimination has the following form.

$$\begin{pmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathrm{x} & \mathrm{x} \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & 0 & \mathrm{x} & \mathrm{x} \\ 0 & 0 & 0 & \mathrm{x} \end{pmatrix}.$$

In each case x just indicates an entry which is likely to be non-zero and in the above we are just considering the changes in the matrix without considering the changes that are also done to the right hand side vector. The first matrix refers to the starting matrix A and the final matrix is an upper triangular matrix which we refer to as U. There are 3 steps here with each step associated with creating zero entries below the diagonal in a given column. In the case of a general $n \times n$ matrix there are $n-1$ such steps.

There is a choice at each stage as to which row you use to subtract multiples of from the other rows and there is even the possibility of re-ordering the unknowns which involves re-ordering the columns. In the basic version of Gauss elimination none of the above is done and we always subtract multiples of the current top row from the rows below. Thus in the basic form you are not also making decisions about which row to use to make the hand computations as easy as possible and you are not changing one row by multiplying it by a scalar as you may have done when dealing with small matrices in level one. Basic Gauss elimination is hence one specific order of the operations that you were taught at level one in the context of solving a linear system and we consider next how to describe this in a matrix way and we show that it is equivalent to a factorization.

To describe the operations in a matrix way let $A^{(0)} = A$ and let $A^{(k)}$, $k = 1, \ldots, n-1$ denote the intermediate matrices with $U = A^{(n-1)}$ denoting the final upper triangular matrix assuming the procedure runs to completion. In the case $n = 4$ the matrices are

$$A^{(0)} = A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}, \quad A^{(1)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & a_{34}^{(1)} \\ 0 & a_{42}^{(1)} & a_{43}^{(1)} & a_{44}^{(1)} \end{pmatrix},$$

$$A^{(2)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & 0 & a_{43}^{(2)} & a_{44}^{(2)} \end{pmatrix}, \quad U = A^{(3)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & a_{24}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & a_{34}^{(2)} \\ 0 & 0 & 0 & a_{44}^{(3)} \end{pmatrix}.$$

As a consequence of the order in which the reduction is done the first row never changes, the second row never changes once $A^{(1)}$ has been computed, the third row never changes once $A^{(2)}$ has been computed and the final entry to be computed is the $4, 4$ entry of U. We show next how to describe each reduction step as a matrix multiplication with

$$A^{(1)} = M_1 A^{(0)}, \quad A^{(2)} = M_2 A^{(1)}, \quad U = A^{(3)} = M_3 A^{(2)}$$

so that

$$U = M_3 M_2 M_1 A$$

and

$$A = (M_3 M_2 M_1)^{-1} U = (M_1^{-1} M_2^{-1} M_3^{-1}) U = LU$$

where

$$L = M_1^{-1} M_2^{-1} M_3^{-1}.$$

The matrices M_k are known as Gauss transformation matrices, which are described in a moment, and we will show that the matrix L is a unit lower triangular matrix.

Firstly, to obtain $A^{(1)}$ we subtract multiples of the first row from rows 2, 3 and 4. Now the first row at this stage is

$$\left(a_{11}^{(0)}, \quad a_{12}^{(0)}, \quad a_{13}^{(0)}, \quad a_{14}^{(0)} \right)$$

and the ith row, also at this stage, is

$$\begin{pmatrix} a_{i1}^{(0)}, & a_{i2}^{(0)}, & a_{i3}^{(0)}, & a_{i4}^{(0)} \end{pmatrix}$$

and thus to eliminate x_1 the multiples are

$$m_{i1} = \frac{a_{i1}^{(0)}}{a_{11}^{(0)}}, \quad i = 2, 3, 4.$$

In terms of rows the ith row of the next matrix $A^{(1)}$ is

$$\begin{pmatrix} a_{i1}^{(0)}, & a_{i2}^{(0)}, & a_{i3}^{(0)}, & a_{i4}^{(0)} \end{pmatrix} - m_{i1} \begin{pmatrix} a_{11}^{(0)}, & a_{12}^{(0)}, & a_{13}^{(0)}, & a_{14}^{(0)} \end{pmatrix}, \quad i = 2, 3, 4.$$

To describe this in a matrix way observe that

$$\begin{aligned} \underline{e}_1^T A^{(0)} &= \begin{pmatrix} 1, & 0, & 0, & 0 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \\ &= \begin{pmatrix} a_{11}, & a_{12}, & a_{13}, & a_{14} \end{pmatrix} \\ &= \text{1st row of } A^{(0)}, \end{aligned}$$

and

$$\begin{pmatrix} 0 \\ m_{21} \\ m_{31} \\ m_{41} \end{pmatrix} \underline{e}_1^T A^{(0)}$$

is the 4×4 matrix to subtract from $A^{(0)}$ to get $A^{(1)}$. Thus if we let

$$\underline{m}_1 = \begin{pmatrix} 0 \\ m_{21} \\ m_{31} \\ m_{41} \end{pmatrix}$$

then

$$A^{(1)} = A^{(0)} - \underline{m}_1 \underline{e}_1^T A^{(0)} = (I - \underline{m}_1 \underline{e}_1^T) A^{(0)}$$

and the Gauss transformation matrix at the first step is

$$M_1 = I - \underline{m}_1 \underline{e}_1^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -m_{21} & 1 & 0 & 0 \\ -m_{31} & 0 & 1 & 0 \\ -m_{41} & 0 & 0 & 1 \end{pmatrix}.$$

To get $A^{(2)}$ from $A^{(1)}$ we subtract multiples of the 2nd row from the rows below and the vector of the multipliers is

$$\underline{m}_2 = \begin{pmatrix} 0 \\ 0 \\ m_{32} \\ m_{42} \end{pmatrix}, \quad \text{where } m_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}, \quad i = 3, 4.$$

The Gauss transformation matrix is

$$M_2 = I - \underline{m}_2\underline{e}_2^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -m_{32} & 1 & 0 \\ 0 & -m_{42} & 0 & 1 \end{pmatrix}.$$

Similarly to get $U = A^{(3)}$ from $A^{(2)}$ we have

$$\underline{m}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ m_{43} \end{pmatrix}, \quad \text{where } m_{43} = \frac{a_{43}^{(2)}}{a_{33}^{(2)}},$$

and the Gauss transformation matrix is

$$M_3 = I - \underline{m}_3\underline{e}_3^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -m_{43} & 1 \end{pmatrix}.$$

The next task is describe the inverse matrices M_1^{-1}, M_2^{-1} and M_3^{-1} and the product of these which is $L = M_1^{-1}M_2^{-1}M_3^{-1}$. This is very straightforward as

$$M_k^{-1} = I + \underline{m}_k\underline{e}_k^T, \quad k = 1, 2, 3.$$

This follows by noting that at the kth stage the kth row does not change and thus we can reverse the reduction process by adding the same multiples of the kth row to the other rows. Alternatively we can verify the expression for the inverse by just considering the product

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

where the last equality is because

$$\underline{e}_k^T\underline{m}_k = k\text{th entry of } \underline{m}_k = 0$$

as $\underline{m}_k$ has entries of 0 in positions $1, \ldots, k$. For the product of these matrices first note that

$$\begin{aligned} M_1^{-1}M_2^{-1} &= (I + \underline{m}_1\underline{e}_1^T)(I + \underline{m}_2\underline{e}_2^T) \\ &= I + \underline{m}_1\underline{e}_1^T + \underline{m}_2\underline{e}_2^T + \underline{m}_1(\underline{e}_1^T\underline{m}_2)\underline{e}_2^T \\ &= I + \underline{m}_1\underline{e}_1^T + \underline{m}_2\underline{e}_2^T \end{aligned}$$

where the last equality is because

$$\underline{e}_1^T\underline{m}_2 = 1\text{st entry of } \underline{m}_2 = 0.$$

Similar reasoning gives

$$\begin{aligned} L = M_1^{-1}M_2^{-1}M_3^{-1} &= (I + \underline{m}_1\underline{e}_1^T + \underline{m}_2\underline{e}_2^T)(I + \underline{m}_3\underline{e}_3^T) \\ &= I + \underline{m}_1\underline{e}_1^T + \underline{m}_2\underline{e}_2^T + \underline{m}_3\underline{e}_3^T \end{aligned}$$

where the last equality is because

$$\underline{e}_1^T \underline{m}_3 = \underline{e}_2^T \underline{m}_3 = 0.$$

Thus in full

$$L = I + \underline{m}_1 \underline{e}_1^T + \underline{m}_2 \underline{e}_2^T + \underline{m}_3 \underline{e}_3^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ m_{21} & 1 & 0 & 0 \\ m_{31} & m_{32} & 1 & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{pmatrix}$$

which is a unit lower triangular matrix and the entries below the diagonal are exactly the multipliers used in the basic Gauss elimination process.

The general $n \times n$ case

The above description easily generalises to the case of a general $n \times n$ matrix which requires elimination below the diagonal in columns $1, 2, \ldots, n-1$ and this involves matrices $A^{(0)} = A$, $A^{(1)}, \ldots, A^{(n-1)} = U$. In the general case we have for $k = 1, \ldots, n-1$,

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

In the above the multipliers are

$$m_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k+1, \ldots, n, \tag{2.4.1}$$

the inverse matrices are

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

and for the product we have

$$M_1^{-1} \cdots M_r^{-1} = I + \underline{m}_1 \underline{e}_1^T + \cdots + \underline{m}_r \underline{e}_r^T, \quad r = 2, \ldots, n-1.$$

This can be proved by induction and there is an exercise question about this. For the algorithm to run to completion we need the matrix to be such that we never divide by 0, i.e. we need that $a_{kk}^{(k-1)} \neq 0$ for $k = 1, \ldots, n-1$ and also for U to be invertible we need that $a_{nn}^{(n-1)} \neq 0$. The terms $u_{kk} = a_{kk}^{(k-1)}$ are the **pivot** elements and they are the diagonal entries of U (assuming that the algorithm runs to completion to create U).

To count the number of operations involved note that to get $A^{(1)}$ from A involves computing $(n-1)^2$ entries corresponding to positions $2 \leq i, j \leq n$ and there are 2 floating point operation associated with each entry. Similarly to get $A^{(2)}$ from $A^{(1)}$ requires changing $(n-2)^2$ entries with 2 operations in each case and continuing in this way the total reduction process requires the following number of operations

$$\begin{aligned} &2((n-1)^2 + (n-2)^2 + \cdots + 2^2 + 1) \\ &= 2\frac{(n-1)n(2n-1)}{6} \sim \frac{2n^3}{3} \quad \text{(for large } n\text{)}. \end{aligned}$$

It is not necessary to be too precise here other than to note that we have the n^3 term and we can get the constant $2/3$ by considering the integral $2\int_0^n x^2\,\mathrm{d}x$ as the area under the curve $y = 2x^2$ is similar in some sense to the sum. Hence as we double the size of a matrix we get 4 times as many entries and we require about 8 times as much work to solve a system $A\underline{x} = \underline{b}$ when A is a full matrix with no special properties. This operation count should be compared with the forward and backward substitution algorithms which just has a term of involving n^2. The point here to note is that for large matrices the reduction to triangular form is where most of the computation is done.

Example

Let

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

The Gauss elimination procedure to construct the LU factorization involves the following steps.

Elimination in column 1:

$$\underline{m}_1 = \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix}, \quad A^{(1)} = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & 5 \end{pmatrix}.$$

Elimination in column 2:

$$\underline{m}_2 = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}, \quad U = A^{(2)} = \begin{pmatrix} 2 & 3 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 4 \end{pmatrix}.$$

The unit lower triangular matrix is

$$L = I + \underline{m}_1\underline{e}_1^T + \underline{m}_2\underline{e}_2^T = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}.$$

Thus we have shown that

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

To check the calculations using Matlab all that is needed is to put the following.

```
A=[2, 3, 1; -2, -2, -2; -2, -4, 4]
[L U]=lu(A)
```

The output generated is

```
A =
    2   3   1
   -2  -2  -2
   -2  -4   4
L =
    1   0   0
   -1   1   0
   -1  -1   1
U =
    2   3   1
    0   1  -1
    0   0   4
```

It is worth noting here that the Matlab command `lu` will not necessarily give the LU factorization when used in this way as it always uses pivoting which is described shortly. In this particular example the pivoting decisions was that nothing needed to be re-arranged.

The factorization of sub-matrices

For a $n \times n$ matrix $A = (a_{ij})$ the $k \times k$ principal sub-matrix is

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

In the previous example we can immediately note that for the sub-matrices of size $k = 1$ and $k = 2$ we have

$$2 = (1)(2)$$

and

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

This shows that in this example the factorization of the full matrix also gives the factorization of all the principal sub-matrices. To prove that this is true in general we just need to consider what matrix multiplication means when we have $A = LU$ with $A = (a_{ij})$, $L = (l_{ij})$ and $U = (u_{ij})$.

$$\begin{aligned} a_{ij} = (LU)_{ij} &= (i\text{th row of } L)(j\text{th column of } U) \\ &= \sum_{r=1}^{n} l_{ir} u_{rj} \\ &= \sum_{r=1}^{\min\{i,j\}} l_{ir} u_{rj} \end{aligned}$$

where the last equality is because L and U are triangular and thus $l_{ir} = 0$ for $r > i$ and $u_{rj} = 0$ for $r > j$. If we restrict the indices to $1 \le i, j \le k$ then these entries of A are entirely determined by the entries of L and U in this range, i.e.

$$\begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \cdots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix} = \begin{pmatrix} l_{11} & & \\ \vdots & \ddots & \\ l_{k1} & \cdots & l_{kk} \end{pmatrix} \begin{pmatrix} u_{11} & \cdots & u_{1k} \\ & \ddots & \vdots \\ & & u_{kk} \end{pmatrix}, \quad k = 1, 2, \dots, n.$$

Performing the calculations in a different order

In computer packages the order in which the entries are determined when computing a LU factorization is generally different to that which is done in the basic Gauss elimination algorithm. To illustrate that a different order can be used consider again the previous example.

$$A = \begin{pmatrix} 2 & 3 & 1 \\ -2 & -2 & -2 \\ -2 & -4 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}.$$

Motivated by the observation that the factorization of A also means the factorization of all principal sub-matrices we can calculate the 1st column of U, then the 2nd row of L, then the 2nd column of U, then the 3rd row of L and finally the 3rd column of U. Thus

$$\begin{aligned} a_{11} = 2 &= u_{11}, \\ a_{21} = -2 &= l_{21}u_{11}, \quad \text{giving } l_{21} = -1, \\ a_{12} = 3 &= u_{12}, \\ a_{22} = -2 &= l_{21}u_{12} + u_{22} = -3 + u_{22}, \quad \text{giving } u_{22} = 1. \end{aligned}$$

At this stage we hence have

$$\begin{pmatrix} 2 & 3 & 1 \\ -2 & -2 & -2 \\ -2 & -4 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}.$$

Continuing to get the 3rd row of L and the 3rd column of U.

$$\begin{aligned} a_{31} = -2 &= l_{21}u_{11}, \quad \text{giving } l_{31} = -1, \\ a_{32} = -4 &= l_{31}u_{12} + l_{32}u_{22} = -3 + l_{32}, \quad \text{giving } l_{32} = -1, \\ a_{13} = 1 &= u_{13}, \\ a_{23} = -2 &= l_{21}u_{13} + u_{23} = -1 + u_{23}, \quad \text{giving } u_{23} = -1, \\ a_{33} = 4 &= l_{31}u_{13} + l_{32}u_{23} + u_{33} = -1 + 1 + u_{33}, \quad \text{giving } u_{33} = 4. \end{aligned}$$

This procedure of starting with $u_{11} = a_{11}$ and then successively computing a row of L followed by a column of U can be generalised to the case of an $n \times n$ matrix A and it can be implemented more efficiently on a computer than is the case when the Gauss elimination order of operations is used. There is no saving when small problems are done by hand calculations other than we more quickly determine that a principal sub-matrix is singular as this is the case if we obtain a diagonal entry of U which is equal to 0. More precisely, if the diagonal entries $u_{11}, \dots, u_{k-1,k-1}$ are non-zero but $u_{kk} = 0$ then the $k \times k$ principal sub-matrix has zero determinant and is not invertible and we cannot continue.

2.5 Partial pivoting and the factorization $PA = LU$

Basic Gauss elimination for a non-singular matrix A only works if all the pivot elements are non-zero and this is the case if and only if all the principal sub-matrices of A are invertible. Situations do arise when the matrix A has such properties but otherwise we need to consider re-arranging the rows at each stage. We consider some examples to illustrate this.

The basic Gauss elimination algorithm fails for the 2×2 system

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

although we can spot the solution $x_2 = 1$ and $x_1 = 2$. To modify the basic Gauss elimination method we need to swap the equations to give

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

which is already in upper triangular form.

The basic Gauss elimination algorithm works in theory if we change slightly the previous example and have

$$\begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

With hand computations

$$\underline{m}_1 = \begin{pmatrix} 0 \\ 10^{20} \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{pmatrix}.$$

However with floating point arithmetic U is rounded to

$$\begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix}$$

on a computer. The problem with doing this calculation on a computer is that although A is not nearly singular, and the inverse is

$$A^{-1} = \frac{1}{10^{-20} - 1} \begin{pmatrix} 1 & -1 \\ -1 & 10^{-20} \end{pmatrix} \approx \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix},$$

both L and U are very close to being singular and combining this with rounding errors that occur leads to the linear system not being solved accurately on a computer with floating point arithmetic. In this particular case solving $L\underline{y} = \underline{b}$ gives

$$y_1 = 1, \quad y_2 = 3 - 10^{20} \quad \text{which rounds to } -10^{20}.$$

Then solving with the matrix U after rounding, i.e. solving $U\underline{x} = \underline{y}$ gives

$$x_2 = 1, \quad \text{and } x_1 = 10^{20}(1 - 1) = 0$$

whereas the exact solution is very close to $(2, 1)^T$. It is the rounding error which lead to the $(1-1) = 0$ part in the computation of x_1 and the inaccurate result for this component.

If we had done the calculation exactly then at this stage we would have had a very small number here and the computation of the large number times the small number would have given the correct value for x_1.

To get an accurate answer in this example we swap the rows to give

$$\begin{pmatrix} 1 & 1 \\ 10^{-20} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 10^{-20} & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1-10^{-20} \end{pmatrix}$$

and everything works well.

Now consider the 3×3 matrix

$$A = \begin{pmatrix} 4 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Basic Gauss elimination starts and we get

$$\underline{m}_1 = \begin{pmatrix} 0 \\ 1/2 \\ 1/4 \end{pmatrix}, \quad A^{(1)} = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 0 & 1/2 \\ 0 & 1/2 & 3/4 \end{pmatrix}.$$

As the entry in the $2,2$ position is 0 the basic procedure cannot continue but if we swap the 2nd and 3rd rows to give

$$\underline{\tilde{m}}_1 = \begin{pmatrix} 0 \\ 1/4 \\ 1/2 \end{pmatrix}, \quad \tilde{A}^{(1)} = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 1/2 & 3/4 \\ 0 & 0 & 1/2 \end{pmatrix}$$

then we are already in upper triangular form. For the factorization we have

$$PA = \begin{pmatrix} 4 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1/4 & 1 & 0 \\ 1/2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & 2 & 1 \\ 0 & 1/2 & 3/4 \\ 0 & 0 & 1/2 \end{pmatrix}.$$

The permutation matrix P is obtained by swapping rows 2 and 3 of the identity matrix, i.e.

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

To check if Matlab does the same re-arrangements and gives the same output you just need to put the following.

```
A=[4, 2, 1; 2, 1, 1; 1, 1, 1]
[L, U, P]=lu(A)
```

This generates the following output.

```
A =
    4    2    1
    2    1    1
    1    1    1
```

```
L =
   1.00000   0.00000   0.00000
   0.25000   1.00000   0.00000
   0.50000   0.00000   1.00000
U =
   4.00000   2.00000   1.00000
   0.00000   0.50000   0.75000
   0.00000   0.00000   0.50000
P =
   1   0   0
   0   0   1
   0   1   0
```

In the examples given we have either had to swap rows to avoid dividing by 0 or it was desirable to swap to avoid very large entries in L and U. The strategy in partial pivoting is to swap so that the largest entry is put in the pivot position after considering all the candidates for the pivot entries in the column being reduced. In the general $n \times n$ case this means that when we are at the stage of creating $A^{(k)}$ from $A^{(k-1)}$ instead of just computing the multipliers

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

(as first given in (2.4.1) on page 2-8) the strategy involves first determining $r \ge k$ such that

$$|a_{rk}^{(k-1)}| = \max\left\{|a_{kk}^{(k-1)}|, \dots, |a_{nk}^{(k-1)}|\right\}$$

and then swapping row r and row k of the relevant matrices if $r > k$. After the swapping this guarantees that all the multipliers (which are entries of L) have magnitude of less than or equal to 1. With slight variations this is what is done in practice and it works well in practice in almost all cases although it is not guaranteed to work in every possible case for matrices A which are well conditioned. The exercise sheets contain an example illustrating when this pivoting strategy fails (at least when n is not small) but failure in practice is actually rare, i.e. if matrices are generated randomly then the probability of failure is extremely low. The analysis of this is beyond the scope of this module.

2.6 Remarks about the inverse matrix A^{-1}

In this chapter Gauss elimination and the related LU factorization have been considered as a way of solving a general linear system $A\underline{x} = \underline{b}$. In level 1 you would have also been taught methods for finding the inverse matrix A^{-1} and then you would possibly have used it to construct the solution to the linear system by doing the multiplication

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

In level 1 this is likely to have only been considered when the number of unknowns $n = 2$ or $n = 3$. This is not the most efficient approach for these small problems but not too much work is required overall. For larger problems that you do on a computer there is about 3 times as much work involved in solving a linear system by first computing the inverse matrix as compared to using Gauss elimination. Thus when we write $\underline{x} = A^{-1}\underline{b}$ it should be considered as a way of describing the solution and not as a way of computing the solution. There are not too many situations when you need to compute A^{-1} but in cases that you do the methods to do this can make use of the methods described so far in this chapter as follows.

Before an algorithm is given we note that if $\underline{e}_j$ denotes the usual base vector (i.e. the jth column of I) then

$$\underline{x}_j = A^{-1}\underline{e}_j = j\text{th column of } A^{-1}$$

and thus the jth column of the inverse matrix satisfies the linear system

$$A\underline{x}_j = \underline{e}_j \quad \text{or equivalently } PA\underline{x}_j = P\underline{e}_j$$

for any permutation matrix P. Hence if we successively choose $\underline{b}$ to be $\underline{e}_1, \dots, \underline{e}_n$ and solve $A\underline{x} = \underline{b}$ then we get A^{-1} column-by-column. To do this efficiently we first factorize A and then repeatedly use the forward and backward substitution algorithms. An algorithm is hence as follows.

Step 1: Construct the factorization $PA = LU$.

Step 2: For $j = 1, 2, \dots, n$ solve

$$LU\underline{x}_j = P\underline{e}_j.$$

Note that $P\underline{e}_j$ is just a re-arrangement of the entries in $\underline{e}_j$ and thus it is one of the other base vectors. In an efficient implementation there is no multiplication done in determining the vector which is described as $P\underline{e}_j$.

To summarize the statements in this section, we should not first compute an inverse matrix in order to help solve a linear system but we should instead use the techniques for solving linear systems to compute the inverse matrix.

The following is a Matlab script to check the claim that using the inverse matrix takes about 3 times longer to solve a large system.

```
% create a random n-by-n matrix
% and create a problem with  a known solution y
n=8000;
A=rand(n, n);
y=ones(n, 1);
b=A*y;

% use \ to solve the system and time it and check the accuracy
tic;
x=A\b;
e=toc;
acc=norm(x-y, inf);
fprintf('n=%d, time taken to solve=    %6.2f, accuracy=%10.2e\n', ...
         n, e, acc);
```

```
% repeat via first computing the inverse
tic;
x=inv(A)*b;
e=toc;
acc=norm(x-y, inf);
fprintf('n=%d, time taken using inv(A)=%6.2f, accuracy=%10.2e\n', ...
         n, e, acc);
```

The output generated on a laptop which was new in 2015 is given below with the time taken being in seconds.

```
n=8000, time taken to solve=      6.29, accuracy=  6.49e-11
n=8000, time taken using inv(A)= 17.23, accuracy=  5.97e-10
```

If you try this then you will get different numbers as the random matrix is likely to be different on each attempt and your computer speed is likely to be different. You should however observe that the approach involving computing the inverse matrix is slightly less accurate as well as taking about 3 times longer. The storage of the matrix with $n = 8000$ involves 64×10^6 entries with each entry requiring 8 bytes and thus it involves 512×10^6 bytes of the memory. If we double n to 16000 then the storage increases by a factor of 4 to 2048×10^6 bytes and the time taken increases by a factor of about 8 and this took 47 and 143 seconds respectively on the same computer. Problems of this size are approaching the limits of what can be done with such equipment in a reasonable time.

For some final comments about the inverse matrix recall that it was one of the terms in the definition of the matrix condition number, i.e.

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

(The matrix condition number was introduced in section 1.7.) The size of this number is relevant to how accurately we can expect to solve a linear system and to partly explain why this is the case consider the following linear systems.

$$\begin{aligned} A\underline{x} &= \underline{b}, \\ A(\underline{x}+\Delta\underline{x}) &= \underline{b}+\Delta\underline{b}. \end{aligned}$$

When $\|\Delta\underline{b}\|$ is very small the two systems have the same matrix and nearly the same right hand side and we would expect the solutions to be close and by subtracting the equations we immediately get that

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

and hence

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

To obtain the size of $\|\Delta\underline{x}\|$ relative to the size of $\|\underline{x}\|$ observe that as $\underline{b} = A\underline{x}$ it follows that

$$\|\underline{b}\| = \|A\underline{x}\| \le \|A\|\,\|\underline{x}\|, \quad \text{and this implies that } \frac{1}{\|\underline{x}\|} \le \frac{\|A\|}{\|\underline{b}\|}.$$

Combining the last two results gives

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

This shows that a small relative change in the right hand side vector $\underline{b}$ can lead to a much larger change in the solution when $\kappa(A)$ is large. A similar conclusion is reached if there is also a small change to the matrix although the details are much longer and are not done here.

2.7 Summary and some comments

If you have grasped the material in this chapter then it should have extended your knowledge of the problem of solving $A\underline{x} = \underline{b}$ when n is large so that hand calculations are no longer feasible and the computer has to be used. Some key theoretical points are as follows.

1. Linear systems with triangular matrices can be solved using backward substitution in the case of an upper triangular matrix and forward substitution in the case of a lower triangular matrix.
2. We can solve $LU\underline{x} = \underline{b}$ by solving $L\underline{y} = \underline{b}$ followed by solving $U\underline{x} = \underline{y}$. Hence if $A = LU$ then we can solve $A\underline{x} = \underline{b}$ quickly. If the factorization is $PA = LU$ then we similar consider $LU\underline{x} = P\underline{b}$.
3. A Gauss transformation matrix is of the form $M_k = I - \underline{m}_k \underline{e}_k^T$ with the first k entries of $\underline{m}_k$ being 0. These matrices have the properties that
$$M_k^{-1} = I + \underline{m}_k \underline{e}_k^T, \quad M_1^{-1} M_2^{-1} \cdots M_r^{-1} = I + \underline{m}_1 \underline{e}_1^T + \cdots + \underline{m}_r \underline{e}_r^T$$
with all the matrices being unit lower triangular.
4. Basic Gauss elimination involves no re-arrangement of the rows and when the algorithm runs to completion it is equivalent to a factorization $A = LU$ with $L = I + \underline{m}_1 \underline{e}_1^T + \cdots + \underline{m}_{n-1} \underline{e}_{n-1}^T$ being unit lower triangular and with U being upper triangular. When the factorization is possible we also have the factorization of all the principal sub-matrices. For a $n \times n$ matrix it takes $\mathcal{O}(2n^3/3)$ operations to compute the factorization with an operation meaning a multiplication or an addition/subtraction.
5. Partial pivoting involves swapping rows to give a more stable procedure and it corresponds to a factorization of the form $PA = LU$ where P is a permutation matrix.
6. Instead of getting A^{-1} and using this to solve a linear system we do the reverse and solve linear systems if we need to compute A^{-1} as the jth column of A^{-1} satisfies $A\underline{x}_j = \underline{e}_j$.

If more time was available then other direct methods involving factorising the matrix A could be considered. As one example, there is complete pivoting which involves possibly re-ordering both the rows of the matrix and the columns of the matrix and mathematically this corresponds to a factorization of the form

$$PAQ = LU$$

where now both P and Q are permutation matrices. This is more stable than the partial pivoting described earlier but the extra stability nearly doubles the time taken to solve a linear system and it is rarely necessary.

It is also possible to factorize A in a more stable way using special orthogonal matrices leading to a factorization of the form

$$A = QR$$

where here Q is an orthogonal matrix and R is an upper triangular matrix (the term right triangular is also used here with the factorization known as the QR factorization). Again the stability comes at a cost as it takes about twice as long to compute this compared with the time taken to compute the factorization $PA = LU$.

Much of the description has been concerned with general square matrices with no special properties and there was a comment that for basic Gauss elimination to work we need all leading principal sub-matrices to be non-singular. A common situation when we have this case is when the matrix A is symmetric and is positive definite which means that $\underline{x}^T A \underline{x} > 0$ for all $\underline{x} \neq \underline{0}$. The symmetric and positive definite property implies that all the eigenvalues of the matrix are positive and in particular such a matrix is non-singular. We can quickly deduce from this that all leading principal sub-matrices are also symmetric and positive definite, and hence non-singular, and we can factorize in a symmetric way as

$$A = LL^T,$$

where L now denotes a general lower triangular matrix with positive diagonal entries. This is known as a Cholesky factorization.

As a final comment, when the matrix is banded we can often factorise in a banded way and the case of a tri-diagonal linear system will be briefly considered in a later chapter when the finite difference method is used to approximately solve two-point boundary value problems.

Chapter 3

The problem $\underline{u}' = A\underline{u}$, $\underline{u}(0) = \underline{u}_0$

3.1 Introduction

In this chapter we consider how to solve a first order linear system of ordinary differential equations (ODEs) with constant coefficients which we write in a matrix-vector form as

$$\underline{u}' = A\underline{u}, \quad \underline{u}(0) = \underline{u}_0. \tag{3.1.1}$$

If $A = (a_{ij})$ is an $n \times n$ matrix then in full the differential equation part of this is

$$\frac{\mathrm{d}}{\mathrm{d}x}\begin{pmatrix} u_1(x) \\ \vdots \\ u_n(x) \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \cdots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}\begin{pmatrix} u_1(x) \\ \vdots \\ u_n(x) \end{pmatrix}.$$

The key result of the chapter is that in the cases that we consider the solution always exists and it can be expressed in terms of the eigenvalues and eigenvectors of A when A has a complete set of eigenvectors.

3.2 Using eigenvalues and eigenvectors

Let $\underline{v} \neq \underline{0}$ be an eigenvector of A with eigenvalue λ, i.e. $A\underline{v} = \lambda\underline{v}$. Also let

$$\underline{y}(x) = \mathrm{e}^{\lambda x}\underline{v}. \tag{3.2.1}$$

If we differentiate with respect to x then we get

$$\underline{y}'(x) = \lambda \mathrm{e}^{\lambda x}\underline{v}. \tag{3.2.2}$$

If we multiply the vector in (3.2.1) by the matrix A then we get

$$A\underline{y}(x) = \mathrm{e}^{\lambda x}A\underline{v} = \mathrm{e}^{\lambda x}\lambda\underline{v} \tag{3.2.3}$$

where the last equality follows the eigenvector property. Both (3.2.2) and (3.2.3) are the same and thus the vector in (3.2.1) satisfies the differential equation. As every eigenvalue/eigenvector pair gives a solution of the system of differential equations and as the differential equation

$$\underline{u}' - A\underline{u} = \underline{0}$$

is linear it follows that linear combinations of different solutions is also a solution. Thus if $\lambda_1, \lambda_2, \dots, \lambda_n$ denotes the eigenvalues of A and $\underline{v}_i \neq \underline{0}$ is an eigenvector associated with eigenvalue λ_i then

$$\underline{u}(x) = c_1 \mathrm{e}^{\lambda_1 x} \underline{v}_1 + \cdots + c_n \mathrm{e}^{\lambda_n x} \underline{v}_n \tag{3.2.4}$$

satisfies the differential equations for all values of $c_1, \dots, c_n$. What we have not yet done is to determine if we can also satisfy the initial condition $\underline{u}(0) = \underline{u}_0$ for some choice of $\underline{c} = (c_i)$. To make progress here note that when $x = 0$ we have

$$\underline{u}(0) = c_1 \underline{v}_1 + \cdots + c_n \underline{v}_n = \left(\underline{v}_1, \dots, \underline{v}_n\right) \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = V \underline{c}$$

where

$$V = \left(\underline{v}_1, \dots, \underline{v}_n\right).$$

(The matrix V with the eigenvectors as the columns is the same matrix as appeared in section 1.4.2.) Provided we can solve for $\underline{c}$ the equations

$$V \underline{c} = \underline{u}_0 \tag{3.2.5}$$

then (3.2.4) is the solution to (3.1.1). The linear system with the matrix V has a unique solution if V is invertible and this is the case if and only if the eigenvectors $\underline{v}_1, \dots, \underline{v}_n$ are linearly independent. This condition on the eigenvectors corresponds to the case that the matrix A is diagonalisable and a sufficient condition which guarantees this is when the eigenvalues $\lambda_1, \dots, \lambda_n$ are distinct.

We summarise the method.

1. Determine the eigenvalues and eigenvectors of A which we denote by $\lambda_1, \dots, \lambda_n$ and $\underline{v}_1, \dots, \underline{v}_n$.
2. Form the matrix $V = (\underline{v}_1, \dots, \underline{v}_n)$ and, if V is invertible, solve
$$V \underline{c} = \underline{u}_0.$$
3. If we obtained $\underline{c}$ in step 2 then the solution is given by
$$\underline{u}(x) = \sum_{i=1}^{n} c_i \mathrm{e}^{\lambda_i x} \underline{v}_i.$$

In the above we have a representation of the exact solution although numerical methods are needed to determine the eigenvalues and eigenvectors of A and the use of a computer helps to solve $V \underline{c} = \underline{u}_0$. When n is as small as 2 hand calculations can be done and we illustrate this next with some examples.

3.3 Examples with $n = 2$

In the following examples we start with two cases which can easily also be done by techniques that you have already met and hence you can quite quickly confirm the solution.

1. Consider the system

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad \underline{u}(0) = \begin{pmatrix} 2 \\ 4 \end{pmatrix}.$$

Here the matrix is

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

and the characteristic equation is

$$\det(A - \lambda I) = \lambda^2 - 1 = 0.$$

The eigenvalues are $\lambda_1 = -1$ and $\lambda_2 = 1$. To get the eigenvectors first consider

$$A - \lambda_1 I = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad A - \lambda_2 I = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}.$$

By inspection we can take the eigenvectors as

$$\underline{v}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \text{and} \quad \underline{v}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The general solution is

$$\underline{u}(x) = c_1 \mathrm{e}^{-x} \begin{pmatrix} 1 \\ -1 \end{pmatrix} + c_2 \mathrm{e}^{x} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

To satisfy the condition at $x = 0$ we need to solve

$$\underline{u}(0) = V\underline{c} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}.$$

If we add the two equations we get $2c_2 = 6$, $c_2 = 3$ and $c_1 = -1$.

If we consider the answer in components then

$$\begin{aligned} u_1(x) &= -\mathrm{e}^{-x} + 3\mathrm{e}^{x}, \\ u_2(x) &= \mathrm{e}^{-x} + 3\mathrm{e}^{x}. \end{aligned}$$

It is put in this form as you may note that the problem itself could have been tackled in a different way by noting that $u_1' = u_2$ so that $u_1'' = u_2'$ and from the other differential equation $u_2' = u_1$ and thus $u_1'' = u_1$. We similarly get $u_2'' = u_2$. By methods that you have done previously you you should know that the solution involves e^{-x} and e^{x}.

2. Consider now the system

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}' = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad \underline{u}(0) = \begin{pmatrix} 2 \\ 4 \end{pmatrix}.$$

Here the matrix is

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

and the characteristic equation is

$$\det(A-\lambda I)=\lambda^2+1=0.$$

The eigenvalues are the complex conjugate pair $\lambda_1=-i$ and $\lambda_2=i$. To get the eigenvectors first consider

$$A-\lambda_1 I=\begin{pmatrix} i & -1\\ 1 & i\end{pmatrix} \quad\text{and}\quad A-\lambda_2 I=\begin{pmatrix} -i & -1\\ 1 & -i\end{pmatrix}.$$

By inspection we can take the eigenvectors as

$$\underline{v}_1=\begin{pmatrix}1\\ -i\end{pmatrix} \quad\text{and}\quad \underline{v}_2=\begin{pmatrix}1\\ i\end{pmatrix}.$$

The general solution is

$$\underline{u}(x)=c_1\mathrm{e}^{-ix}\begin{pmatrix}1\\ -i\end{pmatrix}+c_2\mathrm{e}^{ix}\begin{pmatrix}1\\ i\end{pmatrix}.$$

To satisfy the condition at $x=0$ we need to need to solve

$$\underline{u}(0)=V\underline{c}=\begin{pmatrix}1 & 1\\ -i & i\end{pmatrix}\begin{pmatrix}c_1\\ c_2\end{pmatrix}=\begin{pmatrix}2\\ 4\end{pmatrix}.$$

If we multiply the first equation by i and add to the second equation we get

$$2ic_2=4+2i,\quad c_2=1-2i.$$

Then

$$c_1=2-c_2=1+2i.$$

The coefficients are a complex conjugate pair which is because the solution is real. From what has been done so far we can write the solution as

$$\underline{u}(x)=(1+2i)\mathrm{e}^{-ix}\begin{pmatrix}1\\ -i\end{pmatrix}+(1-2i)\mathrm{e}^{ix}\begin{pmatrix}1\\ i\end{pmatrix}.$$

To express it in a form only involving real quantities note that

$$\mathrm{e}^{-ix}=\cos x-i\sin x \quad\text{and}\quad \mathrm{e}^{ix}=\cos x+i\sin x.$$

and

$$\begin{aligned}(1+2i)(\cos x-i\sin x) &= (\cos x+2\sin x)+i(2\cos x-\sin x),\\ (1-2i)(\cos x+i\sin x) &= (\cos x+2\sin x)-i(2\cos x-\sin x).\end{aligned}$$

Hence

$$\begin{aligned}u_1(x) &= 2\cos x+4\sin x,\\ u_2(x) &= 4\cos x-2\sin x.\end{aligned}$$

As in the previous example this answer could have been obtained by noting that $u_1'=-u_2$ so that $u_1''=-u_2'=-u_1$ and similarly $u_2''=-u_2$. You should know that the solutions to these equations involve $\cos x$ and $\sin x$.

3. Consider now the system

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}' = \begin{pmatrix} 6 & 6 \\ -2 & -7 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \quad \underline{u}(0) = \begin{pmatrix} 20 \\ -7 \end{pmatrix}.$$

Here the matrix is

$$A = \begin{pmatrix} 6 & 6 \\ -2 & -7 \end{pmatrix}$$

and the characteristic equation is

$$\begin{aligned} \det(A - \lambda I) &= (6-\lambda)(-7-\lambda) + 12, \\ &= \lambda^2 + \lambda - 30 = (\lambda + 6)(\lambda - 5). \end{aligned}$$

The eigenvalues are $\lambda_1 = -6$ and $\lambda_2 = 5$. To get the eigenvectors first consider

$$A - \lambda_1 I = \begin{pmatrix} 12 & 6 \\ -2 & -1 \end{pmatrix} \quad \text{and} \quad A - \lambda_2 I = \begin{pmatrix} 1 & 6 \\ -2 & -12 \end{pmatrix}.$$

By inspection we can take the eigenvectors as

$$\underline{v}_1 = \begin{pmatrix} 1 \\ -2 \end{pmatrix} \quad \text{and} \quad \underline{v}_2 = \begin{pmatrix} -6 \\ 1 \end{pmatrix}.$$

The general solution is

$$\underline{u}(x) = c_1 \mathrm{e}^{-6x} \begin{pmatrix} 1 \\ -2 \end{pmatrix} + c_2 \mathrm{e}^{5x} \begin{pmatrix} -6 \\ 1 \end{pmatrix}.$$

To satisfy the condition at $x = 0$ we need to need to solve

$$\underline{u}(0) = V\underline{c} = \begin{pmatrix} 1 & -6 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 20 \\ -7 \end{pmatrix}.$$

If we use basic Gauss elimination to solve then we have

$$\begin{pmatrix} 1 & -6 & 20 \\ -2 & 1 & -7 \end{pmatrix} \to \begin{pmatrix} 1 & -6 & 20 \\ 0 & -11 & -33 \end{pmatrix}$$

giving $c_2 = -3$ and $c_1 = 20 + 6c_2 = 2$. Thus to summarize, the solution is

$$\underline{u}(x) = \mathrm{e}^{-6x} \begin{pmatrix} 2 \\ -4 \end{pmatrix} + \mathrm{e}^{5x} \begin{pmatrix} 18 \\ -3 \end{pmatrix}.$$

As some comments on the examples, in all cases we had distinct eigenvalues which guarantees that the matrix $V = (\underline{v}_1, \underline{v}_2)$ has linearly independent columns and is invertible. In one of the examples we had a complex conjugate pair of eigenvalues and the corresponding eigenvectors occurred as a complex conjugate pair. Examples with complex conjugate eigenvalues tend to be a longer to do by hand calculations to put the answer in real form and it does need knowledge of what the exponential means for a general complex number $\lambda = p + iq$ with $p, q \in \mathbb{R}$. We can take the following as the definition of the exponential term in this case.

$$\mathrm{e}^{\lambda x} = \mathrm{e}^{(p+iq)x} = \mathrm{e}^{px}\mathrm{e}^{iqx} = \mathrm{e}^{px}(\cos(qx) + i\sin(qx))$$

and hence

$$\mathrm{e}^{\overline{\lambda} x} = \mathrm{e}^{(p-iq)x} = \mathrm{e}^{px}\mathrm{e}^{-iqx} = \mathrm{e}^{px}(\cos(qx) - i\sin(qx)) = \overline{\mathrm{e}^{\lambda x}}.$$

Note that the magnitude in both cases is e^{px}, i.e. it just involves the real part of the eigenvalue.

3.4 The exponential matrix

If you refer to section 1.4 in the revision chapter then note that for a diagonalisable matrix we had that there exists a matrix $V = (\underline{v}_1, \ldots, \underline{v}_n)$ such that

$$V^{-1}AV = D \quad \text{or equivalently} \quad A = VDV^{-1}$$

with $D = \operatorname{diag}\{\lambda_1, \ldots, \lambda_n\}$ containing the eigenvalues and with $\underline{v}_1, \ldots, \underline{v}_n$ being the eigenvectors. We have a similar set-up here with

$$\begin{aligned}
\underline{u}(x) &= c_1 e^{\lambda_1 x}\underline{v}_1 + \cdots + c_n e^{\lambda_n x}\underline{v}_n \\
&= (\underline{v}_1, \ldots, \underline{v}_n)\begin{pmatrix} c_1 e^{\lambda_1 x} \\ \vdots \\ c_n e^{\lambda_n x}\end{pmatrix} \\
&= (\underline{v}_1, \ldots, \underline{v}_n)\begin{pmatrix} e^{\lambda_1 x} & & \\ & \ddots & \\ & & e^{\lambda_n x}\end{pmatrix}\begin{pmatrix} c_1 \\ \vdots \\ c_n\end{pmatrix} \\
&= V\exp(xD)\underline{c}
\end{aligned}$$

where

$$\exp(xD) = \begin{pmatrix} e^{\lambda_1 x} & & \\ & \ddots & \\ & & e^{\lambda_n x}\end{pmatrix}$$

is the exponential of the diagonal matrix D. Now as

$$V\underline{c} = \underline{u}_0, \quad \underline{c} = V^{-1}\underline{u}_0$$

and we get the representation

$$\underline{u}(x) = \left(V\exp(xD)V^{-1}\right)\underline{u}_0.$$

In the case of a diagonalisable matrix the quantity in the brackets is known as the exponential matrix $\exp(xA)$, i.e.

$$\exp(xA) = V\exp(xD)V^{-1}$$

and the solution of the ODEs can be expressed in the form

$$\underline{u}(x) = \exp(xA)\underline{u}_0.$$

It needs to be appreciated that this does not actually help us to solve the problem but it does give a neat way of describing the solution which is consistent with the scalar case $u_1' = a_{11}u_1$.

As a final comment, the exponential matrix is implemented in Matlab as the function `expm()` and it does also exist for square matrices which are are not diagonalisable. In fact if we let $B = xA$ then $\exp(B)$ can be expressed in other ways and in particular it has the series expansion

$$\exp(B) = I + B + \frac{1}{2}B^2 + \frac{1}{6}B^3 + \cdots + \frac{1}{n!}B^n + \cdots$$

which can be shown to converge for all matrices B and we have that

$$\underline{u}(x) = \exp(xA)\underline{u}_0$$

in all cases, i.e. diagonalisable or not. However when the matrix is not-diagonalisable we cannot so neatly compute things as above and we require what are known as Jordan blocks and for the solution $\underline{u}(x)$ this leads to terms such as $x\mathrm{e}^{\lambda_k x}$, $x^2\mathrm{e}^{\lambda_k x}$, etc. when λ_k is a repeated eigenvalue with insufficient eigenvectors. These more complicated theoretical cases will not be considered in this module although it is possible to describe the following 2×2 case without details which are very long.

Let

$$A = \begin{pmatrix} \lambda_1 & \alpha \\ 0 & \lambda_2 \end{pmatrix}$$

which has eigenvalues of λ_1 and λ_2. It is deficient only when we have $\lambda_2 = \lambda_1$ with $\alpha \ne 0$. To determine the form of $\exp(xA)$ in the deficient case we first consider the case when $\lambda_2 \ne \lambda_1$ and then take the limit of the expression as $\lambda_2 \to \lambda_1$. Suppose then that $\lambda_2 \ne \lambda_1$. To get the eigenvectors note that

$$A - \lambda_1 I = \begin{pmatrix} 0 & \alpha \\ 0 & \lambda_2-\lambda_1 \end{pmatrix} \quad \text{and} \quad A - \lambda_2 I = \begin{pmatrix} \lambda_1-\lambda_2 & \alpha \\ 0 & 0 \end{pmatrix}$$

and thus for the eigenvectors we can take

$$\underline{v}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \underline{v}_2 = \begin{pmatrix} \alpha \\ \lambda_2-\lambda_1 \end{pmatrix} \ne 0.$$

The matrix $V = (\underline{v}_1,\ \underline{v}_2)$ and the inverse V^{-1} are

$$V = \begin{pmatrix} 1 & \alpha \\ 0 & \lambda_2-\lambda_1 \end{pmatrix} \quad \text{and} \quad V^{-1} = \begin{pmatrix} 1 & \dfrac{-\alpha}{\lambda_2-\lambda_1} \\ 0 & \dfrac{1}{\lambda_2-\lambda_1} \end{pmatrix}.$$

If $D = \mathrm{diag}\,\{\lambda_1, \lambda_2\}$ then

$$V\exp(xD) = \begin{pmatrix} \mathrm{e}^{\lambda_1 x} & \alpha \mathrm{e}^{\lambda_2 x} \\ 0 & (\lambda_2-\lambda_1)\mathrm{e}^{\lambda_2 x} \end{pmatrix}, \quad V\exp(xD)V^{-1} = \begin{pmatrix} \mathrm{e}^{\lambda_1 x} & \alpha\left(\dfrac{\mathrm{e}^{\lambda_2 x}-\mathrm{e}^{\lambda_1 x}}{\lambda_2-\lambda_1}\right) \\ 0 & \mathrm{e}^{\lambda_2 x} \end{pmatrix}.$$

Letting $\lambda_2 \to \lambda_1$ gives

$$A = \begin{pmatrix} \lambda_1 & \alpha \\ 0 & \lambda_1 \end{pmatrix} \quad \text{with} \quad \exp(xA) = \begin{pmatrix} \mathrm{e}^{\lambda_1 x} & \alpha x \mathrm{e}^{\lambda_1 x} \\ 0 & \mathrm{e}^{\lambda_1 x} \end{pmatrix}.$$

The form of $\exp(xA)$ when $\lambda_2 = \lambda_1$ could also have been obtained with a similar length of workings using the series after first noting that

$$A = \begin{pmatrix} \lambda_1 & \alpha \\ 0 & \lambda_1 \end{pmatrix} \quad \text{gives} \quad A^n = \begin{pmatrix} \lambda_1^n & n\alpha\lambda_1^{n-1} \\ 0 & \lambda_1^n \end{pmatrix}.$$

There is a question on the first exercise sheet similar to this to get the expression for A^n.

3.5 Higher order systems of ODEs with constant coefficients

You should already be familiar with finding the general solution of differential equations of the form

$$y'' + b_1 y' + b_0 y = 0, \quad b_0 \text{ and } b_1 \text{ being constants.}$$

To solve this the technique to use is to consider a candidate solution of the form

$$y(x) = \mathrm{e}^{mx}$$

and substituting into the equation gives

$$\mathrm{e}^{mx}(m^2 + b_1 m + b_0) = 0$$

and this is true provided m is such that

$$m^2 + b_1 m + b_0 = 0,$$

which is known as the auxiliary equation. If we assume that the quadratic has distinct roots m_1 and m_2 then the general solution is of the form

$$y(x) = B_1 \mathrm{e}^{m_1 x} + B_2 \mathrm{e}^{m_2 x}, \tag{3.5.1}$$

where B_1 and B_2 are constants. This can be considered as a particular case of what has been given earlier in this chapter as we can convert this second order ODE into a system of first order ODEs by letting

$$u_1 = y, \quad u_2 = y'$$

and then $u_1' = y' = u_2$ and we get $u_2' = y'' = -b_1 y' - b_0 y = -b_1 u_2 - b_0 u_1$. If we write this in matrix vector notation then we have

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}' = \begin{pmatrix} u_2 \\ -b_0 u_1 - b_1 u_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -b_0 & -b_1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

As we have already given the general solution in (3.5.1) we can put this in vector notation as

$$\underline{u}(x) = \begin{pmatrix} y(x) \\ y'(x) \end{pmatrix} = B_1 \mathrm{e}^{m_1 x} \begin{pmatrix} 1 \\ m_1 \end{pmatrix} + B_2 \mathrm{e}^{m_2 x} \begin{pmatrix} 1 \\ m_2 \end{pmatrix}.$$

This is of the form

$$\underline{u}(x) = c_1 \mathrm{e}^{\lambda_1 x} \underline{v}_1 + c_2 \mathrm{e}^{\lambda_2 x} \underline{v}_2$$

with $c_1 = B_1$ and $c_2 = B_2$ and with $(1, m_k)^T$ being an eigenvector with $\lambda_k = m_k$ as the eigenvalue for $k = 1, 2$. To check the eigenvector part we just need to consider the product

$$\begin{pmatrix} 0 & 1 \\ -b_0 & -b_1 \end{pmatrix} \begin{pmatrix} 1 \\ m_k \end{pmatrix} = \begin{pmatrix} m_k \\ -b_0 - b_1 m_k \end{pmatrix}.$$

As m_k is a root of the quadratic we have

$$m_k^2 + b_1 m_k + b_0 = 0, \quad \text{which re-arranges to } -b_0 - b_1 m_k = m_k^2$$

and it follows that

$$\begin{pmatrix} 0 & 1 \\ -b_0 & -b_1 \end{pmatrix} \begin{pmatrix} 1 \\ m_k \end{pmatrix} = m_k \begin{pmatrix} 1 \\ m_k \end{pmatrix}$$

which confirms the eigenvalue/eigenvector property as required. Although we did not need to construct the characteristic equation for A here it does immediately follow that

$$\det(A-\lambda I) = \begin{vmatrix} -\lambda & 1 \\ -b_0 & -b_1-\lambda \end{vmatrix} = \lambda(b_1+\lambda)+b_0 = \lambda^2+b_1\lambda+b_0$$

and thus the characteristic equation is the same as the auxiliary equation.

The above can be generalised to an ODE of any order of the form

$$y^{(n)}+b_{n-1}y^{(n-1)}+\cdots+b_1y'+b_0y=0$$

where $b_0, b_1, \ldots, b_{n-1}$ are constants. We define u_1, $u_2, \ldots, u_n$ as follows.

$$\begin{aligned} u_1 &= y, \\ u_2 &= y' = u_1', \\ u_3 &= y'' = u_2', \\ \cdots & \quad \cdots \\ u_n &= y^{(n-1)} = u_{n-1}' \end{aligned}$$

and note that from the differential equation

$$u_n' = y^{(n)} = -b_0u_1 - b_1u_2 - \cdots - b_{n-1}u_{n-1}.$$

Thus we have

$$\underline{u}' = A\underline{u} \quad \text{with } A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \cdots & \cdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ -b_0 & -b_1 & \cdots & \cdots & -b_{n-1} \end{pmatrix}. \tag{3.5.2}$$

The characteristic equation of the matrix is the auxiliary equation and is given by

$$\lambda^n + b_{n-1}\lambda^{n-1} + \cdots + b_1\lambda + b_0 = 0. \tag{3.5.3}$$

If you search through books and you search on the internet then you will find such a matrix A in (3.5.2) referred to as the transpose of the companion matrix associated with the polynomial given on the left hand side of (3.5.3). Finding the roots of a polynomial can hence be converted to a problem of finding the eigenvalues of such a matrix and this is what is often done in computer packages to determine all the roots of a polynomial.

As a final point here about converting one high order ODE into a system of first order ODEs, the initial condition $\underline{u}(0)$ is a condition involving $y(0)$, $y'(0), \ldots, y^{(n-1)}(0)$, i.e. it involves the function and derivatives at $x=0$. This is an example of an initial value problem. This term will be seen again when numerical techniques are considered and in the next chapter the case of the extra conditions being at more than one point will also be considered when we consider the two point boundary value problem.

3.6 Summary

1. The problem $\underline{u}'(x) = A\underline{u}(x)$, $\underline{u}(0) = \underline{u}_0$, where $A = (a_{ij})$ is a $n \times n$ constant matrix, can be solved by a procedure which involves finding the eigenvalues and eigenvectors of the matrix A when A is a diagonalisable matrix. With the eigenvalues and eigenvectors found the general solution is given by
$$\underline{u}(x) = \sum_{i=1}^{n} c_i \mathrm{e}^{\lambda_i x} \underline{v}_i, \tag{3.6.1}$$
where $A\underline{v}_i = \lambda_i \underline{v}_i$, $i = 1, \ldots, n$. To obtain the particular solution satisfying $\underline{u}(0) = \underline{u}_0$ we must solve
$$V\underline{c} = \underline{u}_0, \quad \text{where } V = (\underline{v}_1, \ldots, \underline{v}_n).$$

2. With a real matrix A the eigenvalues and eigenvectors may be complex but they occur in complex conjugate pairs and the solution $\underline{u}(x)$ is real when $\underline{u}_0$ is real. To deal with the complex case note that
$$\begin{aligned} \exp((p+iq)x) &= \mathrm{e}^{px}\mathrm{e}^{iqx} = \mathrm{e}^{px}(\cos(qx) + i\sin(qx)), \\ \exp((p-iq)x) &= \mathrm{e}^{px}\mathrm{e}^{-iqx} = \mathrm{e}^{px}(\cos(qx) - i\sin(qx)) \\ &= \overline{\exp((p+iq)x)}. \end{aligned}$$

3. The behaviour of the general solution in (3.6.1) as $x \to \infty$ depends on the eigenvalues $\lambda_1, \ldots, \lambda_n$. If the real part of λ_i is negative then $\mathrm{e}^{\lambda_i x} \to 0$ as $x \to \infty$ and if this is the case for all the eigenvalues then $\underline{u}(x) \to \underline{0}$ as $x \to \infty$ for all values of $c_1, \ldots, c_n$.

4. With
$$A = VDV^{-1}, \quad D = \mathrm{diag}\{\lambda_1, \ldots, \lambda_n\}$$
and
$$\exp(xA) = V\exp(xD)V^{-1}$$
the solution can be represented in the form
$$\underline{u}(x) = \exp(xA)\underline{u}_0.$$
This does not help us solve the problem but it is neat way of expressing the solution and it helps to understand the structure of the problem.

   In all cases, diagonalisable or not,
$$\exp(xA) = I + xA + \frac{x^2}{2|}A^2 + \frac{x^3}{3!}A^3 + \cdots + \frac{x^m}{m!}A^m + \cdots .$$

5. A higher order ODE can be converted into a system of first order ODEs by letting $u_1 = y$, $u_2 = y'$, $u_3 = y''$, etc. so that $u_1' = u_2$, $u_2' = u_3$ etc..

Chapter 4

The finite difference method for the 2-point boundary value problem

4.1 Introduction

In the previous chapter linear ordinary differential equations (ODEs) of the form

$$\underline{u}' = A\underline{u}, \quad \underline{u}(0) = \underline{u}_0$$

were considered, with A being a $n \times n$ matrix of constants, and it was shown that the solution could be expressed in the form

$$\underline{u}(x) = \exp(xA)\underline{u}_0,$$

where in the case of a diagonalisable matrix

$$\begin{aligned} \exp(xA) &= V\exp(xD)V^{-1}, \quad V = (\underline{v}_1, \ldots, \underline{v}_n), \\ \exp(xD) &= \operatorname{diag}\left\{\mathrm{e}^{\lambda_1 x}, \ldots, \mathrm{e}^{\lambda_n x}\right\}. \end{aligned}$$

In the above $\underline{v}_i \neq \underline{0}$ is an eigenvector of A with eigenvalue λ_i. It was also shown that a linear higher order ODE with constant coefficients can also be converted into a system of first order ODEs and we get a similar closed form expression for the solution.

Closed form expressions for the solution of differential equations are actually rare and usually numerical methods are the only way to approximately solve a given problem. In this chapter we consider the finite difference technique which can be used in the case of the 2-point boundary value problem and comment at the end of the chapter on numerical methods for the initial value problem for more general problems than were described in the previous chapter.

The specific problem for most of this chapter is the following: Find $u(x)$, $a \le x \le b$, such that

$$u''(x) = p(x)u'(x) + q(x)u(x) + r(x), \quad a < x < b, \tag{4.1.1}$$

$$u(a) = g_1, \quad u(b) = g_2, \tag{4.1.2}$$

where $p(x)$, $q(x)$ and $r(x)$ are suitable functions defined in $[a, b]$ and $q(x) \ge 0$ on $[a, b]$. Suitable functions in this context means that they have sufficiently many bounded derivatives on $[a, b]$. The condition $q(x) \ge 0$ is a standard sufficient condition to ensure that a solution does exist. The solution to (4.1.1)-(4.1.2) depends on the functions $p(x)$, $q(x)$, $r(x)$ and on the boundary values g_1 and g_2 and it cannot in general be expressed in closed form.

4.2 Finite difference approximations

The method considered in the chapter to approximate the solution to (4.1.1)-(4.1.2) is the finite difference method and we start here by considering how to approximate derivatives using differences involving function values and the key mathematical tool to understand this material is Taylor expansions.

For this section let $u(x)$ denote any sufficiently differentiable function. We will consider Taylor expansions about a point and for this it is convenient here to introduce the points that we will consider later by introducing a uniform mesh of our interval $[a, b]$ involving $N + 1$ equally spaced points with spacing $h = (b - a)/N$ with the points being given by

$$x_i = a + ih, \quad i = 0, 1, \ldots, N. \tag{4.2.1}$$

To shorten the expressions we let $u_i = u(x_i)$, $u_i' = u'(x_i)$, $u_i'' = u''(x_i)$, etc.. With these abbreviations we have the following Taylor expansions about the point x_i:

$$\begin{aligned} u_{i+1} &= u(x_i + h) = u_i + hu_i' + \frac{h^2}{2!}u_i'' + \frac{h^3}{3!}u_i''' + \frac{h^4}{4!}u_i'''' \\ &\quad + \cdots \end{aligned} \tag{4.2.2}$$

$$\begin{aligned} u_{i-1} &= u(x_i - h) = u_i - hu_i' + \frac{h^2}{2!}u_i'' - \frac{h^3}{3!}u_i''' + \frac{h^4}{4!}u_i'''' \\ &\quad + \cdots \end{aligned} \tag{4.2.3}$$

Here we have related $u_{i+1} = u(x_{i+1})$ and $u_{i-1} = u(x_{i-1})$ to u and its derivatives at the nearby point x_i. We can combine these in different ways as follows. If we add (4.2.2) and (4.2.3) then all the terms with odd order derivatives cancel and if we subtract (4.2.3) from (4.2.2) then all the terms with even order derivatives cancel. In the adding case we have

$$u_{i+1} + u_{i-1} = 2u_i + h^2 u_i'' + \frac{h^4}{12}u_i'''' + \cdots$$

and in the subtracting case we have

$$u_{i+1} - u_{i-1} = 2hu_i' + \frac{h^3}{3}u_i''' + \cdots .$$

By rearranging these we get expressions for u_i'' and u_i' given by

$$u_i'' = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - \frac{h^2}{12}u_i'''' + \cdots \tag{4.2.4}$$

and

$$u_i' = \frac{u_{i+1} - u_{i-1}}{2h} - \frac{h^2}{6}u_i''' + \cdots . \tag{4.2.5}$$

The central difference approximation to u_i'' is given by

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

and the central difference approximation to u_i' is given by

$$\frac{u_{i+1}-u_{i-1}}{2h}$$

and the error in both of these is written as $\mathcal{O}(h^2)$. This order notation is used frequently in this context when the actually expression involved is of the form ch^2, where c does not involve h, and we are not too interested in the details of the term c.

If you are someone who likes a bit more analysis then we can do all of the above a bit more rigorously by using Taylor's series with a remainder term at each step and we repeat this next and start with the assumption that $u(x)$ is 4-times continuously differentiable. Instead of (4.2.2) and (4.2.3) we have

$$u_{i+1} = u(x_i+h) = u_i + hu_i' + \frac{h^2}{2!}u_i'' + \frac{h^3}{3!}u_i''' + \frac{h^4}{4!}u''''(\xi_1) \tag{4.2.6}$$

$$u_{i-1} = u(x_i-h) = u_i - hu_i' + \frac{h^2}{2!}u_i'' - \frac{h^3}{3!}u_i''' + \frac{h^4}{4!}u''''(\xi_2) \tag{4.2.7}$$

where $\xi_1 \in (x_i, x_{i+1})$ and $\xi_2 \in (x_{i-1}, x_i)$. Then

$$u_i'' = \frac{u_{i+1}-2u_i+u_{i-1}}{h^2} - \frac{h^2}{24}\left(u''''(\xi_1)+u''''(\xi_2)\right).$$

The error term as given is a bit awkward but it can be tidied up by using the intermediate value theorem by considering the continuous function

$$f(t) = u''''(t) - \frac{1}{2}\left(u''''(\xi_1)+u''''(\xi_2)\right).$$

Observe that

$$f(\xi_1) = \frac{1}{2}\left(u''''(\xi_1)-u''''(\xi_2)\right) = -f(\xi_2).$$

The continuous function hence changes sign between ξ_2 and ξ_1 and thus there exists

$$\xi \in (\xi_2, \xi_1) \subset (x_{i-1}, x_{i+1})$$

with $f(\xi)=0$, i.e.

$$u''''(\xi) = \frac{1}{2}\left(u''''(\xi_1)+u''''(\xi_2)\right)$$

and we can write

$$u_i'' = \frac{u_{i+1}-2u_i+u_{i-1}}{h^2} - \frac{h^2}{12}u''''(\xi).$$

Similar reasoning leads to the existence of $\alpha \in (x_{i-1}, x_{i+1})$ such that

$$u_i' = \frac{u_{i+1}-u_{i-1}}{2h} - \frac{h^2}{6}u'''(\alpha).$$

4.3 The FDM for the two-point BVP

In the following U_i will denote a finite difference approximation to $u_i = u(x_i)$, i.e. we use upper case letters for the approximation and lower case letters for the exact solution to the differential equation. For the uniform mesh $a = x_0 < x_1 < \cdots < x_N = b$ described in (4.2.1) we have $N + 1$ such values corresponding to $i = 0, 1, \ldots, N$. To motivate how these are going to be defined consider replacing the derivatives in (4.1.1) by difference expressions for the interior mesh points corresponding to $1 \le i \le N - 1$. With $p_i = p(x_i)$, $q_i = q(x_i)$ and $r_i = r(x_i)$ we have

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = p_i \left(\frac{u_{i+1} - u_{i-1}}{2h} \right) + q_i u_i + r_i + \mathcal{O}(h^2). \tag{4.3.1}$$

The exact values satisfy this equation but it involves a term $\mathcal{O}(h^2)$ which we do not know precisely. In a finite difference scheme we omit the $\mathcal{O}(h^2)$ term and our main requirement on the terms $U_0, U_1, \ldots, U_N$ are that they satisfy

$$\begin{aligned} &\frac{U_{i+1} - 2U_i + U_{i-1}}{h^2} = p_i \left(\frac{U_{i+1} - U_{i-1}}{2h} \right) + q_i U_i + r_i, \\ &i = 1, 2, \ldots, N - 1. \end{aligned} \tag{4.3.2}$$

This gives $N - 1$ equations with $N + 1$ unknowns. For the two additional equations we note (4.1.2) and impose the conditions that

$$U_0 = g_1 \quad \text{and} \quad U_N = g_2,$$

i.e. the approximate solution exactly satisfies the boundary conditions at $x = a$ and at $x = b$. Thus with U_0 and U_N known (4.3.2) give $N - 1$ equations which we need to solve to get $U_1, \ldots, U_{N-1}$ and we consider this below. First however we note some terminology associated with approximating (4.3.1) by (4.3.2). The quantity

$$L_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - \left(p_i \left(\frac{u_{i+1} - u_{i-1}}{2h} \right) + q_i u_i + r_i \right) = \mathcal{O}(h^2)$$

is often referred to as the local truncation error and it indicates how closely the exact values satisfy the difference equations.

We now re-arrange (4.3.2) into the the linear equations that we solve to get the approximations $U_1, \ldots, U_{N-1}$. If we multiply (4.3.2) by h^2 and collect together all the parts involving U_{i-1}, U_i and U_{i+1} on the left hand side then we first get

$$U_{i-1} - 2U_i + U_{i+1} = \frac{hp_i}{2}(U_{i+1} - U_{i-1}) + q_i h^2 U_i + h^2 r_i$$

and finally we have

$$\begin{aligned} &\left(-1 - \frac{hp_i}{2} \right) U_{i-1} + \left(2 + h^2 q_i \right) U_i + \left(-1 + \frac{hp_i}{2} \right) U_{i+1} = -h^2 r_i, \\ &i = 1, 2, \ldots, N - 1. \end{aligned}$$

When $i = 1$ the equation involves $U_0 = g_1$ and putting this on the right hand side gives

$$(2 + h^2 q_1) U_1 + \left(-1 + \frac{hp_1}{2} \right) U_2 = -h^2 r_1 + \left(1 + \frac{hp_1}{2} \right) g_1.$$

When $i = N-1$ the equation involves $U_N = g_2$ and putting this on the right hand side gives

$$\begin{aligned}\left(-1-\frac{hp_{N-1}}{2}\right)U_{N-2} + \left(2+h^2q_{N-1}\right)U_{N-1}\\ = -h^2r_{N-1} + \left(1-\frac{hp_{N-1}}{2}\right)g_2.\end{aligned}$$

The equations when $i = 1$ and $i = N-1$ only involve 2 unknowns whilst the ones for $i = 2,\dots,N-2$ involve 3 unknowns. We have a tri-diagonal linear system for $\underline{U} = (U_1,\dots,U_{N-1})^T$ of the form

$$A\underline{U} = \underline{c} \tag{4.3.3}$$

where the matrix is given by

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & a_{N-2,N-1} \\ 0 & \cdots & 0 & a_{N-1,N-2} & a_{N-1,N-1} \end{pmatrix},$$

$$a_{i,i-1} = -1-\frac{hp_i}{2},\quad a_{ii} = 2+h^2q_i,\quad a_{i,i+1} = -1+\frac{hp_i}{2} \tag{4.3.4}$$

and the right hand side vector $\underline{c} = (c_i)$ is such that

$$c_1 = -h^2r_1 + \left(1+\frac{hp_1}{2}\right)g_1, \tag{4.3.5}$$

$$c_i = -h^2r_i, \quad 2 \le i \le N-2, \tag{4.3.6}$$

$$c_{N-1} = -h^2r_{N-1} + \left(1-\frac{hp_{N-1}}{2}\right)g_2. \tag{4.3.7}$$

One particular case of the above worth noting is when the function $p(x) = 0$ so that in the numerical scheme $p_1 = p_2 = \cdots = p_{N-1} = 0$. The matrix A simplifies to

$$A = \begin{pmatrix} 2+h^2q_1 & -1 & & & & \\ -1 & 2+h^2q_2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2+h^2q_{N-2} & -1 \\ & & & & -1 & 2+h^2q_{N-1} \end{pmatrix}. \tag{4.3.8}$$

If you recall that it was stated that $q(x) \ge 0$ is a sufficient condition to guarantee the solution of the two-point boundary value problem and it can be shown that this also guarantees that the matrix above is non-singular and satisfies what is known as a positive definite property. We will not pursue these aspects here other than to note that there is a solution to the linear system to determine $\underline{U}$.

4.3.1 Computational resources

We can store the tri-diagonal matrix A in an efficient way by just storing the sub-diagonal, diagonal and super-diagonal entries and we can solve the system efficiently by Gauss elimination by noting that when we eliminate below the diagonal in each column there is only one entry to eliminate at each step. We show next how this can be done and we then indicate what is already available in Matlab.

A tri-diagonal solver involving $\mathcal{O}(m)$ storage and operations

It is customarily when we have tri-diagonal matrices to adopt a one-subscript notation (a_i, d_i and b_i) for the entries instead of the double subscript notation a_{ij}. In the following m is the size of the system (thus $m = N-1$ when we relate this to the two-point boundary value problem). Thus we will write the system as

$$\begin{aligned} d_1U_1 + b_2U_2 &= c_1 \\ a_{i-1}U_{i-1} + d_iU_i + b_{i+1}U_{i+1} &= c_i, \quad 2 \le i \le m-1, \\ a_{m-1}U_{m-1} + d_mU_m &= c_m \end{aligned}$$

or as

$$\begin{pmatrix} d_1 & b_2 & \cdots & & & 0 \\ a_1 & d_2 & b_3 & & & \vdots \\ \vdots & a_2 & d_3 & b_4 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & b_m \\ 0 & \cdots & & & a_{m-1} & d_m \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ \vdots \\ U_{m-1} \\ U_m \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{m-1} \\ c_m \end{pmatrix}.$$

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

$$d_2'U_2 + b_3U_3 = c_2', \quad \text{where} \quad l_1 = a_1/d_1, \; d_2' = d_2 - l_1b_2, \; c_2' = c_2 - l_1c_1 \; .$$

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

$$\begin{array}{lll} l_2 = a_2/d_2', & d_3' = d_3 - l_2b_3, & c_3' = c_3 - l_2c_2, \\ l_3 = a_3/d_3', & d_4' = d_4 - l_3b_4, & c_4' = c_4 - l_3c_3. \\ \cdots & \cdots & \cdots \end{array}$$

With $d_1' = d_1$ and $c_1' = c_1$ we hence have

$$l_k = a_k/d_k', \quad d_{k+1}' = d_{k+1} - l_kb_{k+1}, \quad c_{k+1}' = c_{k+1} - l_kc_k', \\ k = 1, 2, \ldots, m-1 \; .$$

In matrix form we have created the upper triangular system

$$\begin{pmatrix} d_1' & b_2 & \cdots & & & 0 \\ & d_2' & b_3 & & & \vdots \\ & & d_3' & b_4 & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & b_m \\ 0 & \cdots & & & & d_m' \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ \vdots \\ U_{m-1} \\ U_m \end{pmatrix} = \begin{pmatrix} c_1' \\ c_2' \\ \vdots \\ \vdots \\ c_{m-1}' \\ c_m' \end{pmatrix}$$

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

$$\begin{aligned} U_m &= c_m'/d_m' \\ U_i &= (c_i' - b_{i+1}U_{i+1})/d_i' \quad \text{for } i = m-1, m-2, \ldots, 1. \end{aligned}$$

Hence to summarize this an an algorithm we have the following.

```
$d_1' = d_1$, $c_1' = c_1$.
For $k = 1, 2, \ldots, m-1$.
    $l_k = a_k/d_k'$.
    $d_{k+1}' = d_{k+1} - l_k b_{k+1}$.
    $c_{k+1}' = c_{k+1} - l_k c_k'$.
End For loop
$U_m = c_m'/d_m'$.
For $i = m-1, \ldots, 1$.
    $U_i = (c_i' - b_{i+1}U_{i+1})/d_i'$
End For loop
```

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

Using `spdiags` in Matlab to have $\mathcal{O}(m)$ storage and $\mathcal{O}(m)$ operations

Whichever programming language is used there are not too many statements needed to implement the algorithm in the previous subsection. Matlab has a number of functions which deal with sparse matrices and in particular the function `spdiags` is available to quickly set-up banded matrices. If, as in the last subsection, m is number of unknowns (we write `m` in the statements) and in Matlab `a`, `d` and `b` are column vectors of length m then we can set-up the matrix in sparse form by the statement

```
A=spdiags([a, d, b], -1:1, m, m);
```

Here `-1:1` means `[-1, 0, 1]` and the statement indicates that the first `m-1` entries of `a` are on the band below the diagonal, `d` contains the `m` diagonal entries and the last `m-1` entries of `b` are on the band above the diagonal. The part `[a, d, b]` is a $m \times 3$ matrix and note that the last entry of `a` and the first entry of `b` are not used. With `A` sparsely stored and with `c` being a column vector of length `m` the Matlab statement

```
UU=A\c;
```

efficiently solves the equations with an algorithm which involves $\mathcal{O}(m)$ operations.

A less efficient implementation in Matlab using `diag`

Matlab also has a builtin command called `diag` which can be used to create diagonal matrices or more generally a matrix with one non-zero band parallel with the main diagonal. Thus if `a`, `d` and `b` are as in the last subsection then the tri-diagonal matrix can be created with the command

```
A=diag(a(1:m-1), -1) + diag(d) + diag(b(2:m), 1);
```

which involves the sum of 3 matrices. When `diag` is used with 2 arguments the 2nd argument indicates which band to use and when the band is the sub-diagonal or super-diagonal it is necessary to give exactly `m`-1 entries.

The difference between the set-up with `diag` and the set-up using `spdiags` is that in this case `A` is stored as a full matrix whereas in the previous case only the non-zero entries are stored. Hence this version is wasteful in resources when `m` is large. The time taken to solve the linear system is also much greater although it is not as long as you might initially think from what was given in chapter 2 about solving linear equations. When you use \ in a statement such as `UU=A\c` Matlab first does some checks on the matrix in order to select what it considers to be the best technique to use. In this case it detects that the matrix is tri-diagonal and it uses an efficient technique but this is after $\mathcal{O}(m^2)$ checks. Thus, to summarize, this approach is inefficient in storage and in the number of operations compared with the approach using `spdiags` as they are both $\mathcal{O}(m^2)$ compared with $\mathcal{O}(m)$ in the sparse version. This may not matter too much if you only wish to solve one system and the speed of the computer is such that it is 0.01 seconds with the sparse version and 1 second with the full version. However if this or similar linear systems are to be solved a 1000 times in an application or if m needs to be much larger then the difference becomes significant.

4.3.2 Experiments indicating the convergence of the method

In the derivation of the finite difference scheme there was equation (4.3.1) that the exact values satisfied followed by the system (4.3.2) which defined the finite difference scheme used in the remaining parts of this chapter. From this it is possible to get equations for the errors $u_i - U_i$ and from this it can be shown that the method does converge as the number of intervals $N \to \infty$. It is usual to describe the convergence in terms of the mesh parameter

$$h = \frac{b-a}{N} > 0.$$

What can be shown is that as $h \to 0$ the errors behave like

$$|u_i - U_i| = \mathcal{O}(h^2)$$

provided all the conditions mentioned earlier are satisfied and in particular that the exact solution $u(x)$ is 4-times continuously differentiable. If N is sufficiently large then when we double N the error on average decreases is about $1/4$ of what it was before and similarly if we increase N by a factor of 10 then the error is about $1/100$ of what it was, i.e. we gain about two more decimal digits of accuracy. We can check these claims on a manufactured problem with a known exact solution as follows.

We take the interval $[a, b] = [0, \pi]$, we take $p(x) = q(x) = 0$ and we take $r(x)$ so that the exact solution is

$$u(x) = \exp(0.1x)\cos(3x).$$

In this simplified case the equation is just

$$u'' = r, \quad \text{with } r = \exp(0.01x)((0.01 - 9)\cos(3x) - 0.6\sin(3x))$$

and the boundary conditions are $u(0) = g_1 = 1$ and $u(\pi) = g_2 = -\exp(0.1\pi)$. A complete short Matlab script which implements the finite difference scheme for $N = 8, 16, \dots, 2^{14} = 16384$ and compares the approximate solution with the known exact solution follows.

```
% Manufactured problem u''=r, 0<x<pi
% with known solution uex
uex =@(x) exp(0.1*x).*cos(3*x);
r   =@(x) exp(0.1*x).*( (0.01-9)*cos(3*x)-0.6*sin(3*x) );

% set the interval [a, b] and boundary values of U
a=0;
b=pi;
g1=uex(a);
g2=uex(b);

fprintf('%5s  %12s %10s\n', 'N', 'error', 'ratio');
er=zeros(12, 1);

% loop to get the solution for larger and larger N
for k=1:12
  % set N, h and the uniform mesh points as x
  N=2^(2+k);
  x=linspace(a, b, N+1);
  x=x(:);
  xx=x(2:N);
  h=(b-a)/N;

  % set-up the tri-diagonal matrix A
  N1=N-1;
  o=ones(N1, 1);
  A=spdiags([-o, 2*o, -o], -1:1, N1, N1);

  % set-up the rhs vector c
  c=-h*h*r(xx);
  c(1)=c(1)+g1;
  c(N1)=c(N1)+g2;

  % get the fd solution U and save the error
  U=[g1; zeros(N1, 1); g2];
  U(2:N)=A\c;
  er(k)=norm(U-uex(x), inf);

  % show the error and the error ratio
  if k>1
    rat=er(k-1)/er(k);
    fprintf('%5d  %12.3e %10.7f\n', N, er(k), rat);
  else
    fprintf('%5d  %12.3e\n', N, er(k));
  end
end
```

The output generated is as follows.

```
    N         error      ratio
    8     2.152e-01
   16     5.473e-02  3.9314256
   32     1.350e-02  4.0528885
   64     3.376e-03  4.0002883
  128     8.435e-04  4.0024833
  256     2.109e-04  3.9994355
  512     5.272e-05  4.0002049
 1024     1.318e-05  4.0000512
 2048     3.295e-06  3.9999897
 4096     8.238e-07  3.9999987
 8192     2.059e-07  4.0000128
16384     5.147e-08  4.0009746
```

The output clearly shows that the ∞-norm of the exact error does indeed decrease like $\mathcal{O}(h^2)$ as the ratios of the errors are close to 4.

4.4 Remarks about other boundary conditions

In some applications we have ODEs of the form (4.1.1) with a boundary condition at one or more of the end points a and b involving a derivative of u, e.g. $u'(a) = g_3$ of $u'(b) = g_3$. In full this type of problem is either of the form

$$\begin{aligned} u''(x) &= p(x)u'(x) + q(x)u(x) + r(x), \quad a < x < b, \\ u'(a) &= g_3, \quad u(b) = g_2, \end{aligned}$$

or

$$\begin{aligned} u''(x) &= p(x)u'(x) + q(x)u(x) + r(x), \quad a < x < b, \\ u(a) &= g_1, \quad u'(b) = g_3. \end{aligned}$$

This complicates things a little but it still possible to adapt the method and retain the $\mathcal{O}(h^2)$ accuracy provided we have a difference approximation for derivative boundary condition of sufficient accuracy which we can do as follows.

Consider again Taylor series expansions about $x_N = b$ for $u(x_{N-1}) = u(b-h)$ and now also for $u(x_{N-2}) = u(b-2h)$.

$$\begin{aligned} u_{N-1} &= u_N - hu'_N + \frac{h^2}{2}u''_N - \frac{h^3}{6}u'''_N + \cdots \\ u_{N-2} &= u_N - 2hu'_N + \frac{4h^2}{2}u''_N - \frac{8h^3}{6}u'''_N + \cdots . \end{aligned}$$

We want to combine these to eliminate the u''_N term and hence we take

$$4u_{N-1} - u_{N-2} = 3u_N - 2hu'_N + \frac{4h^3}{6}u'''_N + \cdots$$

and this rearranges to give

$$u'_N = \frac{u_{N-2} - 4u_{N-1} + 3u_N}{2h} + \frac{h^2}{3}u'''_N + \cdots .$$

This motivates that to approximate the boundary condition $u'(b) = g_3$ we take

$$\frac{U_{N-2} - 4U_{N-1} + 3U_N}{2h} = g_3.$$

To use this we need to add it to the equations to be solved as now U_N is also an unknown and if the other boundary condition is still $U_0 = g_1$ then we have a linear system for $U_1, \ldots, U_N$ although we do not consider the details further here in these notes.

4.5 Summary

1. An exact solution of a ODE in closed form is not usually possible and numerical techniques are needed to approximate the solution.

2. Using Taylor's series gives the following

$$\begin{aligned} u_{i+1} &= u(x_i + h) = u_i + hu'_i + \frac{h^2}{2!}u''_i + \frac{h^3}{3!}u'''_i + \frac{h^4}{4!}u''''_i + \cdots \\ u_{i-1} &= u(x_i - h) = u_i - hu'_i + \frac{h^2}{2!}u''_i - \frac{h^3}{3!}u'''_i + \frac{h^4}{4!}u''''_i + \cdots \end{aligned}$$

   The central difference approximations for u'_i and u''_i that are used in the finite difference scheme are

$$\frac{U_{i+1} - U_{i-1}}{2h} \quad \text{and} \quad \frac{U_{i+1} - 2U_i + U_{i-1}}{h^2}.$$

   Both of these have an accuracy which is written as $\mathcal{O}(h^2)$.

3. The finite difference scheme described in this chapter for the two-point boundary value problem gives a tri-diagonal linear system to solve and the storage of the matrix and the amount of computation are both $\mathcal{O}(N)$ when we have $N + 1$ mesh points in $[a, b]$. The error in the basic scheme is $\mathcal{O}(h^2)$ where $h = (b - a)/N$ is the mesh spacing in a uniform mesh. We can implement this efficiently in Matlab using `spdiags` when setting up the matrix so that sparse storage mode is used.

4. We can also deal with derivative boundary conditions but we need to use a sufficient number of points to retain the $\mathcal{O}(h^2)$ accuracy. In the case of approximating $u'(b) = g_3$ this involves

$$\frac{U_{N-2} - 4U_{N-1} + 3U_N}{2h} = g_3.$$

   This was derived by considering the Taylor expansions of $u_{N-1} = u(b - h)$ and $u_{N-2} = u(b - 2h)$ about the point b.

4.6 Remarks about initial value problems

If more time was available then numerical methods for the initial value problem

$$u' = f(t, u(t)), \quad u(t_0) = u_0 \qquad \text{or} \qquad \underline{u}' = \underline{f}(t, \underline{u}(t)), \quad \underline{u}(t_0) = \underline{u}_0 \tag{4.6.1}$$

would be given. Here we are now using t for the variable as in applications this denotes time. In (4.6.1) the first case is the scalar case and the other case is the vector case. With suitable properties on f or $\underline{f}$ it can be proved that a solution exists in a vicinity of $t = t_0$ but it may not always exist for all t. The chapter about $\underline{u}' = A\underline{u}$ was a special case of the vector case corresponding to $\underline{f}(t, \underline{u}(t)) = A\underline{u}(t)$ with A being a matrix of constants.

As this section just contains comments on methods in this context we restrict these comments to the scalar case. Firstly, as an attempt to indicate the complexity that can arise consider what is involved to get expressions for the time derivatives of $u(t)$. For example, we might wish to consider the Taylor expansion about t_0 as we know $u(t_0) = u_0$. The first derivative is given by the differential equation, i.e.

$$u'(t) = f(t, u(t)).$$

To get the second derivative with respect to t we need to use the chain rule and we get

$$\begin{aligned} u''(t) &= \frac{\partial f}{\partial t}(t, u(t))) + \frac{\partial f}{\partial u}(t, u(t))u'(t) \\ &= \frac{\partial f}{\partial t}(t, u(t)) + \frac{\partial f}{\partial u}(t, u(t))f(t, u(t)). \end{aligned}$$

It is getting a bit cumbersome to show the evaluation point $(t, u(t))$ for each term and thus we shorten to just putting

$$u'' = \frac{\partial f}{\partial t} + f\frac{\partial f}{\partial u}. \tag{4.6.2}$$

To get the next derivative we need to now use both the chain rule and the product rule and this gives

$$\begin{aligned} u''' &= \left(\frac{\partial^2 f}{\partial t^2} + \frac{\partial^2 f}{\partial u \partial t}u'\right) + \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}u'\right)\frac{\partial f}{\partial u} + f\left(\frac{\partial^2 f}{\partial t \partial u} + \frac{\partial^2 f}{\partial u^2}u'\right) \\ &= \frac{\partial^2 f}{\partial t^2} + 2f\frac{\partial^2 f}{\partial u \partial t} + \left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial u}\right)\frac{\partial f}{\partial u} + f\frac{\partial^2 f}{\partial u^2} \end{aligned} \tag{4.6.3}$$

where the last line is obtained be replacing u' by f and collecting together some of the terms. As this illustrates each new derivative gets progressively more complicated and we stop at this point.

There is a class of methods which are known as Taylor series methods which needs the expressions for derivatives of u but they are not used in practice as they require that partial derivatives of f are determined whereas other methods of comparable accuracy only need that we can evaluate $f(t, v)$ for any values of t and v. A particular well known class of such methods are Runge Kutta methods which we just state.

In the following let h be small and let $t_i = t_0 + ih$ so that $t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} \cdots$ denote equally spaced discrete times. In all the methods we start by taking $U_0 = u_0$.

Euler's method involves defining U_1, $U_2, \dots$ by

$$U_{n+1} = U_n + hf(t_n, U_n), \quad n = 0, 1, 2, \dots. \tag{4.6.4}$$

When we start we make an error of magnitude $\mathcal{O}(h^2)$ when we obtain U_1 and the terminology is that this is the local truncation error. If we require N steps to reach a final time $t_f = t_0 + Nh$ then the accumulation of errors of magnitude $\mathcal{O}(h^2)$ at each step is that the error in the approximation $U_N \approx u_n = u(t_f)$ has an error of order $\mathcal{O}(h)$ as

$$Nh^2 = (t_f - t_0)h.$$

To describe the operations in the form of an algorithm or pseudo code can be done as follows.

```
For $n = 0, 1, 2, \dots$
    $k_1 = f(t_n, U_n)$,
    $U_{n+1} = U_n + hk_1$.
End For loop
```

One of the Runge Kutta methods of order 2 known as Huen's involves the following.

```
For $n = 0, 1, 2, \dots$
    $k_1 = f(t_n, U_n)$,
    $k_2 = f(t_n + h, U_n + hk_1)$,
    $U_{n+1} = U_n + \frac{h}{2}(k_1 + k_2)$.
End For loop
```

This involves 2 function evaluations to get the approximation U_{n+1} once U_n is already known. If we require N steps to reach a final time $t_f = t_0 + Nh$ then the accumulation of the errors $U_N \approx u_n = u(t_f)$ has an error of order $\mathcal{O}(h^2)$.

The most well known of the Runge Kutta methods is the Runge Kutta method of order 4 and this involves the following.

```
For $n = 0, 1, 2, \dots$
    $k_1 = f(t_n, U_n)$,
    $k_2 = f(t_n + h/2, U_n + (h/2)k_1)$,
    $k_3 = f(t_n + h/2, U_n + (h/2)k_2)$,
    $k_4 = f(t_n + h, U_n + hk_3)$,
    $U_{n+1} = U_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$.
End For loop
```

This involves 4 function evaluations to get the approximation U_{n+1} once U_n is already known. If we require N steps to reach a final time $t_f = t_0 + Nh$ then the accumulation of the errors $U_N \approx u_n = u(t_f)$ has an error of order $\mathcal{O}(h^4)$. This method was developed over 100 years ago and it is still used today.

Chapter 5

Fourier series

5.1 Introduction and periodic functions

Fourier series is concerned with representing a periodic function in terms of appropriate sine and cosine terms and the series are named after Joseph Fourier (1768–1830). Joseph Fourier developed the series in his work to solve the heat equation which is a partial differential equation which governs how the temperature in a body changes in both space and time. As part of `MA2715` there is not enough time to consider this application but we will consider how to obtain Fourier series for several different functions, we consider some manipulations with the series such as term-by-term integration and differentiation and we will state sufficient conditions for the series to be the same as the function it represents.

A periodic function $f(x)$ is one those graph repeats itself and in terms of a formula a function has period $T > 0$ if

$$f(x+T) = f(x), \quad \text{for all } x \in \mathbb{R}.$$

The smallest value of $T > 0$ for which this is true is sometimes called the least period or the fundamental period although when the context is clear some texts just use the term period to mean the least period. For example, the least period of $\cos\theta$ and $\sin\theta$ is 2π, i.e.

$$\cos(\theta + 2\pi) = \cos(\theta), \quad \sin(\theta + 2\pi) = \sin(\theta),$$

and there is no smaller number such that we have these relations. As Fourier series involve expansions in terms of cosines and sines it is convenient in terms of simplifying the formulas to start with 2π-periodic functions and then to generalise later to the case when the period can be any $T = 2L > 0$.

5.2 The Fourier series for a 2π-periodic function $f(x)$

We will need some revision and preliminary results about cosines and sines before we can justify the formula that appear but first it helps to indicate what is a Fourier series for a "suitable" 2π-periodic function f. The term "suitable" for this module will be piecewise smooth which here means piecewise continuous and in an interval where it is continuous it has a derivative except possibly at a finite number of points. For example, the functions f_1 and f_2 defined on $(-\pi, \pi]$ given by

$$f_1(x) = \begin{cases} 1, & \text{if } 0 \le x \le \pi, \\ 0, & \text{if } -\pi < x < 0, \end{cases} \quad \text{and} \quad f_2(x) = |x| = \begin{cases} x, & \text{if } 0 \le x \le \pi, \\ -x, & \text{if } -\pi < x < 0, \end{cases} \tag{5.2.1}$$

satisfy the requirements and are shown in figures 5.1 and 5.2 on page 5-3. In the context of Fourier series we extend these 2π periodically so that they are defined for all $x \in \mathbb{R}$ and a sketch of these in $(-3\pi, 3\pi]$ are shown in figures 5.3 and 5.4 on page 5-4. Note that $f_1(x)$ has jump discontinuities at points $k\pi$ when k is an integer and $f_2(x)$ is continuous on $\mathbb{R}$ with the first derivative having jump discontinuities at $k\pi$ when k is an integer.

The Fourier series of a 2π-periodic function $f(x)$ is written as

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nx) + b_n \sin(nx)\right), \tag{5.2.2}$$

$$\text{where} \quad a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\,\mathrm{d}x \quad \text{and} \quad b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\,\mathrm{d}x. \tag{5.2.3}$$

The meaning of the symbol $\sim$ here is that the series is determined by f and it represents f in some way but it is not necessarily the case that the value

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nx) + b_n \sin(nx)\right) = \lim_{N\to\infty} \left(\frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \cos(nx) + b_n \sin(nx)\right) \right)$$

is the same as $f(x)$ for all $x \in (-\pi, \pi]$. As we will indicate later, it is the same for all x in this interval in the case of $f_2(x)$ and in the case of the discontinuous function $f_1(x)$ it is the same at all points of continuity of $f_1(x)$ but it is not the same at the points $k\pi$, $k \in \mathbb{Z}$ where we have the jump discontinuity.

5.3 The orthogonality properties of $\cos(nx)$ and $\sin(nx)$ on $(-\pi, \pi)$

In the case of column vectors $\underline{x} = (x_i) \ne \underline{0}$ and $\underline{y} = (y_i) \ne \underline{0}$ in $\mathbb{R}^n$ we say that they are orthogonal if

$$\underline{x}^T \underline{y} = x_1 y_1 + \cdots + x_n y_n = 0.$$

The operation $\underline{x}^T \underline{y}$ is also called the dot product, scalar product or inner product of the two vectors. We also have inner products for other types of objects and in the case of functions $f(x)$ and $g(x)$ defined on $(-\pi, \pi)$ we say that they are orthogonal if

$$\int_{-\pi}^{\pi} f(x) g(x)\,\mathrm{d}x = 0.$$

We next show that we have this property when we consider the functions

$$1,\ \cos(x),\ \sin(x),\ \cos(2x),\ \sin(2x), \dots, \cos(nx),\ \sin(nx), \dots. \tag{5.3.1}$$

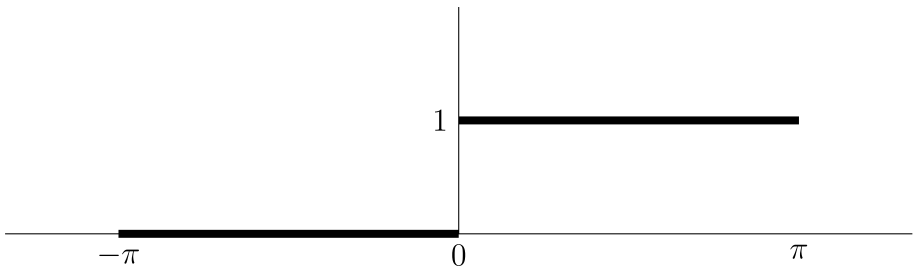

Figure 5.1: Sketch of $f_1(x)$ on $-\pi < x \le \pi$.

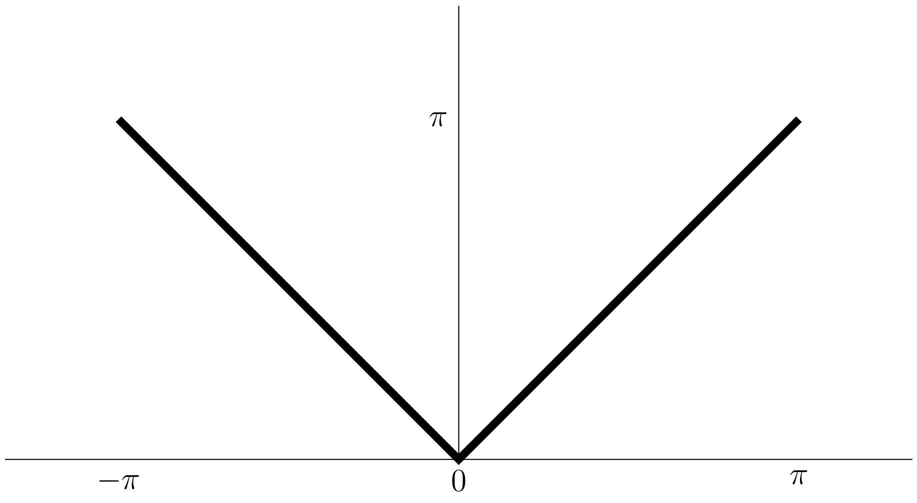

Figure 5.2: Sketch of $f_2(x) = |x|$ on $-\pi < x \le \pi$.

To help evaluate the integrals that we encounter we need to recall some of the trig. identities involving the addition formulas which in complex form can we written as

$$\mathrm{e}^{i(a+b)} = \mathrm{e}^{ia}\mathrm{e}^{ib}.$$

In terms of cosines and sines we have

$$\begin{aligned}\cos(a+b) + i\sin(a+b) &= (\cos(a) + i\sin(a))(\cos(b) + i\sin(b)) \\ &= (\cos(a)\cos(b) - \sin(a)\sin(b)) + i(\sin(a)\cos(b) + \cos(a)\sin(b)).\end{aligned}$$

If we replace b by $-b$ then we have

$$\cos(a-b) + i\sin(a-b) = (\cos(a)\cos(b) + \sin(a)\sin(b)) + i(\sin(a)\cos(b) - \cos(a)\sin(b)). \tag{5.3.2}$$

If we add (5.3.2) and (5.3.2) and divide by 2 then we get

$$\cos(a)\cos(b) = \frac{\cos(a+b) + \cos(a-b)}{2}, \tag{5.3.3}$$

$$\sin(a)\cos(b) = \frac{\sin(a+b) + \sin(a-b)}{2}. \tag{5.3.4}$$

Similarly if we subtract (5.3.2) from (5.3.2) and divide by 2 then we get

$$\sin(a)\sin(b) = \frac{\cos(a-b) - \cos(a+b)}{2}, \tag{5.3.5}$$

$$\cos(a)\sin(b) = \frac{\sin(a+b) - \sin(a-b)}{2}. \tag{5.3.6}$$

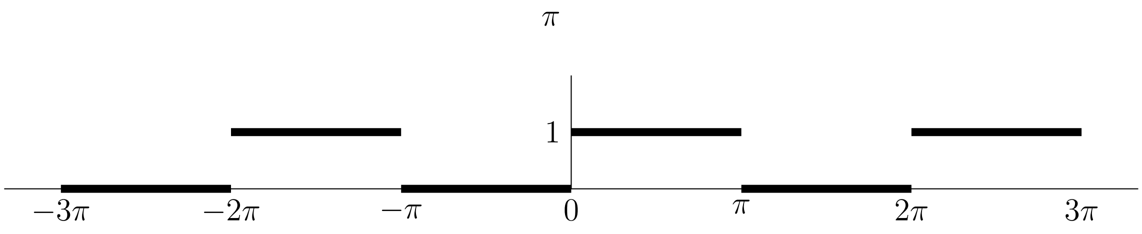

Figure 5.3: Sketch of the 2π periodic function $f_1(x)$ on $-3\pi < x \le 3\pi$.

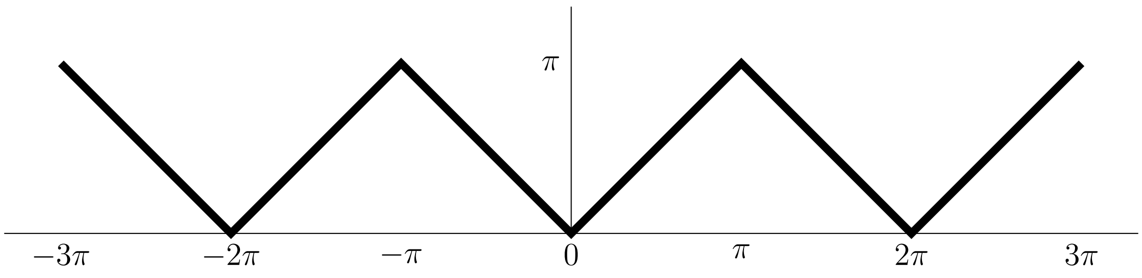

Figure 5.4: Sketch of the 2π periodic function $f_2(x)$ on $-3\pi < x \le 3\pi$

Now for the evaluation of the integrals we have that when p is an integer and $p \ne 0$ we immediately obtain

$$\int_{-\pi}^{\pi} \cos(px)\,\mathrm{d}x = 0, \quad \int_{-\pi}^{\pi} \sin(px)\,\mathrm{d}x = 0$$

and when $p = 0$ we trivially have $\cos(px) = 1$ and

$$\int_{-\pi}^{\pi} \mathrm{d}x = 2\pi.$$

We first consider taking $f(x)$ and $g(x)$ to be two different functions from the set of functions listed in (5.3.1). If $f(x) = 1$ and $g(x) = \cos(nx)$ (with $n > 0$) or $g(x) = \sin(nx)$ then we have

$$\int_{-\pi}^{\pi} \cos(nx)\,\mathrm{d}x = \int_{-\pi}^{\pi} \sin(nx)\,\mathrm{d}x = 0.$$

If $n > 0$, $m > 0$ and $n \ne m$ then $n+m$ and $n-m$ are both non-zero integers and we have

$$\begin{aligned} \int_{-\pi}^{\pi} \cos(nx)\cos(mx)\,\mathrm{d}x &= \frac{1}{2}\int_{-\pi}^{\pi} \cos((n+m)x) + \cos((n-m)x)\,\mathrm{d}x = 0, \\ \int_{-\pi}^{\pi} \sin(nx)\sin(mx)\,\mathrm{d}x &= \frac{1}{2}\int_{-\pi}^{\pi} \cos((n-m)x) - \cos((n+m)x)\,\mathrm{d}x = 0. \end{aligned}$$

If n and m are now any integers then as $n+m$ and $n-m$ are both integers we have

$$\begin{aligned} \int_{-\pi}^{\pi} \sin(nx)\cos(mx)\,\mathrm{d}x &= \frac{1}{2}\int_{-\pi}^{\pi} \sin((n+m)x) + \sin((n-m)x)\,\mathrm{d}x = 0, \\ \int_{-\pi}^{\pi} \cos(nx)\sin(mx)\,\mathrm{d}x &= \frac{1}{2}\int_{-\pi}^{\pi} \sin((n+m)x) - \sin((n-m)x)\,\mathrm{d}x = 0. \end{aligned}$$

The inner product of one function with any other function in the list (5.3.1) is hence zero. If we take $f(x) = g(x)$ then when $f(x) = g(x) = 1$ we have

$$\int_{-\pi}^{\pi} \mathrm{d}x = 2\pi \tag{5.3.7}$$

and if we take $f(x) = g(x) = \cos(nx)$ or we take $f(x) = g(x) = \sin(nx)$ then we have

$$\int_{-\pi}^{\pi} \cos^2(nx)\,\mathrm{d}x = \frac{1}{2}\int_{-\pi}^{\pi} 1 + \cos(2nx)\,\mathrm{d}x = \pi, \tag{5.3.8}$$

$$\int_{-\pi}^{\pi} \sin^2(nx)\,\mathrm{d}x = \frac{1}{2}\int_{-\pi}^{\pi} 1 - \cos(2nx)\,\mathrm{d}x = \pi. \tag{5.3.9}$$

5.4 The formula for the Fourier coefficients a_n and b_n

We consider next justifying why a_n and b_n are as given in (5.2.3) and we do this by supposing that we have suitable numbers $a_0, a_1, \ldots, a_n, \ldots$ and $b_0, b_1, \ldots, b_n, \ldots$ so that the series converges to define a function $f(x)$ given by

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx)) \tag{5.4.1}$$

and we suppose that we can integrate this function. Firstly,

$$\begin{aligned}
\int_{-\pi}^{\pi} f(x)\,\mathrm{d}x &= \int_{-\pi}^{\pi} \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx))\,\mathrm{d}x \\
&= \frac{a_0}{2}\int_{-\pi}^{\pi} \mathrm{d}x + \sum_{n=1}^{\infty}\int_{-\pi}^{\pi} (a_n \cos(nx) + b_n \sin(nx))\,\mathrm{d}x \\
&= a_0\pi
\end{aligned}$$

assuming that it is valid to interchange the summation and the integral. If we instead multiply by $\cos(mx)$, $m \ge 1$, and integrate then we get

$$\begin{aligned}
\int_{-\pi}^{\pi} f(x)\cos(mx)\,\mathrm{d}x &= \int_{-\pi}^{\pi} \left(\frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx))\right)\cos(mx)\,\mathrm{d}x \\
&= \frac{a_0}{2}\int_{-\pi}^{\pi} \cos(mx)\,\mathrm{d}x \\
&\quad + \sum_{n=1}^{\infty}\int_{-\pi}^{\pi} (a_n \cos(nx)\cos(mx) + b_n \sin(nx)\cos(mx))\,\mathrm{d}x \\
&= a_m\pi
\end{aligned}$$

by using the orthogonality properties given earlier. Similarly, if we instead multiply by $\sin(mx)$ and integrate then we get

$$\begin{aligned}
\int_{-\pi}^{\pi} f(x)\sin(mx)\,\mathrm{d}x &= \int_{-\pi}^{\pi} \left(\frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(nx) + b_n \sin(nx))\right)\sin(mx)\,\mathrm{d}x \\
&= \frac{a_0}{2}\int_{-\pi}^{\pi} \sin(mx)\,\mathrm{d}x \\
&\quad + \sum_{n=1}^{\infty}\int_{-\pi}^{\pi} (a_n \cos(nx)\sin(mx) + b_n \sin(nx)\sin(mx))\,\mathrm{d}x \\
&= b_m\pi
\end{aligned}$$

where again we use the orthogonality properties given earlier. Hence we have shown that when we have (5.4.1) the coefficients are

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,\mathrm{d}x \quad \text{and} \quad b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,\mathrm{d}x, \quad n = 0, 1, 2, \ldots$$

We can include b_0 as trivially $b_0 = 0$ as the integrand is 0 and note that we have $a_0/2$ in the formula so that we have the same expression for all the a_n terms.

It is worth making some additional comments about the formula for the coefficients. Firstly, when a function is 2π-periodic we can use any interval of length 2π in the formula, i.e. we have

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,\mathrm{d}x = \frac{1}{\pi}\int_{0}^{2\pi} f(x)\cos(nx)\,\mathrm{d}x = \frac{1}{\pi}\int_{\alpha}^{\alpha+2\pi} f(x)\cos(nx)\,\mathrm{d}x$$

and

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,\mathrm{d}x = \frac{1}{\pi}\int_{0}^{2\pi} f(x)\sin(nx)\,\mathrm{d}x = \frac{1}{\pi}\int_{\alpha}^{\alpha+2\pi} f(x)\sin(nx)\,\mathrm{d}x$$

for all $\alpha \in \mathbb{R}$. In these notes we will use $(-\pi, \pi]$ and when odd and even functions are considered as it is convenient to have 0 in the middle of the interval. Secondly, if we write $a_n(f)$ and $b_n(f)$ to indicate more explicitly the dependence of the coefficients on the function then these relations are linear, i.e. if we have functions $f(x)$ and $g(x)$ then the Fourier coefficients of the function $\alpha f(x) + \beta g(x)$ are given by

$$\begin{aligned} a_n(\alpha f + \beta g) &= \alpha a_n(f) + \beta a_n(g), \\ b_n(\alpha f + \beta g) &= \alpha b_n(f) + \beta b_n(g). \end{aligned}$$

This is used in some examples.

5.5 Examples of the coefficients a_n and b_n for various functions

In the following we determine the Fourier series for several different functions and illustrate graphically some of the partial sums

$$f_N(x) = \frac{a_0}{2} + \sum_{n=1}^{N} (a_n \cos(nx) + b_n \sin(nx))$$

to see how they compare with the function $f(x)$ from which they are derived.

1. The Heaviside function is defined by $H : \mathbb{R} \to \mathbb{R}$,

$$H(x) = \begin{cases} 1, & \text{if } x \ge 0, \\ 0, & \text{if } x < 0. \end{cases}$$

The function $f_1(x) = H(x)$ corresponds with this in $(-\pi, \pi]$ which we then continue in a 2π-periodic way. The Fourier coefficient a_0 of f_1 is

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f_1(x)\,\mathrm{d}x = \frac{1}{\pi} \int_0^{\pi} \mathrm{d}x = 1.$$

For $n \ge 1$,

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f_1(x)\cos(nx)\,\mathrm{d}x = \frac{1}{\pi} \int_0^{\pi} \cos(nx)\,\mathrm{d}x = \frac{1}{\pi} \left[\frac{\sin(nx)}{n} \right]_0^{\pi} = 0,$$

$$\begin{aligned} b_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f_1(x)\sin(nx)\,\mathrm{d}x = \frac{1}{\pi} \int_0^{\pi} \sin(nx)\,\mathrm{d}x \\ &= \frac{1}{\pi} \left[\frac{-\cos(nx)}{n} \right]_0^{\pi} = \frac{-\cos(n\pi) + 1}{n\pi} \\ &= \begin{cases} \frac{2}{n\pi}, & \text{if } n \text{ is odd}, \\ 0, & \text{if } n \text{ is even}. \end{cases} \end{aligned}$$

The Fourier series is hence

$$\frac{1}{2} + \frac{2}{\pi} \left(\sin(x) + \frac{\sin(3x)}{3} + \cdots + \frac{\sin((2n-1)x}{2n-1} + \cdots \right).$$

There are some mathematical questions to ask concerning whether or not this series converges and if it does converge then does it converge to $f_1(x)$? Although we do not provide any rigorous proofs, the series does converge for all values of x where $f_1(x)$ is continuous but if we let $x = 0$ of $x = \pi$, which is where $f_1(x)$ is not continuous, then all the sine terms are zero and all the partial sums are $1/2$ and thus in particular it converges to $1/2$ which is not the value of $f_1(0) = f_1(\pi) = 1$. However it does converge to $f_1(x)$ for all other x in the interval, i.e. it converges in $(-\pi, 0)$ and in

$(0, \pi)$. We can illustrate this with the following Matlab statements which computes and displays the partial sums defined by

$$g_{2m-1}(x) = \frac{1}{2} + \frac{2}{\pi} \sum_{k=1}^{m} \frac{\sin(2k-1)x}{2k-1}$$

for $m = 2, 8, 32, 128$. The plots created are shown in figure 5.5 on page 5-9.

```
x=linspace(-pi, pi, 501);
mm=[2, 8, 32, 128];
g=zeros( size(x) );
for k=1:max(mm)
  g=g+sin( (2*k-1)*x )/(2*k-1);

  % create a plot if k is one of the values of mm
  if sum(k==mm)==1
    figure(k)
    y=0.5+(2/pi)*g;
    plot(x, y, 'LineWidth', 2);
    s=sprintf('Fourier series for f1 with terms up to sin(%dx)', 2*k-1);
    title(s, 'FontSize', 14);
  end
end
```

2. We now consider the function $f_2(x) = |x|$ which is even in $(-\pi, \pi)$. As $f_2(x)$ is even and $\sin(nx)$ is odd it follows that

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f_2(x) \sin(nx)\,\mathrm{d}x = 0, \quad n = 1, 2, \ldots$$

For a_0 we have

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f_2(x)\,\mathrm{d}x = \frac{2}{\pi} \int_0^{\pi} x\,\mathrm{d}x = \frac{2}{\pi}\frac{\pi^2}{2} = \pi.$$

For $n \ge 1$ we use integration by parts to evaluate the integrals and we have

$$\begin{aligned} a_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f_2(x) \cos(nx)\,\mathrm{d}x = \frac{2}{\pi} \int_0^{\pi} x \cos(nx)\,\mathrm{d}x \\ &= \frac{2}{\pi} \left(\left[\frac{x \sin(nx)}{n} \right]_0^{\pi} - \int_0^{\pi} \frac{\sin(nx)}{n}\,\mathrm{d}x \right). \end{aligned}$$

For the term in the square brackets the expression is 0 at both limits. Thus

$$\begin{aligned} a_n &= \frac{2}{n\pi} \int_0^{\pi} (-\sin(nx))\,\mathrm{d}x = \frac{2}{n\pi} \left[\frac{\cos(nx)}{n} \right]_0^{\pi} \\ &= \frac{2}{n^2\pi} \left((-1)^n - 1\right) = \begin{cases} 0, & \text{if } n \text{ is even}, \\ \dfrac{-4}{n^2\pi}, & \text{if } n \text{ is odd}. \end{cases} \end{aligned}$$

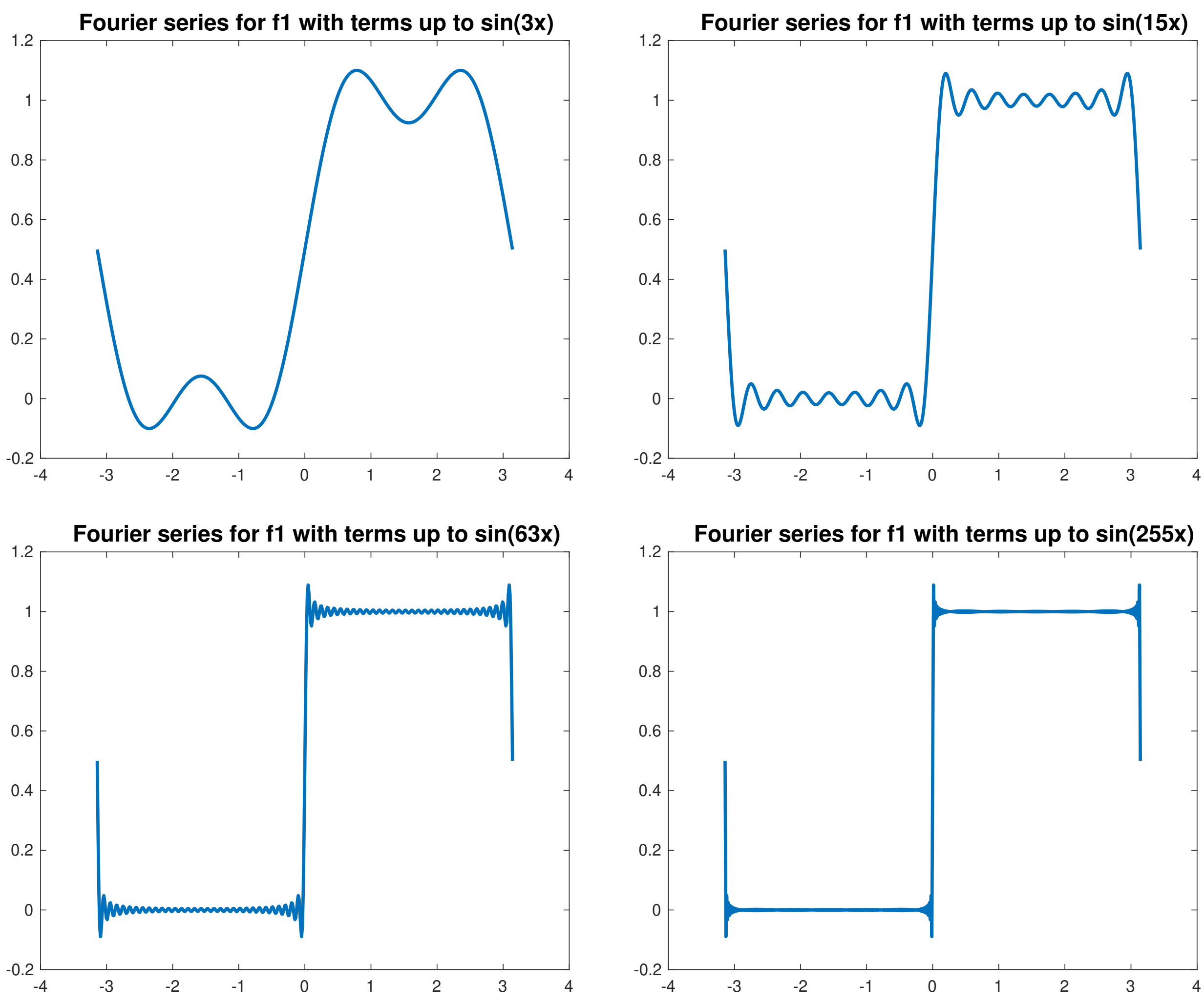

Figure 5.5: Plots of the partial sums of the Fourier series for $f_1(x)$.

In this case the Fourier series does converge to $f_2(x)$ for all $x \in [-\pi, \pi]$ and we can write

$$f_2(x) = |x| = \frac{\pi}{2} - \frac{4}{\pi}\left(\cos(x) + \frac{\cos(3x)}{3^2} + \cdots + \frac{\cos((2n-1)x)}{(2n-1)^2} + \cdots\right), \quad |x| \le \pi.$$

We can illustrate this with the following Matlab statements which computes and displays the partial sums defined by

$$g_{2m-1}(x) = \frac{\pi}{2} - \frac{4}{\pi}\left(\cos(x) + \frac{\cos(3x)}{3^2} + \cdots + \frac{\cos((2m-1)x)}{(2m-1)^2}\right).$$

for $m = 4$, 16. The plots created are shown in figure 5.6 on page 5-10.

```
x=linspace(-pi, pi, 501);
mm=[4, 16];
g=zeros( size(x) );
for k=1:max(mm)
  g=g+cos( (2*k-1)*x )/((2*k-1)^2);

  % create plot if k is one of the values of mm
  if sum(k==mm)==1
    figure(k)
    y=0.5*pi-(4/pi)*g;
    plot(x, y, 'LineWidth', 2);
    axis equal
    s=sprintf('Fourier series for f2 with terms up to cos(%dx)', 2*k-1);
    title(s, 'FontSize', 14);
  end
end
```

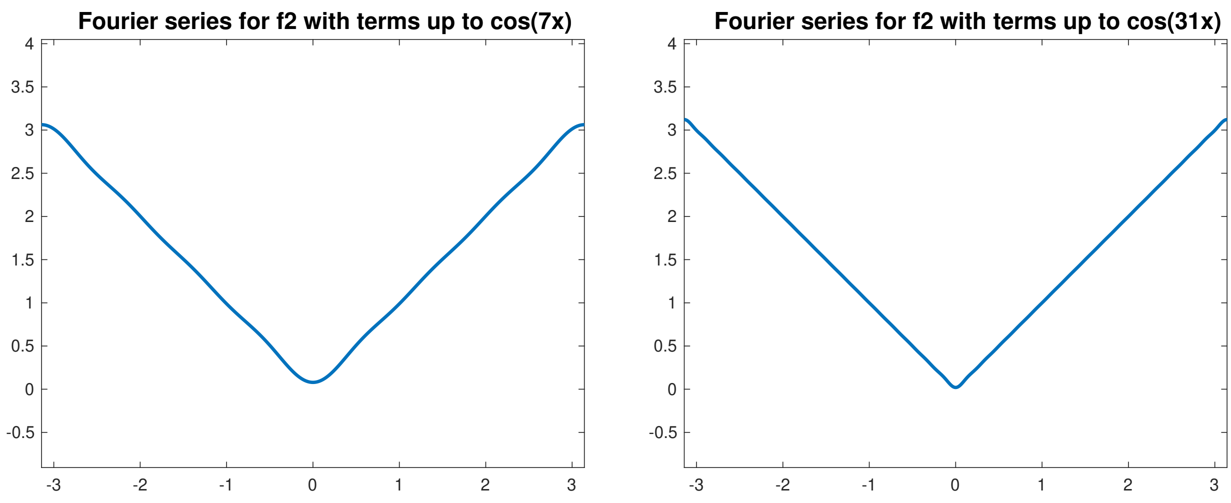

Figure 5.6: Plots of the partial sums of the Fourier series for $f_2(x)$.

As a final point, as the Fourier series converges to $f_2(x)$ for all $x \in [-\pi, \pi]$ we can write

$$|x| = \frac{\pi}{2} - \frac{4}{\pi}\left(\cos(x) + \frac{\cos(3x)}{3^2} + \cdots + \frac{\cos((2n-1)x)}{(2n-1)^2} + \cdots\right), \quad |x| \le \pi.$$

In particular if we let $x = 0$ then

$$0 = \frac{\pi}{2} - \frac{4}{\pi}\left(1 + \frac{1}{3^2} + \frac{1}{5^2} + \cdots\right),$$

which rearranges to

$$\sum_{n=1}^{\infty} \frac{1}{(2n-1)^2} = \frac{\pi^2}{8}.$$

3. We now consider the Fourier series of the function

$$f_3(x) = x, \quad -\pi < x \le \pi,$$

which we continue 2π-periodically. This function is odd in $(-\pi, \pi)$ and a consequence

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} x\cos(nx)\,\mathrm{d}x = 0, \quad n = 0, 1, 2, \ldots.$$

For the b_n coefficients the integrands are even which means that we only need to consider $[0, \pi]$ and to make further progress we need to use integration by parts.

$$\begin{aligned} b_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} x\sin(nx)\,\mathrm{d}x = \frac{2}{\pi}\int_0^{\pi} x\sin(nx)\,\mathrm{d}x \\ &= \frac{2}{\pi}\left(\left[\frac{-x\cos(nx)}{n}\right]_0^{\pi} - \int_0^{\pi}\frac{-\cos(nx)}{n}\,\mathrm{d}x\right). \end{aligned}$$

For the term in the square brackets we have

$$\left[\frac{-x\cos(nx)}{n}\right]_0^{\pi} = \frac{-\pi\cos(n\pi)}{n} = \frac{(-1)^{n+1}\pi}{n}.$$

The part with the integral is 0 as

$$\int_0^{\pi}\cos(nx)\,\mathrm{d}x = \left[\frac{\sin(nx)}{n}\right]_0^{\pi} = 0$$

and thus

$$b_n = \frac{2}{\pi}\frac{(-1)^{n+1}\pi}{n} = \frac{2(-1)^{n+1}}{n}.$$

The Fourier series does converges to $f_3(x)$ in $-\pi < x < \pi$ where $f_3(x)$ is continuous and we can write

$$x = 2\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{n}\sin(nx) = 2\left(\sin(x) - \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} - \cdots\right).$$

In this example if we let $x = \pi$ in the Fourier series then all the sine terms are 0 and the value of the Fourier series is 0 which is not the same as $f_3(\pi) = \pi$ or of the limit $\lim_{x\downarrow -\pi} f_3(x) = -\pi$. At the point of discontinuity we get the average of the two values just given as was similarly the case when $f_1(x)$ was considered. This is actually generally the case as we state in the next section.

As in the previous two examples, below is a short Matlab script to create and plot partial sums for the series with the plots being shown in Figure 5.7 on page 5-13.

```
x=linspace(-pi, pi, 501);
mm=[4, 16, 64, 256];
g=zeros( size(x) );
pm=1;
for k=1:max(mm)
  g=g+pm*sin(k*x)/k;
  pm=-pm;

  % create a plot if k is one of the values of mm
  if sum(k==mm)==1
    figure(k)
    y=2*g;
    plot(x, y, 'LineWidth', 2);
    axis equal
    s=sprintf('Fourier series for f3 with terms up to sin(%dx)', k);
    title(s, 'FontSize', 14);
  end
end
```

4. As a final example, if we let

$$f_4(x) = \frac{f_2(x)+f_3(x)}{2} = \frac{|x|+x}{2} = \begin{cases} 0, & -\pi < x < 0, \\ x, & 0 \le x \le \pi, \end{cases}$$

then we can quickly obtain the Fourier series by combining the series for $f_2(x)$ and $f_3(x)$ giving

$$\begin{aligned} f_4(x) &= \frac{\pi}{4} - \frac{2}{\pi}\left(\cos(x) + \frac{\cos(3x)}{3^2} + \cdots + \frac{\cos((2n-1)x)}{(2n-1)^2} + \cdots\right) \\ &\quad + \left(\sin(x) - \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} - \cdots\right) \end{aligned}$$

for $-\pi < x < \pi$. The series does not converge to $f_4(x)$ at the points of discontinuity when we consider this as 2π periodic function, and in the interval $(-\pi, \pi]$ this is the point $x = \pi$.

5.6 The pointwise convergence of Fourier series

In the examples of section 5.5 we claimed that in the case of the function $f_2(x)$, which is continuous, the Fourier series converges to it for all $x \in \mathbb{R}$. In the other examples we claimed that the series converges at the points of continuity to the functions that they represented and although each series also converges at the points of discontinuity we could observe that the convergence was to the average of the values. It would take a few

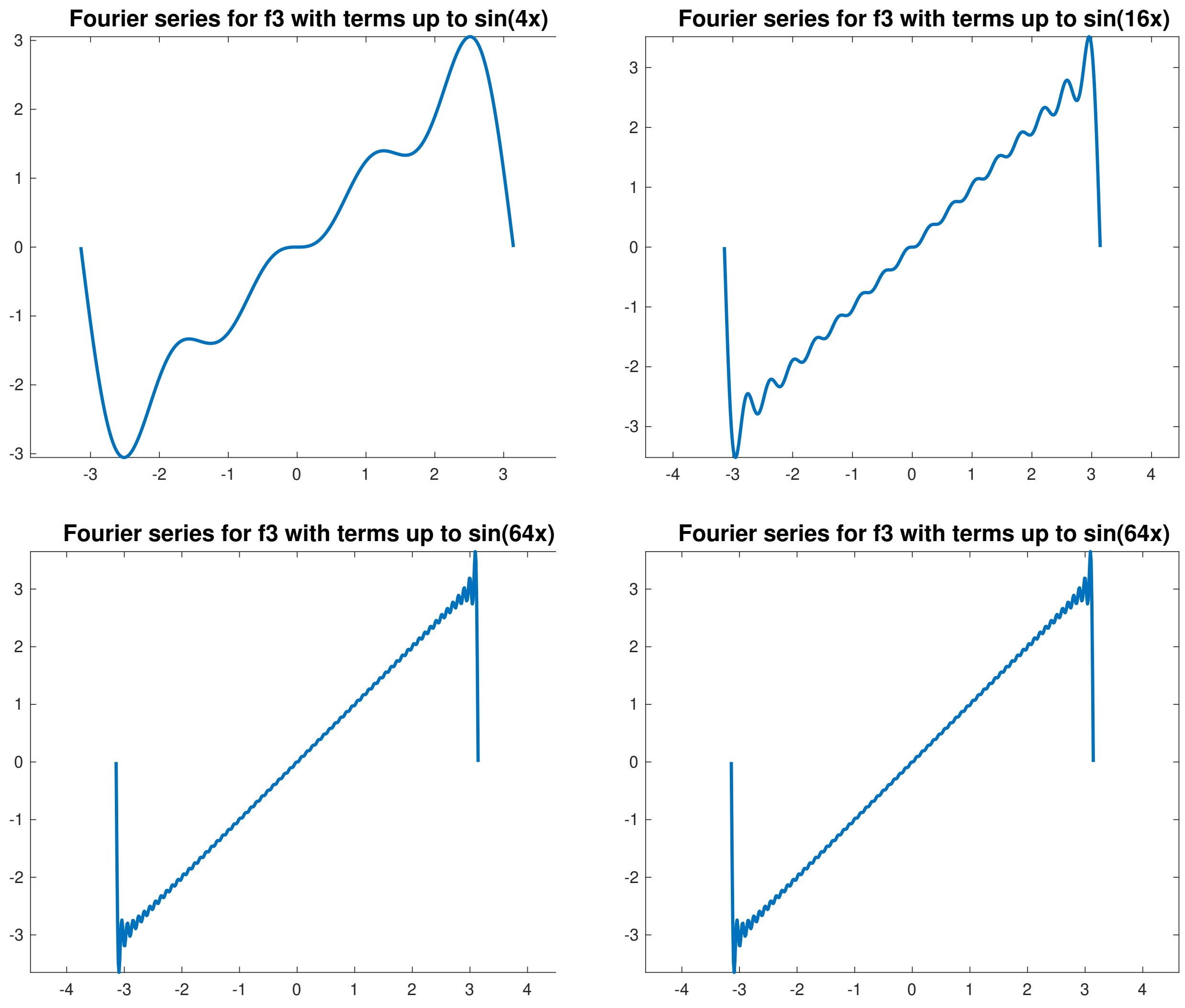

Figure 5.7: Plots of the partial sums of the Fourier series for $f_3(x) = x$.

lectures to prove why this is the case and such proofs will not be done in `MA2715` and we just restrict to stating sufficient conditions for this results.

One commonly used set of sufficient conditions for the pointwise convergence of Fourier series are known as the Dirichlet conditions although these are expressed in slightly different ways in different texts. We have for example the following.

> On page 584 of the book by James et al given at the start of the `MA2715` notes the conditions on $f(x)$ are as follows. The function $f(x)$ is a bounded periodic function such that in any period it has a finite number of isolated maxima and minima and a finite number of points of discontinuity.

> On page 22 of Fourier Analysis by Murray R. Spiegel the function $f(x)$ should be such that $f(x)$ and $f'(x)$ are piecewise continuous. In particular this means that every point $x \in (-\pi, \pi)$ the following left and right limits must exist.

$$f(x-) = \lim_{t \uparrow x} f(t), \quad f(x+) = \lim_{t \downarrow x} f(t),$$
$$f'(x-) = \lim_{t \uparrow x} f'(t), \quad f'(x+) = \lim_{t \downarrow x} f'(t).$$

It is worth mentioning that these are sufficient conditions on $f(x)$ for the following but they are not necessary conditions, i.e. the result may still hold in some cases when they are not satisfied. When the sufficient conditions are satisfied we can say the following.

1. The Fourier series converges to $f(x)$ if x is a point of continuity.
2. The Fourier series converges to
$$\frac{f(x+)+f(x-)}{2}=\frac{\lim_{t\downarrow x} f(t)+\lim_{t\uparrow x} f(t)}{2}$$
if x is a point of discontinuity.

Thus if the 2π-periodic function is continuous on $\mathbb{R}$ then convergence is at all points and we had this case with the even function $f_2(x)$ in the examples. In the case of the function $f_1(x)$ we had a point of discontinuity in the interval $(-\pi,\pi]$ as well as at the end of the interval but in the case of $f_3(x)$ and $f_4(x)$ the point of discontinuity was just at the end of the interval $(-\pi,\pi]$, i.e. the value $f(-\pi):=f(\pi)=\pi$ is not the same as the value $\lim_{x\downarrow -\pi} f(x)=-\pi$.

5.7 Comments about the uniform convergence and the L_2 convergence of Fourier series

In the previous subsection we just stated sufficient conditions for a Fourier series to converge in a pointwise sense and we gave the limit value. The limit value was only the same as $f(x)$ at a point of continuity of $f(x)$. No proofs were given as this would go beyond what can be done in just a few lectures as well as being a bit more difficult and more technical than the other material presented in this chapter. The purpose of this subsection is just to mention that the convergence of a series of functions can be considered in other senses than pointwise convergence.

5.7.1 Uniform convergence

If you are on the mathematics degree then uniform convergence is one of your topics. In the context of Fourier series we have uniform convergence when we have the following. Let
$$s_N(x)=\frac{a_0}{2}+\sum_{n=1}^{N}\left(a_n\cos(nx)+b_n\sin(nx)\right),\quad N=1,2,\dots \tag{5.7.1}$$
with the Fourier coefficients given as before. We say that s_N tends to f uniformly on $[-\pi,\pi]$ if
$$\max\left\{|f(x)-s_N(x)|:\ -\pi\le x\le\pi\right\}\to 0 \quad \text{as } N\to\infty.$$
This was only the case in our examples when the 2π-periodic function $f(x)$ is continuous. Thus, as our examples have shown, we can have pointwise convergence but not uniform convergence.

5.7.2 L_2 convergence

With the conditions on the type of functions being considered these are more than sufficient for us to have what is known as L_2 convergence. With s_N being as defined in (5.7.1) this means that

$$\int_{-\pi}^{\pi} (f(x) - s_N(x))^2 \, \mathrm{d}x \to 0 \quad \text{as } N \to \infty. \tag{5.7.2}$$

There are some relations connected with (5.7.2) which quite quickly follow. Firstly, from the orthogonality properties of functions in the set

$$V_N = \{1, \cos(x), \sin(x), \cos(2x), \sin(2x), \ldots, \cos(Nx), \sin(Nx)\}$$

we have

$$\begin{aligned}\int_{-\pi}^{\pi} s_N(x)^2 \, \mathrm{d}x &= \frac{a_0^2}{4} \int_{-\pi}^{\pi} \mathrm{d}x + \sum_{n=1}^{N} \int_{-\pi}^{\pi} a_n^2 \cos^2(nx) + b_n^2 \sin^2(nx) \, \mathrm{d}x \\ &= \pi \left(\frac{a_0^2}{2} + \sum_{n=1}^{N} (a_n^2 + b_n^2) \right). \end{aligned} \tag{5.7.3}$$

Now if we consider again how we justified the expressions for the Fourier coefficients we have for any function $\pi \in V$ that

$$\int_{-\pi}^{\pi} (f(x) - s_N(x)) \phi(x) \, \mathrm{d}x = 0.$$

s_N is a linear combination of functions in V_N and thus

$$\int_{-\pi}^{\pi} (f(x) - s_N(x)) s_N(x) \, \mathrm{d}x = 0 \quad \text{which gives} \quad \int_{-\pi}^{\pi} f(x) s_N(x) \, \mathrm{d}x = \int_{-\pi}^{\pi} s_N^2(x) \, \mathrm{d}x.$$

Using both these results gives

$$\int_{-\pi}^{\pi} (f(x) - s_N(x))^2 \, \mathrm{d}x = \int_{-\pi}^{\pi} (f(x) - s_N(x)) f(x) \, \mathrm{d}x = \int_{-\pi}^{\pi} (f(x)^2 - s_N(x)^2) \, \mathrm{d}x.$$

That is

$$\int_{-\pi}^{\pi} f(x)^2 \, \mathrm{d}x = \int_{-\pi}^{\pi} s_N(x)^2 \, \mathrm{d}x + \int_{-\pi}^{\pi} (f(x) - s_N(x))^2 \, \mathrm{d}x.$$

Thus when we have (5.7.2) the use of (5.7.3) gives, as $N \to \infty$,

$$\frac{1}{\pi} \int_{-\pi}^{\pi} f(x)^2 \, \mathrm{d}x = \frac{a_0^2}{2} + \sum_{n=1}^{\infty} (a_n^2 + b_n^2). \tag{5.7.4}$$

Equation (5.7.4) is known as Parseval's identity.

5.8 Half range series

In the material on Fourier series given so far we have considered representing a function $f(x)$ specified on $(-\pi, \pi]$ by a Fourier series with the series representing the 2π-periodic extension of $f(x)$. If we keep with the period being 2π then we can instead start with functions just given on $(0, \pi)$ which we extend to $(-\pi, \pi)$ as an odd function or we can extend to $(-\pi, \pi)$ as an even function and consider the series that we get in each case. The Fourier series for the odd extension will just involve sine terms and the Fourier series for the even extension will just involve cosine terms. The series created in this way are known as the half range Fourier sine series or the half range Fourier cosine series. As the odd extension generates an odd function on $(-\pi, \pi)$ it follows that the integrand $f(x)\sin(nx)$ in the formula for the coefficient is an even function. Similarly, as the even extension generates an even function on $(-\pi, \pi)$ the integrand $f(x)\cos(nx)$ is an even function. In both cases the formula for the coefficients in the half range series just involve integrals on $(0, \pi)$ and we have the following.

The half range cosine series for $f(x)$ defined on $(0, \pi)$ is

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(nx), \quad a_n = \frac{2}{\pi}\int_0^{\pi} f(x)\cos(nx)\,\mathrm{d}x. \tag{5.8.1}$$

The half range sine series for $f(x)$ defined on $(0, \pi)$ is

$$f(x) \sim \sum_{n=1}^{\infty} b_n \sin(nx), \quad b_n = \frac{2}{\pi}\int_0^{\pi} f(x)\sin(nx)\,\mathrm{d}x. \tag{5.8.2}$$

The examples given already fit, or nearly fit, this category already.

In the case of $f_1(x)$ involving the Heaviside function if we instead take

$$\tilde{f}_1(x) = 2f_1(x) - 1 = \begin{cases} 1, & \text{if } 0 \le x \le \pi, \\ -1, & \text{if } -\pi < x < 0, \end{cases} \tag{5.8.3}$$

then we have an odd function (at most points) and the series only involves sine terms, i.e.

$$\tilde{f}_1(x) \sim \frac{4}{\pi}\sum_{n=1}^{\infty} \frac{\sin((2n-1)x}{2n-1}. \tag{5.8.4}$$

We can use the equal sign here if $x \in (0, \pi)$.

In the case of $f_2(x) = |x|$ the series is already the half range cosine series and in the case of $f_3(x) = x$ the series is already the half range sine series. The functions $f_2(x)$ and $f_3(x)$ are the same in $(0, \pi)$ and as a consequence we have two representations of x in $[0, \pi)$, i.e. for $0 \le x < \pi$,

$$\begin{aligned} x &= \frac{\pi}{2} - \frac{4}{\pi}\left(\cos(x) + \frac{\cos(3x)}{3^2} + \cdots + \frac{\cos((2n-1)x}{(2n-1)^2} + \cdots\right) \\ &= 2\left(\sin(x) - \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} + \cdots + (-1)^{n+1}\frac{\sin(nx)}{n} + \cdots\right). \end{aligned}$$

5.9 Differentiation and integration of Fourier series

We have considered Fourier series for functions whose 2π-periodic extensions are discontinuous at certain points and for the example which was continuous everywhere the derivative had discontinuities at certain points. We now consider when it is valid to integrate or differentiate a Fourier series of a function to obtain the Fourier series of a related function and although rigorous proofs are not given we attempt to partially justify the results. When the term-by-term operations are valid this will give one method of generating one Fourier series from another.

In summary, as all the functions that we consider are piecewise continuous it is valid to integrate term-by-term and the new function is "smoother" although care is needed related to the $a_0/2$ term when this is non-zero. It is only valid to differentiate term-by-term when the 2π periodic extension is continuous.

5.9.1 Integrating a Fourier series

With the summary already given stating that we can integrate term-by-term we consider here examples.

The function $f_2(x)$ considered earlier is such that $f_2(x) = |x|$ when $-\pi < x < \pi$ and the derivative is the piecewise continuous function

$$\tilde{f}_2'(x) = \begin{cases} 1, & \text{if } x > 0, \\ -1, & \text{if } x < 0. \end{cases}$$

The point $x = 0$ is a point of discontinuity of $\tilde{f}_2'$. This function coincides with (5.8.3) given earlier (at least at all points except $x = 0$) and we already have the Fourier series (5.8.4) which we repeat again here. For $x \in (-\pi, 0) \cup (0, \pi)$

$$\tilde{f}_1(x) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\sin((2n-1)x}{2n-1}.$$

If we integrate 1 then we get x and if we integrate -1 then we get $-x$ and thus the function $f_2(x)$ can be written as

$$\begin{aligned} f_2(x) &= \int_0^x \tilde{f}_1(t)\,\mathrm{d}t = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{1}{2n-1} \int_0^x \sin((2n-1)t)\,\mathrm{d}t \\ &= \frac{4}{\pi} \sum_{n=1}^{\infty} \left[\frac{-\cos((2n-1)t)}{(2n-1)^2} \right]_0^x \\ &= \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{1}{(2n-1)^2} \left(1 - \cos((2n-1)x\right) && (5.9.1) \\ &= \frac{a_0}{2} - \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\cos((2n-1)x)}{(2n-1)^2} && (5.9.2) \end{aligned}$$

if we define

$$\frac{a_0}{2} = \frac{4}{\pi} \sum_{1}^{\infty} \frac{1}{(2n-1)^2}.$$

Hence we get the series for $f_2(x)$ from the series for $\tilde{f}_1(x)$ by integrating term-by-term although the constant term appears differently. To complete the details you still need to consider

$$\frac{1}{\pi}\int_{-\pi}^{\pi} f_2(x)\,\mathrm{d}x = \frac{2}{\pi}\int_0^{\pi} x\,\mathrm{d}x = \pi$$

to establish that $a_0 = \pi$ and

$$\frac{\pi}{2} = \frac{4}{\pi}\sum_1^\infty \frac{1}{(2n-1)^2}.$$

If we consider the integration of a Fourier series more generally then at points of continuity we have

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^\infty (a_n\cos(nt) + b_n\sin(nt))$$

and we can shift the constant term to the left hand side to give

$$f(t) - \frac{a_0}{2} = \sum_{n=1}^\infty (a_n\cos(nt) + b_n\sin(nt))$$

and integrating from 0 to x gives

$$\int_0^x f(t)\,\mathrm{d}t - \frac{a_0 x}{2} = \sum_{n=1}^\infty \left(\frac{a_n}{n}\sin(nx) - \frac{b_n}{n}(\cos(nx)-1)\right).$$

As an example, consider again the case $f(x) = |x|$ in $(-\pi,\pi)$ which is even and the Fourier series only involves the cosine terms. Recall that we had

$$|t| = \frac{\pi}{2} - \frac{4}{\pi}\left(\cos(t) + \frac{\cos(3t)}{3^2} + \cdots + \frac{\cos((2n-1)t}{(2n-1)^2} + \cdots\right)$$

so that

$$|t| - \frac{\pi}{2} = -\frac{4}{\pi}\left(\cos(t) + \frac{\cos(3t)}{3^2} + \cdots + \frac{\cos((2n-1)t}{(2n-1)^2} + \cdots\right).$$

Now $|t| = t$ for $t > 0$ and $|t| = -t$ for $t < 0$ and if we define $g(x)$ by

$$g(x) = \int_0^x |t|\,\mathrm{d}t = \begin{cases} x^2/2, & x \ge 0, \\ -x^2/2, & x \le 0 \end{cases}$$

then term-by-term integration gives

$$\phi(x) = g(x) - \frac{\pi x}{2} = -\frac{4}{\pi}\sum_{n=1}^\infty \frac{\sin((2n-1)x}{(2n-1)^3}.$$

The piecewise defined function $\phi(x)$ is continuously differentiable and to directly check this and determine the values note that

$$\phi(x) = \begin{cases} \frac{1}{2}(x^2 - \pi x) = \frac{1}{2}x(x-\pi), & \text{when } 0 \le x \le \pi, \\ \frac{1}{2}(-x^2 - \pi x) = -\frac{1}{2}x(x+\pi), & \text{when } -\pi \le x \le 0. \end{cases}$$

The function is 0 at $-\pi$, 0 and π where the pieces join and the left and right derivatives agree at the join points and we have

$$\phi'(0) = -\frac{\pi}{2} \quad \text{and } \phi'(-\pi) = \phi'(\pi) = \frac{\pi}{2}.$$

On $[-\pi,\pi]$ the function $\phi(x)$ is piecewise defined with the two quadratic pieces giving a continuously differentiable function when we consider the 2π periodic extension.

5.9.2 Differentiating a Fourier series

We have already stated that for the piecewise smooth functions that we have considered it is only valid to differentiate the series term-by-term if the 2π periodic extension of the function is continuous. As we have taken $(-\pi, \pi]$ as our interval this means that for such a function $f(x)$ we need to have

$$f(\pi) = \lim_{x \uparrow \pi} f(x) = \lim_{x \downarrow -\pi} f(x).$$

As our function $f(x)$ is continuous we have at all points in $[-\pi, \pi]$

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nx) + b_n \sin(nx) \right).$$

Term-by-term differentiation gives

$$\sum_{n=1}^{\infty} \left(n b_n \cos(nx) - n a_n \sin(nx) \right).$$

Now if we consider the function $f'(x)$ which is piecewise defined and we are able to integrate it then the Fourier coefficients are as follows.

$$\tilde{a}_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f'(x) \cos(nx) \,\mathrm{d}x \quad \text{and} \quad \tilde{b}_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f'(x) \sin(nx) \,\mathrm{d}x.$$

We can re-express both of these by integration by parts.

$$\begin{aligned} \tilde{a}_n &= \frac{1}{\pi} \left[f(x) \cos(nx) \right]_{-\pi}^{\pi} - \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)(-n \sin(nx)) \,\mathrm{d}x = n b_n, \\ \tilde{b}_n &= \frac{1}{\pi} \left[f(x) \sin(nx) \right]_{-\pi}^{\pi} - \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)(n \cos(nx)) \,\mathrm{d}x = -n a_n, \end{aligned}$$

where in the first case we need to use that $f(-\pi) = f(\pi)$. Thus the Fourier coefficients that we obtain by term-by-term differentiation are the same as what we get when we use the formula with the function as $f'(x)$ and we have

$$f'(x) \sim \sum_{n=1}^{\infty} \left(n b_n \cos(nx) - n a_n \sin(nx) \right).$$

The Fourier series for $f'(x)$ is the same as $f'(x)$ at points of continuity of $f'(x)$ and it is the average of the left and right limits at points of discontinuity of $f'(x)$.

5.10 The Fourier series for a function $f(x)$ of period $2L$

For the final part of these notes on Fourier series suppose that the least period of $f(x)$ is $2L > 0$, i.e.

$$f(x + 2L) = f(x), \quad \text{for all } x \in \mathbb{R}.$$

As we have already described the 2π periodic case we can adjust for this by letting

$$t = \frac{\pi x}{L} \quad \text{or equivalently} \quad x = \frac{Lt}{\pi}$$

so that $x = \pm L$ correspond respectively to $t = \pm\pi$ and first consider the related 2π periodic function

$$g(t) = f\left(\frac{Lt}{\pi}\right).$$

Equivalently we can write

$$f(x) = g\left(\frac{\pi x}{L}\right).$$

From what has already been done the Fourier series for $g(t)$ is

$$g(t) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nt) + b_n \sin(nt)\right).$$

Hence the corresponding Fourier series for $f(x)$ is

$$f(x) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right).$$

This is the form of a Fourier series when the period is $2L$. To express the Fourier coefficients in terms of integrals involving $f(x)$ can be done using the substitution indicated above, i.e.

$$t = \frac{\pi x}{L}, \quad \frac{\mathrm{d}t}{\mathrm{d}x} = \frac{\pi}{L}, \quad g(t) = f(x).$$

Thus

$$g(t)\cos(nt)\frac{\mathrm{d}t}{\mathrm{d}x} = f(x)\cos\left(\frac{n\pi x}{L}\right)\frac{\pi}{L}, \quad g(t)\sin(nt)\frac{\mathrm{d}t}{\mathrm{d}x} = f(x)\sin\left(\frac{n\pi x}{L}\right)\frac{\pi}{L},$$

and the Fourier coefficients can be written as

$$\begin{aligned} a_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} g(t)\cos(nt)\,\mathrm{d}t = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)\,\mathrm{d}t, \\ b_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} g(t)\sin(nt)\,\mathrm{d}t = \frac{1}{L}\int_{-L}^{L} f(x)\sin\left(\frac{n\pi x}{L}\right)\,\mathrm{d}t. \end{aligned}$$

Example

Let $f(x) = |\sin(x)|$ which periodic and continuous. As $\sin(x+\pi) = -\sin(x)$ it follows that $f(x+\pi) = f(x)$ and the least period is $2L = \pi$ so that $L = \pi/2$. The function is even and thus only cosine terms are involved in the Fourier series which is of the form

$$f(x) = |\sin(x)| = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(2nx).$$

As the function is even we get the coefficients by just integrating over half the range, i.e. $[0, \pi/2]$.

$$a_0 = \frac{4}{\pi}\int_0^{\pi/2} \sin(x)\,\mathrm{d}x = \frac{4}{\pi}\left[-\cos(x)\right]_0^{\pi/2} = \frac{4}{\pi}.$$

For $n \ge 1$,

$$a_n = \frac{4}{\pi} \int_0^{\pi/2} \sin(x)\cos(2nx)\,\mathrm{d}x.$$

Now recall the trig. identity

$$2\sin(x)\cos(2nx) = \sin((2n+1)x) - \sin((2n-1)x).$$

Thus

$$\begin{aligned} \int_0^{\pi/2} 2\sin(x)\cos(2nx)\,\mathrm{d}x &= \int_0^{\pi/2} \sin((2n+1)x) - \sin((2n-1)x)\,\mathrm{d}x \\ &= \left[\frac{-\cos((2n+1)x)}{2n+1} + \frac{\cos((2n-1)x)}{2n-1}\right]_0^{\pi/2} \\ &= \frac{1}{2n+1} - \frac{1}{2n-1} = \frac{-2}{4n^2-1}. \end{aligned}$$

Hence

$$a_n = -\frac{4}{\pi}\left(\frac{1}{4n^2-1}\right).$$

The Fourier series is

$$|\sin(x)| = \frac{2}{\pi} - \frac{4}{\pi}\sum_{n=1}^{\infty} \frac{\cos(2nx)}{4n^2-1}.$$