A Review of Selected Topics from Numerical Analysis

Lloyd D. Fosdick
Elizabeth R. Jessup

September 5, 1995

High Performance Scientific Computing
University of Colorado at Boulder

The following are members of
the HPSC Group of the Department of Computer Science
at the University of Colorado at Boulder:

Lloyd D. Fosdick
Elizabeth R. Jessup
Carolyn J. C. Schauble
Gitta O. Domik

Contents

A Review of Selected Topics from Numerical Analysis *

Lloyd D. Fosdick
Elizabeth R. Jessup

September 5, 1995

The course "High-Performance Scientific Computing" assumes an introductory knowledge of numerical analysis such as you might gain from any of a number of textbooks; this includes the ones by Conte and deBoor [Conte & de Boor 80], Dahlquist and Björk [Dahlquist & Björck 74], or Kahaner, Moler and Nash [Kahaner et al 89]. These are referred to as CdB, DB, and KMN in the sequel.

The purpose of this review is simply to record definitions, formulas, and concepts you are expected to know; and to define the notation we will use. It makes no serious attempt to explain or teach this material — for that you should refer to one of the texts just cited, or some similar text. The topics we cover are:

- Notation
- Error
- Taylor's series

*This work has been supported by the National Science Foundation under an Educational Infrastructure grant, CDA-9017953. It has been produced by the HPSC Group, Department of Computer Science, University of Colorado, Boulder, CO 80309. Please direct comments or queries to Elizabeth Jessup at this address or e-mail jessup@cs.colorado.edu.

- Elementary linear algebra
- Elementary numerical solution of ordinary differential equations
- Elementary Fourier series

1 Notation

$f'(x)$: This is $\frac{df}{dx}$, and similarly for the higher derivatives $f''(x)$, $f'''(x)$, etc..

$f'(x_i)$: $\frac{df}{dx}$ evaluated at the point x_i, and similarly for the higher derivatives $f''(x_i)$, $f'''(x_i)$, etc.

$\mathcal{O}(n^k)$: $f(n)$ is $\mathcal{O}(n^k)$ if

$$lim_{n\to\infty}(\frac{f(n)}{n^k}) = C,$$

where C is a nonzero constant. Alternatively, this notation is used to represent limiting behavior as a parameter goes to zero; e.g., $\mathcal{O}(h^k)$ means

$$lim_{h\to 0}(\frac{f(h)}{h^k}) = C,$$

***flt(expression)*:** the computed value of *expression*: usually different from the value of *expression* because of roundoff error.

$[a, b]$: the closed interval from a to b (i.e., includes endpoints).

(a, b): the open interval from a to b (i.e., does not include endpoints).

$[a, b)$, $(a, b]$: semiclosed intervals.

$\mathcal{T}_n(f(x_0 + \delta x))$: Taylor's polynomial of degree n for approximating $f(x)$ at $x_0 + \delta x$.

$\mathcal{R}^n$: Vector space of dimension n, real-valued elements; we use $\mathcal{C}^n$ if the elements are complex.

$\mathcal{R}^{r\times c}$: The space of matrices with r rows and c columns, real elements; we use $\mathcal{C}^{r\times c}$ if the elements are complex.

$x^{(i)}$: The i^{th} vector of a set of vectors.

$e^{(i)}$: The i^{th} unit vector (1 in row i, 0 elsewhere).

$A^{(i)}$: The i^{th} matrix of a set of matrices.

x_i: The i^{th} element of a vector.

$a_{i,j}$ **or** $(A)_{i,j}$: The element of a matrix A located in row i and column j.

$a_{i,:}$: The i^{th} row of a matrix A.

$a_{:,j}$: The j^{th} column of a matrix A.

$A_{i:i',j:j'}$: The rectangular submatrix of a matrix A consisting of row i to row i' and column j to column j', inclusive.

$(A)_{i,j}$: The (i,j) minor of a matrix A .

$det(A)$: Determinant of A.

x^T, A^T: Transpose of vector x, matrix A.

$\| x \|$: Norm of x; specific norms are $\| x \|_1$, $\| x \|_2$, and $\| x \|_\infty$.

$\hat{x}$: An eigenvector.

2 Error

Errors are caused by rounding of numbers and by formulas and procedures that give only approximate results because they are not carried to a limit (e.g., a truncated Taylor series, Simpson's formula for evaluating an integral, and Euler's method for solving a differential equation). The first kind of error we call *roundoff error*, the second *truncation error*. The total error in a computation is normally the result of these two kinds of error.

Formally, we define the *error* as the difference between the exact value of a quantity and its approximate value. Thus, the error in an approximation to x is:

$$e(x) = x_{exact} - x_{approx}.$$

This error is also referred to as the *absolute error*. The *relative error* is the ratio

$$r(x) = \frac{e(x)}{x_{exact}}.$$

From the two equations above we note that

$$x_{exact} = \frac{x_{approx}}{(1 - r(x))}.$$

In general we don't know x_{exact}, but we often can estimate an upper bound for the magnitude of the relative error, say $|r|_{ub}$, and so we can bound x_{exact} using values we know. If x_{exact} and x_{approx} are both positive then

$$\frac{x_{approx}}{(1 + |r|_{ub})} \leq x_{exact} \leq \frac{x_{approx}}{(1 - |r|_{ub})};$$

if they are both negative then

$$\frac{x_{approx}}{(1 - |r|_{ub})} \leq x_{exact} \leq \frac{x_{approx}}{(1 + |r|_{ub})}.$$

Of course, we should have $|r|_{ub} << 1$.

When two numbers nearly equal in magnitude but with opposite sign are added together, cancellation of leading digits produces a large relative error in the result. For example, suppose we have two numbers:

$$\begin{aligned} x &= +10000.2, \\ y &= -10000.0\ . \end{aligned}$$

Assume that the error in each is less than 0.05 in magnitude, then the magnitude of the relative error in each number is less than 5×10^{-6}. The sum of x and y is 0.2, with an error of at most 0.1 in magnitude, and a relative error as large as 0.5. Thus, we can have a relative error in the sum that is 10^5 larger than the relative error in the operands.

Cancellation of leading digits can lead to a result that might surprise you. A good example is the evaluation of a function at, or very near, one of its roots. For example, suppose that α is a root of the polynomial

$$x^5 + x^3 + x - 10000.19,$$

then by definition

$$\alpha^5 + \alpha^3 + \alpha - 10000.19 = 0.$$

However, if you evaluate the polynomial at α on a computer roundoff error is likely to cause a nonzero result. For example, if the computations were done to an accuracy of about seven decimal digits (corresponding to the accuracy of the type REAL in Fortran on most computers) then you could get a result as large as 0.01 and erroneously conclude that there is a mistake in the value of α.

To see why the result 0.01 for the computed value of the polynomial at α is completely reasonable, consider how the arithmetic might be done. Suppose that the polynomial is evaluated from left to right, then the last operation is

$$T - 10000.19,$$

where T represents the computed value of the expression

$$\alpha^5 + \alpha^3 + \alpha.$$

Now T should be close to 10000.19, but it could easily be off by a unit in the seventh place because of roundoff error. In that case we would get 0.01 for $T - 10000.19$. Thus, an expression which we think should be zero, or close to zero, evaluates to 0.01. The point to recognize here is that *0.01 is close to zero relative to the operands we used.* In fact, we could hardly have gotten much closer to zero: a "very small" result, much smaller than 0.01, would be impossible.

See KMN, chapter 2, for a good discussion of error.

3 Floating-point numbers

The form of a binary floating-point number, x, is

$$x = s \times 2^e,$$

where s is a real number, called the *significand*, and e is an integer, called the *exponent*. In a computer the value of x is usually held as an ordered triple consisting of the sign of the significand, the magnitude of the significand, and the *biased* exponent ($e_b = e + bias$). Usually the significand is normalized so that its magnitude is in $[1, 2)$, and the $bias$, a positive constant, on the exponent is chosen so that $e_b \geq 0$. The value *zero* is treated as a special case, usually with $|s| = e_b = 0$.

Many computers use the IEEE standard for floating-point arithmetic which is defined in the document *An American National Standard & IEEE Standard for Binary Floating-Point Arithmetic*. It is informally described in the tutorial *Elements of IEEE Arithmetic* [Fosdick 95][1]. In this standard the significand for single precision has 24 bits and is normalized to lie in $\pm[1, 2)$; therefore, a change of one unit in the least significant place of the single precision significand, the *unit spacing*, is given by:

$$(\text{unit spacing})_{sp} = 2^{-23} \approx 1.2 \times 10^{-7}.$$

The single precision exponent is in $[-126, 127]$, and so the *range* for single precision is given by:

$$\begin{aligned}(\text{range})_{sp} &= \pm[2^{-126}, (2 - 2^{-23})2^{127}], \\ &\approx \pm[1.2 \times 10^{-38}, 3.4 \times 10^{38}].\end{aligned}$$

The values for double precision are:

$$\begin{aligned}(\text{unit spacing})_{dp} &= 2^{-52} \approx 2.2 \times 10^{-16}, \\ (\text{range})_{dp} &= \pm[2^{-1022}, (2 - 2^{-52})2^{1023}], \\ &\approx \pm[2.2 \times 10^{-308}, 1.8 \times 10^{308}].\end{aligned}$$

Normal rounding in IEEE floating-point is *round-to-nearest*, with *round-to-even* in case of a tie. This means that with normal rounding

$$\begin{aligned}|r|_{ub} &= 2^{-24} \approx 6.0 \times 10^{-8} \quad (\text{single precision}), \\ |r|_{ub} &= 2^{-53} \approx 1.1 \times 10^{-16} \quad (\text{double precision}).\end{aligned}$$

[1]This is available via anonymous ftp at the `cs.colorado.edu` site in the `/pub/HPSC` directory.

4 Taylor's series

Taylor's series provides a tool for approximating a function in terms of its value and the value of its derivatives at a single point. The series approximation is

$$f(x_0+\delta x) \approx f(x_0) + \frac{f'(x_0)}{1!}\delta x + \frac{f''(x_0)}{2!}(\delta x)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(\delta x)^n, \quad (1)$$

where

$$\begin{aligned} f(x) &= \text{the function to be approximated,} \\ x_0 &= \text{the point where values of } f,\ f',\ f'', \ldots \text{ are known,} \\ x_0+\delta x &= \text{the point where } f \text{ is approximated.} \end{aligned}$$

The expression on the right of equation (1) is a polynomial in δx which is called *Taylor's polynomial* and denoted by $\mathcal{T}_n(f(x_0+\delta x))$, thus

$$\mathcal{T}_n(f(x_0+\delta x)) = f(x_0) + \frac{f'(x_0)}{1!}\delta x + \frac{f''(x_0)}{2!}(\delta x)^2 + \ldots + \frac{f^{(n)}(x_0)}{n!}(\delta x)^n.$$

The truncation error in Taylor's approximation is given by the the *remainder*, denoted by R_{n+1}:

$$R_{n+1} = f(x_0+\delta x) - \mathcal{T}_n(f(x_0+\delta x)).$$

There are two formulas for the remainder, one in terms of a derivative of $f(x)$, the other in terms of an integral of $f(x)$:

$$\begin{aligned} R_{n+1} &= \frac{f^{(n+1)}(x_0+\xi)}{(n+1)!}(\delta x)^{n+1}, \quad \xi \in (0,\delta x), \\ &= \frac{1}{n!}\int_0^{\delta x} (\delta x - \zeta)^n f^{(n+1)}(x_0+\zeta)\, d\zeta. \end{aligned}$$

In practice we usually cannot evaluate the remainder, but we can use it to estimate a bound on the error. For example, if we know a number F such that

$$|f^{(n+1)}(x_0+\xi)| < F \quad \text{if} \quad \xi \in (0,\delta x),$$

then we know from the above formulas that

$$|R_{n+1}| < \frac{F}{(n+1)!}|\delta x|^{n+1}.$$

We can illustrate these ideas with a simple example. Let

$$\begin{aligned} f(x) &= \sin(x), \\ f(x_0+\delta x) &\approx \mathcal{T}_4(\sin(x_0+\delta x)). \end{aligned}$$

Then

$$\begin{aligned} \sin(x_0+\delta x) &\approx \sin(x_0) + \frac{\cos(x_0)}{1!}\delta x - \frac{\sin(x_0)}{2!}(\delta x)^2 - \\ &\quad \frac{\cos(x_0)}{3!}(\delta x)^3 + \frac{\sin(x_0)}{4!}(\delta x)^4. \end{aligned}$$

If $x_0 = 0$, this equation becomes

$$\sin(\delta x) \approx \delta x - \frac{1}{6}(\delta x)^3.$$

The error in this approximation is R_5, which in the derivative form can be expressed as

$$R_5 = \frac{\cos(\xi)}{120}(\delta x)^5.$$

Since the magnitude of the cosine is at most 1, we have

$$|R_5| \leq \frac{1}{120}(\delta x)^5.$$

5 Linear algebra

Linear algebra deals with quantities we call *scalars, vectors,* and *matrices.* Scalars are simply numbers. Vectors and matrices are reviewed below. A good reference for this section is KMN.

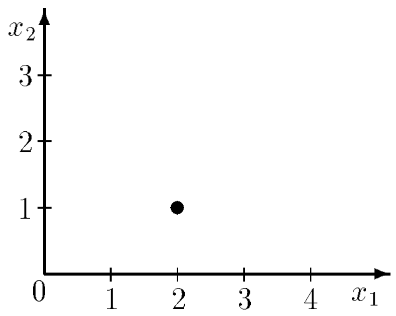

Figure 1: Geometric representation of $x^T = (2\ 1)$.

5.1 Vectors

A vector is an ordered list of scalars often written as

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}. \tag{2}$$

This is *column* form. It is also written in *row* form:

$$x^T = \begin{pmatrix} x_1 & x_2 & \ldots & x_n \end{pmatrix}.$$

We put the superscript T on vectors in row form unless the context makes it obvious that they are row vectors: the T stands for *transpose*. The numbers x_i are called the *elements* of x.

To identify the fact that x has n elements we write the expression $x \in \mathcal{R}^n$ if the elements of x are real numbers, the $\mathcal{R}$ standing for "real." Similarly, we write $x \in \mathcal{C}^n$ if the elements are complex numbers.

Geometrically, x represents a point in an n-dimensional space: the elements giving the coordinate values. Thus the vector

$$x = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

represents the point in figure 1.

The *length* of a vector is simply the number of elements it has: x in equation (2) has length n. There is a chance for confusion here because the word "length" is also used in the geometrical sense of distance. The context should make the meaning clear. Often the phrase *Euclidean length* is used when speaking in geometric terms. The Euclidean length is defined by

$$\text{Euclidean length of } x = (\sum_{i=1}^{n} x_i^2)^{1/2}. \tag{3}$$

5.1.1 Vector arithmetic

There follows a list of common arithmetic operations involving vectors and scalars. In this list a is a scalar and x and y are vectors of length n.

$$\begin{aligned}
&\text{(scalar times vector)} \\
ax &= \begin{pmatrix} ax_1 \\ ax_2 \\ \vdots \\ ax_n \end{pmatrix} \\
&\text{(vector plus vector)} \\
x + y &= \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix} \\
&\text{(scalar product, or dot product)} \\
x^T y &= \sum_{i=1}^{n} x_i y_i \\
&\text{(vector product, or cross product)} \\
xy^T &= \begin{pmatrix} x_1y_1 & x_1y_2 & \dots & x_1y_n \\ x_2y_1 & x_2y_2 & \dots & x_2y_n \\ \vdots & \vdots & & \vdots \\ x_ny_1 & x_ny_2 & \dots & x_ny_n \end{pmatrix}
\end{aligned}$$

If $x^{(1)}, x^{(2)}, \dots, x^{(k)}$ are vectors and $a_1, a_2, \dots, a_k$ are scalars then the expression

$$a_1x^{(1)} + a_2x^{(2)} + \ldots + a_kx^{(k)}$$

is referred to as a *linear combination* of the vectors $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$.

5.1.2 Special vectors: unit and zero

The unit vector, $e^{(j)}$, is defined as follows:

$$e_i^{(j)} = \begin{cases} 1 & (i = j), \\ 0 & (i \neq j). \end{cases}$$

We can express x in terms of unit vectors as follows:

$$x = \sum_{i=1}^{n} x_i e^{(i)}.$$

The unit vectors are also known as the *canonical vectors*.

The vector with all elements equal to zero is often written simply as "0"; e.g., in the vector equation

$$x + (-x) = 0.$$

As in this case, the meaning of 0 is usually clear from the context. Sometimes this vector is called the *null* vector and denoted ϕ.

5.1.3 Orthogonal vectors

A pair of vectors x and y is said to be *orthogonal* if

$$x^T y = 0.$$

Note that the unit vectors $e^{(i)}$ are orthogonal: we say they form an *orthogonal set of vectors*.

5.1.4 Vector norms

Informally the *norm* of a vector is a scalar that represents the size or magnitude of the vector. Formally, we say that the norm of a vector x is a scalar, represented by $\| x \|$, with the following properties:

1. $\| x \| > 0$ if $x \neq 0$;
2. $\| x \| = 0$ if and only if $x = 0$;
3. If a is any scalar, then $\| ax \| = |a| \, \| x \|$;
4. If y is also a vector, then $\| x + y \| \leq \| x \| + \| y \|$.

The last property is sometimes called the *triangle inequality.*

There are three commonly used norms: $\| x \|_1$, called the *1-norm*; $\| x \|_2$, called the *Euclidean norm* or *2-norm*; and $\| x \|_\infty$, called the *infinity norm.* They are defined as follows:

$$\| x \|_1 = \sum_{i=1}^{n} |x_i|, \tag{4}$$

$$\| x \|_2 = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}}, \tag{5}$$

$$\| x \|_\infty = \max_{1 \leq i \leq n} |x_i|. \tag{6}$$

It is easy to verify that each of these norms satisfies the four conditions enumerated above. Notice that $\| x \|_2$ is the Euclidean length, equation (3).

5.1.5 Linear independence

The notion of linear independence is important because it allows us to identify sets of vectors in terms of which other vectors can be expressed. It also provides a convenient way to characterize matrices and the existence of solutions to a system of equations.

Consider the following four vectors:

$$y^{(1)} = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \quad y^{(2)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad y^{(3)} = \begin{pmatrix} 2 \\ 7 \\ 0 \end{pmatrix}, \quad y^{(4)} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

It is obvious that

$$y^{(3)} = 2y^{(1)} + 3y^{(2)},$$

and by rearranging the terms we can equally well express $y^{(2)}$ as a linear combination of $y^{(1)}$ and $y^{(3)}$

$$y^{(2)} = -\frac{2}{3}y^{(1)} + \frac{1}{3}y^{(3)}$$

and similarly for $y^{(3)}$. The fact that any one of these three vectors can be expressed as a linear combination of the other two is expressed by saying that set of vectors $\{y^{(1)}, y^{(2)}, y^{(3)}\}$ is *linearly dependent*. Notice, however that we cannot express $y^{(4)}$ as a linear combination of $y^{(1)}$ and $y^{(2)}$: this is obvious because $y^{(1)}$ and $y^{(2)}$ have their third element equal to zero so there is no way to form a linear combination of them to produce $y^{(4)}$. We express this fact by saying that the set of vectors $\{y^{(1)}, y^{(2)}, y^{(4)}\}$ is *linearly independent.*

Formally, we say that the vectors $x^{(1)}, x^{(2)}, \ldots, x^{(k)}$ are linearly independent if the equation

$$a_1x^{(1)} + a_2x^{(2)} + \ldots + a_kx^{(k)} = 0$$

can only be satisfied when *all* of the coefficients a_i are zero: a set of vectors that is not linearly independent is said to be linearly dependent. The set $\{y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}\}$ must be linearly dependent because the four vectors are all of length three.

In the space $\mathcal{R}^n$, the unit vectors are linearly independent.

Any vector in $\mathcal{R}^n$ can be expressed as a linear combination of the vectors in *any* linearly independent set of vectors in $\mathcal{R}^n$. We express the notion that a linearly independent set of vectors can be used to represent any vector in $\mathcal{R}^n$ by saying that *the set spans the space.*

5.2 Matrices

Matrices are rectangular arrays of numbers such as

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & -1 & 0 \\ 4 & 4 & 2 \end{pmatrix}, \quad \begin{pmatrix} 1.2 & -3.4 \\ 2.5 & 1.9 \\ -0.78 & 2.9 \\ -1.0 & 0.99 \end{pmatrix}.$$

The general form for the matrix A is given in the following expression:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,c} \\ a_{2,1} & a_{2,2} & \dots & a_{2,c} \\ \vdots & \vdots & & \vdots \\ a_{r,1} & a_{r,2} & \dots & a_{r,c} \end{pmatrix}.$$

This matrix has $r * c$ elements, arranged in r rows and c columns: we say it is an $r \times c$ matrix — the mathematical statement for this is $A \in \mathcal{R}^{r \times c}$ if the elements of A are real and $A \in \mathcal{C}^{r \times c}$ if the elements of A are complex. If $r = c$, we say that A is *square*.

The line extending from the upper left corner to the lower right corner of a square matrix is the *main diagonal*; thus the elements on the main diagonal are $a_{1,1}, a_{2,2}, \dots, a_{n,n}$.

A matrix is said to be *order* n if it is an $n \times n$ square matrix.

A matrix is said to be *rank* m if m of its columns are linearly independent. The number of linearly independent columns of a matrix equals its number of linearly independent rows.

A vector is a special case of a matrix. The vector $x \in \mathcal{R}^n$ can also be regarded as a matrix in the space $\mathcal{R}^{n \times 1}$, and the vector x^T can be regarded as a matrix in the space $\mathcal{R}^{1 \times n}$.

It is useful to be able to refer to columns or rows of a matrix. For this we use $a_{:,j}$ to denote the j^{th} column of A; and $a_{i,:}$ to denote the i^{th} row of A. Similarly, we use $A_{i:i',j:j'}$ to denote the *submatrix* consisting of the rows i to i' and columns j to j', inclusive, of A.

In referring to an element of a matrix A, we use the notation $a_{i,j}$ as above: the same name but in lower case. Occasionally, we use $(A)_{i,j}$ to denote the same thing.

5.2.1 Matrix arithmetic

There follows a list of arithmetic operations involving matrices. In this list A and B are order n, and C is an $n \times m$ matrix; x is a column vector of n elements; p is a scalar. Notice that an expression like $A_{i,:}B_{:,j}$ is the dot product of the i^{th} row of A and the j^{th} column of B.

(scalar times array)

$$pA = \begin{pmatrix} pa_{1,1} & pa_{1,2} & \dots & pa_{1,n} \\ pa_{2,1} & pa_{2,2} & \dots & pa_{2,n} \\ \vdots & \vdots & & \vdots \\ pa_{n,1} & pa_{n,2} & \dots & pa_{n,n} \end{pmatrix}$$

(matrix plus matrix)

$$A + B = \begin{pmatrix} a_{1,1}+b_{1,1} & a_{1,2}+b_{1,2} & \dots & a_{1,n}+b_{1,n} \\ a_{2,1}+b_{2,1} & a_{2,2}+b_{2,2} & \dots & a_{2,n}+b_{2,n} \\ \vdots & \vdots & & \vdots \\ a_{n,1}+b_{n,1} & a_{n,2}+b_{n,2} & \dots & a_{n,n}+b_{n,n} \end{pmatrix}$$

(array times vector)

$$Ax = \begin{pmatrix} a_{1,:}^T x \\ a_{2,:}^T x \\ \vdots \\ a_{n,:}^T x \end{pmatrix}$$

(vector times array)

$$x^T A = \begin{pmatrix} x^T a_{:,1} & x^T a_{:,2} & \dots & x^T a_{:,n} \end{pmatrix}$$

(matrix times matrix : square)

$$AB = \begin{pmatrix} a_{1,:}^T b_{:,1} & a_{1,:}^T b_{:,2} & \dots & a_{1,:}^T b_{:,n} \\ a_{2,:}^T b_{:,1} & a_{2,:}^T b_{:,2} & \dots & a_{2,:}^T b_{:,n} \\ \vdots & \vdots & & \vdots \\ a_{n,:}^T b_{:,1} & a_{n,:}^T b_{:,2} & \dots & a_{n,:}^T b_{:,n} \end{pmatrix}$$

(matrix times matrix : rectangular)

$$AC = \begin{pmatrix} a_{1,:}^T c_{:,1} & a_{1,:}^T c_{:,2} & \dots & a_{1,:}^T c_{:,m} \\ a_{2,:}^T c_{:,1} & a_{2,:}^T c_{:,2} & \dots & a_{2,:}^T c_{:,m} \\ \vdots & \vdots & & \vdots \\ a_{n,:}^T c_{:,1} & a_{n,:}^T c_{:,2} & \dots & a_{n,:}^T c_{:,m} \end{pmatrix}$$

The matrix-vector and matrix-matrix products can be rewritten in a variety of other ways. For example, they may be written to show operations in terms of the columns of the matrix A instead of the rows of the matrix A.

Such rearrangements are often helpful in devising efficient algorithms. For more information on alternative formulations, see, for example, [Strang 80].

5.2.2 Special matrices: diagonal, tridiagonal, symmetric, triangular, identity, and zero

The list of definitions follows:

$$
\begin{aligned}
(\text{diagonal}: \ \mathrm{D}) & \\
d_{i,j} &= \begin{cases} 0 & (i \neq j) \\ \text{arbitrary} & (i = j) \end{cases} \\
(\text{tridiagonal}: \ \mathrm{T}) & \\
t_{i,j} &= \begin{cases} 0 & (|i-j| > 1) \\ \text{arbitrary} & (|i-j| \leq 1) \end{cases} \\
(\text{symmetric}: \ \mathrm{A}) & \\
a_{i,j} &= a_{j,i} \quad (\text{all } i,j) \ . \\
(\text{upper triangular}: \ \mathrm{U}) & \\
u_{i,j} &= \begin{cases} 0 & (i > j) \\ \text{arbitrary} & (i \leq j) \end{cases} \\
(\text{lower triangular}: \ \mathrm{L}) & \\
l_{i,j} &= \begin{cases} 0 & (i < j) \\ \text{arbitrary} & (i \geq j) \end{cases} \\
(\text{identity}: \ \mathrm{I}) & \\
(I)_{i,j} &= \begin{cases} 0 & (i \neq j) \\ 1 & (i = j) \end{cases} \\
(\text{zero}: \ \Phi) & \\
\phi_{i,j} &= 0 \quad (\text{all } i,j)
\end{aligned}
$$

5.2.3 Minor, determinant, transpose, inverse, singular

Let $A^{(n)}$ be order n. The (i,j) *minor* of $A^{(n)}$ is the order $n-1$ square obtained be deleting the i^{th} row and j^{th} column of $A^{(n)}$. For example, the

$(1,3)$ minor of $A^{(5)}$, which we denote $(A^{(5)})_{1,3}$, is

$$(A^{(5)})_{1,3} = \begin{pmatrix} a_{2,1} & a_{2,2} & a_{2,4} & a_{2,5} \\ a_{3,1} & a_{3,2} & a_{3,4} & a_{3,5} \\ a_{4,1} & a_{4,2} & a_{4,4} & a_{4,5} \\ a_{5,1} & a_{5,2} & a_{5,4} & a_{5,5} \end{pmatrix}.$$

We define the *determinant* of $A^{(n)}$, $\det(A)$, recursively as follows:

$$\det(A^{(1)}) = a_{11}, \tag{7}$$

$$\det(A^{(k+1)}) = \sum_{j=1}^{k+1}(-1)^{j+1}(A^{(k+1)})_{1,j}\det((A^{(k+1)})_{1,j}). \tag{8}$$

When expressed in this form we say that the determinant is expanded along the first row. We could expand along any row, or column. The determinant of a product of matrices is the product of their determinants:

$$\det(AB) = \det(A)\det(B).$$

A square matrix is said to be *nonsingular* if its columns are linearly independent; if the columns are not linearly independent then it is *singular*. (If its columns are linearly independent then its rows are also linearly independent, and conversely.) The determinant of a singular matrix is zero; conversely, if the determinant of a matrix is zero then the matrix is singular.

The *transpose* of the matrix A, denoted A^T, is the matrix obtained by interchanging rows and columns of A; for example

$$\begin{pmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{pmatrix}^T = \begin{pmatrix} 0 & 8 & 16 \\ 2 & 10 & 18 \\ 4 & 12 & 20 \\ 6 & 14 & 22 \end{pmatrix}.$$

The transpose of the product of matrices is the product of their transposes in reverse order:

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

If A is a square matrix then

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

The *inverse* of the square matrix A is a square matrix, denoted by A^{-1}, such that

$$AA^{-1} = A^{-1}A = I,$$

where I is the identity matrix. The inverse does not exist if A is singular. The inverse of a product of matrices is the product of their inverses in reverse order:

$$(AB)^{-1} = B^{-1}A^{-1}.$$

5.2.4 Matrix norms

The norm of a matrix A is a scalar, represented by $\| A \|$, with the following properties:

1. $\| A \| > 0$ if $A \neq$ a matrix of zeros;
2. $\| A \| = 0$ if and only if $A =$ a matrix of zeros;
3. If p is any scalar, then $\| pA \| = |p| \; \| A \|$;
4. If B is also a matrix, then $\| A + B \| \leq \| A \| + \| B \|$;

A matrix norm we use in this course is the infinity norm defined by

$$\| A \|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{i,j}|. \tag{9}$$

This norm has the additional important property that

$$\|Ax\|_\infty \leq \|A\|_\infty \|x\|_\infty,$$

where x is any vector. This property is often described by saying that the matrix norm $\| \; \|_\infty$ defined by equation (9) is *consistent* with the vector norm $\| \; \|_\infty$ defined by equation (6). There are also matrix norms consistent with the two vector norms $\| \; \|_1$ and $\| \; \|_2$ defined by equations (4) and (5). For more information on matrix norms, see [Conte & de Boor 80] or [Golub & Van Loan 89].

5.3 Linear equations

A system of n linear equations in n unknowns $(\, x_1 \quad x_2 \quad \ldots \quad x_n)^T\,)$ is given below.

$$\begin{array}{rcl} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n & = & b_2 \\ \vdots & \vdots & \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n & = & b_n \end{array} \tag{10}$$

This system is written in matrix form simply as

$$Ax = b. \tag{11}$$

We call A the coefficient matrix and b the right-hand side. Notice that another way of writing this equation is

$$a_{:,1}x_1 + a_{:,2}x_2 + \ldots + a_{:,n}x_n = b,$$

which shows that the problem of solving a system of simultaneous linear equations can be thought of as the problem of finding a linear combination of the columns of the coefficient matrix that is equal to the right hand side. Thus if the columns of A are linearly independent we know that there is a solution because any vector in $\mathcal{R}^n$ can be expressed as a linear combination of the members of any set of linearly independent vectors in $\mathcal{R}^n$. On the other hand, even when A is singular the equations have a solution if it happens that b is expressible as a linear combination of the columns of A; or, more technically, if b lies in the space spanned by the columns of A. For example if

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 5 \\ 1 & 8 & 9 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 6 \\ 10 \end{pmatrix},$$

then A is singular ($A_{:,3} = A_{:,1} + A_{:,2}$) but there is a solution:

$$x = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix}.$$

Indeed there is an infinity of solutions. On the other hand, if

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

then there is no solution.

Formally, we can express the solution by

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

(This comes from multiplying both sides of equation (11) by A^{-1} and using the fact that $A^{-1}A = I$.) But without knowing A^{-1} this is useless, and computing A^{-1} is more difficult than solving the problem another way: Gaussian elimination. Before discussing Gaussian elimination we consider a special problem: solve equation (11) when A is triangular.

5.3.1 Solving a triangular system

Suppose that

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} \\ 0 & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} \\ 0 & 0 & a_{3,3} & a_{3,4} & a_{3,5} \\ 0 & 0 & 0 & a_{4,4} & a_{4,5} \\ 0 & 0 & 0 & 0 & a_{5,5} \end{pmatrix}, \tag{12}$$

and that none of the $a_{i,i}$ is zero, then there is an obvious procedure for solving the equations represented by equation (12). Starting with the last row of A and working towards the first we see that

$$\begin{aligned} x_5 &= b_5/a_{5,5}, \\ x_4 &= (b_4 - a_{4,5}x_5)/a_{4,4}, \\ x_3 &= (b_3 - a_{3,4}x_4 - a_{3,5}x_5)/a_{3,3}, \\ x_2 &= (b_2 - a_{2,3}x_3 - a_{2,4}x_4 - a_{2,5}x_5)/a_{2,2}, \\ x_1 &= (b_1 - a_{1,2}x_2 - a_{1,3}x_3 - a_{1,4}x_4 - a_{1,5}x_5)/a_{1,1}. \end{aligned}$$

This procedure is called *backsolving*. It is not difficult to verify that the computational work required for backsolving is $\mathcal{O}(n^2)$, where n is the order of the coefficient matrix.

The Gaussian elimination algorithm can be used to reduce a general system of linear equations to triangular form.

5.3.2 Gaussian elimination

Gaussian elimination uses *elementary row transformations* to reduce a matrix to upper triangular form. An elementary row transformation modifies a row, say row i, of a matrix according to the formula:

$$a_{i,:}^{(new)} = a_{i,:}^{(old)} + m a_{j,:}^{(old)} \quad (i \neq j), \tag{13}$$

where m is a scalar. Notice that if

$$m = -\frac{a_{i,k}^{(old)}}{a_{j,k}^{(old)}}, \quad \text{then} \quad a_{i,k}^{(new)} = 0.$$

Another important point is that making the same elementary row transformation on a row of the coefficient matrix and on the right-hand side vector, leaves the solution to the system of equations unchanged: if x' is a solution of equation (11) then

$$A^{(old)}x' = b^{(old)},$$

and

$$A^{(new)}x' = b^{(new)}.$$

Thus by a succession of properly chosen elementary row transformations it is possible to make all elements of A below the main diagonal equal to zero, producing a triangular system which has the same solution as the original system.[2]

Gaussian elimination transforms the coefficient matrix to triangular form in stages: in the first stage $n-1$ elementary row transformations make all elements in the first column below the main diagonal equal to zero; in the second stage $n-2$ elementary row transformations make all elements in the second column below the main diagonal equal to zero; and so on until the last column is reached. Thus starting with the initial coefficient matrix $A^{(0)}$, a sequence of matrices $A^{(1)}, A^{(2)}, \ldots, A^{(n-1)}$, and a corresponding sequence of right-hand side vectors is generated, with the final system

$$A^{(n-1)}x = b^{(n-1)}$$

[2]This is true only in a theoretical sense. In practice, roundoff error during Gaussian elimination perturbs the equations so that their solution is changed a little or a lot depending on circumstances. If the coefficient matrix is nearly singular, for example, then the solution may be changed a lot.

having triangular form.

We illustrate the procedure for a system with $n = 4$.

$$A^{(0)} = \begin{pmatrix} 2.0000 & 1.0000 & -1.0000 & 4.0000 \\ 1.0000 & -0.5000 & 1.5000 & 3.0000 \\ 0.5000 & -0.2500 & 3.7500 & 2.5000 \\ 0.2500 & -0.1250 & 0.7500 & 1.8750 \end{pmatrix}, \quad b^{(0)} = \begin{pmatrix} 1.0000 \\ -0.5000 \\ 2.2500 \\ -3.8125 \end{pmatrix}. \tag{14}$$

After the first three elementary row transformation the first stage is completed and we have

$$A^{(1)} = \begin{pmatrix} 2.0000 & 1.0000 & -1.0000 & 4.0000 \\ 0 & -1.0000 & 2.0000 & 1.0000 \\ 0 & -0.5000 & 4.0000 & 1.5000 \\ 0 & -0.2500 & 0.8750 & 1.3750 \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} 1.0000 \\ -1.0000 \\ 2.0000 \\ -3.9375 \end{pmatrix}.$$

The elementary row transformations made here are given by the following equations:

$$A^{(1)}_{2:4,:} = A^{(0)}_{2:4,:} - \frac{a^{(0)}_{2:4,1}}{a^{(0)}_{1,1}} a^{(0)}_{1,:}, \tag{15}$$

$$b^{(1)}_{2:4} = b^{(0)}_{2:4} - \frac{a^{(0)}_{2:4,1}}{a^{(0)}_{1,1}} b^{(0)}_{1}. \tag{16}$$

The row $a^{(0)}_{1,:}$ in the second term on the right of equation (15) is called the *pivot row* and the main diagonal element in the pivot row, $a^{(0)}_{1,1}$, is called the *pivot element*. The factor multiplying the pivot row is the vector of multipliers (m in equation (13)):

$$-\frac{a^{(0)}_{2:4,1}}{a^{(0)}_{1,1}} = \begin{pmatrix} -1/2 \\ -1/4 \\ -1/8 \end{pmatrix}.$$

At the end of the second stage we have

$$A^{(2)} = \begin{pmatrix} 2.0000 & 1.0000 & -1.0000 & 4.0000 \\ 0 & -1.0000 & 2.0000 & 1.0000 \\ 0 & 0 & 3.0000 & 1.0000 \\ 0 & 0 & 0.3750 & 1.1250 \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} 1.0000 \\ -1.0000 \\ 2.5000 \\ -3.6875 \end{pmatrix}.$$

In this stage the pivot row is the second row of $A^{(1)}$ and the pivot element is $a^{(1)}_{2,2}$. Finally, at the end of the third stage we have

$$A^{(3)} = \begin{pmatrix} 2.0000 & 1.0000 & -1.0000 & 4.0000 \\ 0 & -1.0000 & 2.0000 & 1.0000 \\ 0 & 0 & 3.0000 & 1.0000 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}, \quad b^{(3)} = \begin{pmatrix} 1.0000 \\ -1.0000 \\ 2.5000 \\ -4.0000 \end{pmatrix}.$$

In this stage the pivot row is the third row of $A^{(2)}$ and the pivot element is $a^{(2)}_{3,3}$.

Now the system has been reduced to upper triangular form and the solution can be obtained by backsolving. The number of arithmetic operations required for transforming an order n matrix to triangular form by Gaussian elimination is $\mathcal{O}(n^3)$.

5.3.3 Pivoting

If we exchange a pair of equations in the original system this cannot change the solution: exchanging a pair of equations simply amounts to relabelling them. Thus, in the process of Gaussian elimination we are free to exchange rows of the coefficient matrix and the righthand side without affecting the solution. If the pivot element is zero, then we must exchange rows to avoid division by zero when we compute the scalar multiplier, m, for the elementary row transformation. But, aside from this, it is the practice to exchange rows during Gaussian elimination because analysis and experience shows that greater accuracy is usually obtained if the pivot element is large: the rule for exchanging rows to make the pivot element large is called *partial pivoting.*

Partial pivoting strategy can be described as follows, assuming that stage $k-1$ has just been completed.

- Find the element of maximum magnitude in $a^{(k)}_{k:n,k}$, call it $a^{(k)}_{r,k}$, $k \leq r \leq n$.
- Exchange row r and row k.

We illustrate with a simple example. Suppose that in the last example we started out with the same set of equations but had them in a different order, say with the second and fourth equations interchanged so that the system is:

$$A^{(0)} = \begin{pmatrix} 2.0000 & 1.0000 & -1.0000 & 4.0000 \\ 0.2500 & -0.1250 & 0.7500 & 1.8750 \\ 0.5000 & -0.2500 & 3.7500 & 2.5000 \\ 1.0000 & -0.5000 & 1.5000 & 3.0000 \end{pmatrix}, \quad b^{(0)} = \begin{pmatrix} 1.0000 \\ -3.8125 \\ 2.2500 \\ -0.5000 \end{pmatrix}.$$

Stage 0 has just been "completed" so $k = 1$. The element of maximum magnitude in column 1 is 2.0. Since this element is already in row 1 no exchange is necessary and we proceed with stage 1. Then, at the end of the first stage, we have

$$A^{(1)} = \begin{pmatrix} 2.0000 & 1.0000 & -1.0000 & 4.0000 \\ 0 & -0.2500 & 0.8750 & 1.3750 \\ 0 & -0.5000 & 4.0000 & 1.5000 \\ 0 & -1.0000 & 2.0000 & 1.0000 \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} 1.0000 \\ -3.9375 \\ 2.0000 \\ -1.0000 \end{pmatrix}.$$

Now $k = 2$ and the element of maximum magnitude in $a^{(1)}_{2:4,2}$ is -1.0 in row 4; therefore, we exchange rows 2 and 4. Then we proceed with stage 2, and so forth.

5.3.4 The least squares problem

When the matrix A is $n \times n$, the linear system $Ax = b$ has the unique solution $x = A^{-1}b$ as long as the rank of A is n. On the other hand, when the matrix A is $m \times n$ with $m > n$, there are more equations than unknowns to satisfy them, and the system may have no exact solutions. Such a system is termed *overdetermined*.

When the system is overdetermined, we often seek the solution $\hat{x}$ that minimizes the 2-norm of the residual error:

$$e = \| A\hat{x} - b \|_2.$$

(Note that if $A\hat{x} = b$ exactly, the residual error e is zero.)

The solution $\hat{x}$ that minimizes e is said to satisfy the system *in the least squares sense*. It is the solution that minimizes the Euclidean distance between the exact righthand side b and the approximate one $A\hat{x}$. There are several numerical methods for finding $\hat{x}$ and so solving the *least squares problem*, but we do not review them here. For more details, see CdB, DB, KMN, or [Golub & Van Loan 89].

A common special case of the least squares problem arises when we want to fit a set of data points with a polynomial. Suppose, for example, that we are studying a physical phenomenon known to be modelled by a quadratic. (An example of this is the parabolic trajectory followed by a projectile under the influence of gravity.) In this case, we expect the measured data points (x_1, y_1), (x_2, y_2), (x_3, y_3), and (x_4, y_4) to satisfy, at least approximately, the quadratic $a_1 x_i^2 + a_2 x_i + a_3 = y_i$. That is, the coefficients of the polynomial should satisfy the linear system

$$\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}.$$

This system, however, is overdetermined, and we can at best expect to find a least squares approximation to $a = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 \end{pmatrix}^T$. This approximation can often be a very good one, meaning that the residual error is very small.

5.4 Eigenvalues and eigenvectors

For a given matrix, A, there are certain vectors, $\hat{x}$ that have a special property illustrated by the formula

$$A\hat{x} = \lambda\hat{x}. \tag{17}$$

This equation says that the effect of multiplying $\hat{x}$ by A is the same as multiplying $\hat{x}$ by the scalar λ. Here is an example:

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

The vector $\hat{x}$ is called an eigenvector of A, and the scalar λ is called an eigenvalue of A. Eigenvectors are determined up to a scalar multiplier. This is obvious from equation (17): we can multiply both sides by a scalar and this is the same as multiplying $\hat{x}$ by a scalar. Generally eigenvectors are *normalized* in some way: typically by requiring their norm to have a certain value — e.g., $\| \hat{x} \|_2 = 1$.

An order n matrix has n eigenvalues. Two or more eigenvalues may be equal, and eigenvalues may be complex. In most cases, a computation is necessary to determine the eigenvalues and eigenvectors of a matrix, but there are some simple cases where the eigenvalues are obvious. The eigenvalues of a diagonal matrix are the numbers on the main diagonal, and the eigenvectors are the unit vectors. The eigenvalues of a triangular matrix are also the numbers on the main diagonal, but its eigenvectors are not the unit vectors.

An order n matrix may have up to n linearly independent eigenvectors. A symmetric matrix of order n has n linearly independent eigenvectors. In fact, the eigenvectors of a symmetric matrix are orthogonal to one another.

Eigenvalues and eigenvectors are important for studying the motion of a physical system. For example, studies of the motion of atoms in a solid lead to a matrix whose eigenvalues represent frequencies of oscillation of the atoms. The eigenvectors represent sets of amplitudes for the oscillatory motion.

Eigenvectors and eigenvalues are also important in mathematical analysis; for example, in analyzing the effect of multiplying a vector by a given matrix many times. Suppose that A is a real symmetric matrix of order n with eigenvectors $\hat{x}^{(1)}, \hat{x}^{(2)}, \ldots, \hat{x}^{(n)}$ which are real and linearly independent since A is real and symmetric. Now let y be an arbitrary vector in $\mathcal{R}^n$. Since A has n linearly independent eigenvectors in $\mathcal{R}^n$, we can express any vector, in particular y, in terms of them. Thus,

$$y = \sum_{i=1}^{n} c_i \hat{x}^{(i)},$$

where the c_i's are certain scalars. From this, and equation (17), it follows that

$$A^n y = \sum_{i=1}^{n} c_i \lambda^n \hat{x}^{(i)}.$$

The problem of analyzing the behavior of $A^n y$ as a function of n and y comes up in studying numerical algorithms for solving differential equations.

6 Differential equations

The general form of a *first order* differential equation is

$$y' = f(x, y). \tag{18}$$

This equation is called "first order" because the highest order derivative it contains is the the first derivative: the form of a second order differential equation is

$$y'' = f(x, y, y').$$

The problem of solving these equations amounts to determining y as a function of x on some interval, say $x \in [a, b]$: in what follows, $[a, b] = [0, 1]$.

For a first order equation a condition is necessary to get a particular solution; e.g., specification of the value of y at $x = 0$. (There is a whole family of solutions to the differential equation: the condition determines one member of the family.) For example,

$$y' = y \quad \text{has the solution } y = ce^x, \tag{19}$$

where c is an arbitrary constant. The family of solutions is generated by choosing different values of c. If we require that $y(0) = 2$, then it follows that $c = 2$ and we have a particular solution $2e^x$ for this differential equation.

With a second order differential equation two conditions are required to determine a particular solution. For example, we might require $y(0) = 1$ and $y'(0) = 0$, or we might require $y(0) = 1$ and $y(1) = 0$: in the first case we have an *initial value* problem, in the second a *boundary value* problem.

In practice most differential equations must be solved numerically. A numerical solution consists of a particular solution at a set of points; i.e., $y_1 = y(x_1), y_2 = y(x_2), \ldots$. The simplest way to generate such a solution is by using the first terms of Taylor's series.

6.1 Euler's method

Euler's method uses $\mathcal{T}_1(y(x))$ to generate the solution of a first order system with an initial condition. Thus, to solve equation (18) numerically on the interval $[0, 1]$ with the initial condition $y(0) = y_0$ we proceed as follows. Let $x_0, x_1, \ldots, x_n$ be a set of equally spaced points on $[0, 1]$ with

$$x_0 = 0,\ x_n = 1,\ x_{i+1} - x_i = h = 1/n,\ (0 \le i < n).$$

Then we compute the solution on these points according to the formula:

$$y_{i+1} = y_i + f(x_i, y_i)h. \tag{20}$$

The expression on the right is $\mathcal{T}_1(y(x_i+h))$: notice that $f(x_i, y_i) = y'(x_i)$. The initial condition defines y_0:

$$y(x_0) = \alpha \Rightarrow y_0 = \alpha.$$

Knowing y_0 we can evaluate the right side of equation (20) to obtain y_1, then we can compute y_2 in a similar manner, and so on. This is Euler's method.

In our notation, we make a distinction between the exact solution at x_i and the solution given by Euler's method: $y(x_i)$ is the exact solution; y_i is the solution given by Euler's method: when we want to be explicit about the value of h used we write $y_{i;h}$ in place of y_i.

To illustrate Euler's method we return to the differential equation in equation (19). In this case $f(x_i, y_i) = y_i$ so equation (20) becomes

$$y_{i+1} = y_i + y_i h. \tag{21}$$

If the initial condition is $y(x_0) = 1$ and we use $h = 0.1$, then $y_1 = 1.1$, $y_2 = 1.21$, $y_3 = 1.331$, and so on.

At each step, truncation error is introduced into the solution because equation (20) does not give the exact solution but only an approximation: indeed we know from the remainder for Taylor's approximation that after the first step

$$\begin{aligned} y(x_1) - y_{1;h} &= \frac{y''(x_0+\xi)}{2}h^2, \quad (\xi \in (0,h)) \\ &= \frac{y(x_0+\xi)}{2}h^2. \end{aligned} \tag{22}$$

The truncation error introduced at each step is called the *local truncation error*. We see from equation (22) that the local truncation error in Euler's method is described as $\mathcal{O}(h^2)$.

The total truncation error after k steps, due to the accumulation of local truncation errors, is called the *global truncation error*. The expression

$$y(x_k) - y_{k;h}$$

gives the global truncation error after k steps. A subtle but important point needs to be made here. The local truncation error introduced at the k^{th} step is the difference between the computed solution at x_k and an exact solution at

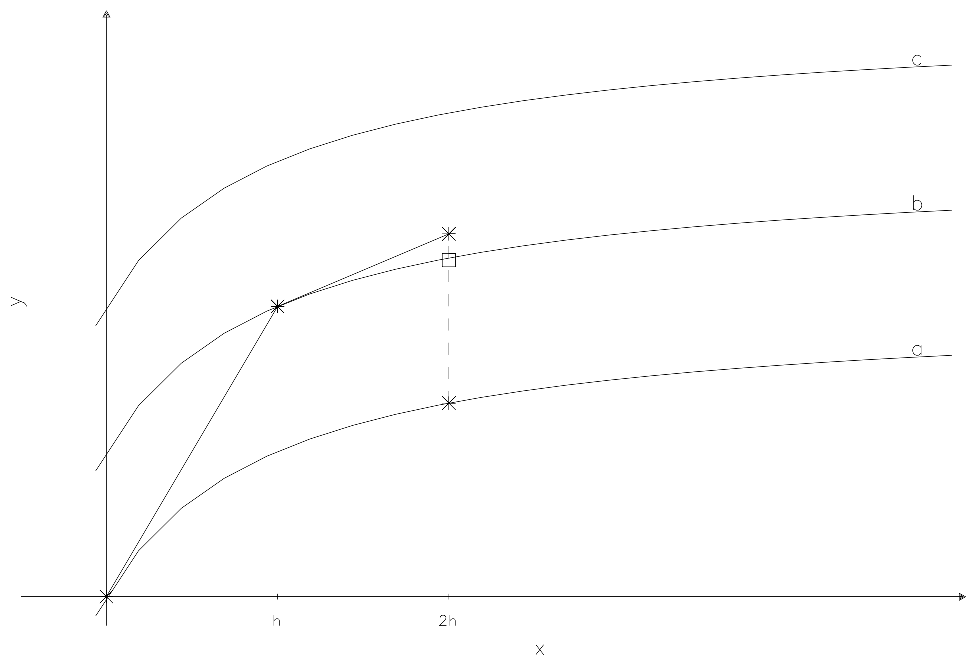

Figure 2: Illustration of the difference between local error and global error. Curves a, b, c are solution curves for the differential equation. The particular solution that satisfies the initial conditions is curve a. The straight-line segments represent the numerical solution. The vertical dashed line extending from the numerical solution down to curve a represents the global error — the accumulated error after two steps; the portion of this dashed line extending just down to curve b, at the box, represents the local error — the error made in the last step of the numerical solution.

x_k— *an exact solution that was equal to* y_{k-1} *at* x_{k-1}. The idea is illustrated in figure 2.

In general the global error is difficult to estimate. However, it is easily computed for the problem above in which the solution was generated by equation (21). We see from this equation that

$$y_n = (1+h)^n y_0.$$

If $y_0 = 1$ the exact solution for the problem is e^x so the global error is given by

$$E_{global} = e^{nh} - (1+h)^n.$$

Expanding the expression on the right to terms involving h^2 we see that

$$\begin{aligned} E_{global} &= (1 + nh + \frac{(nh)^2}{2} + \ldots) - (1 + nh + \frac{n(n-1)h^2}{2} + \ldots) && (23) \\ &= \frac{x_n}{2}h + \ldots && (24) \end{aligned}$$

Thus the global error at x_n is $\mathcal{O}(h)$. While this is a special case it can be shown for the general problem, equation (18), that the global error is $\mathcal{O}(h)$ provided that $y''(x)$ and $\partial f(x,y)/\partial y$ are bounded in the interval of interest; for details see CdB.

Next consider solving the second order initial value problem

$$y'' = f(x,y,y') \quad (\text{initial values} : y(0) = \alpha,\ y'(0) = \beta)$$

by Euler's method. We define $z = y'$ and express the problem in terms of a pair of first-order equations in the variables $x, y,\ z$:

$$\begin{aligned} y' &= z, \\ z' &= f(x,y,z), \end{aligned}$$

with initial conditions

$$y(0) = \alpha, \quad z(0) = \beta.$$

Euler's method in this case is described by the formulas:

$$\begin{aligned} y_{i+1} &= y_i + z_i h, \\ z_{i+1} &= z_i + f(x_i, y_i, z_i)h. \end{aligned}$$

The initial conditions allow computation of y_1, z_1. Substituting these values on the right side then gives us y_2, z_2, and so forth. In a similar way, a third order differential equation can be expressed as three first order differential equations which can be solved in a similar manner, and so on for any order differential equation

6.2 Finite difference methods

These methods are based on *finite difference* approximations of derivatives. Simple finite difference approximations for the first derivative are

$$y'(x) \approx \frac{y(x+\delta x) - y(x)}{\delta x}, \quad y'(x) \approx \frac{y(x) - y(x-\delta x)}{\delta x}. \tag{25}$$

The first is called a *forward difference*, the second a *backward difference*. Using Taylor's series it is easy to verify that the error in each of these approximations is $\mathcal{O}(\delta x)$.

The *central difference* is more accurate: it is

$$y'(x) \approx \frac{y(x+\delta x) - y(x-\delta x)}{2\delta x}.$$

The error in this approximation is $\mathcal{O}(\delta x^2)$. There is a similar, $\mathcal{O}(\delta x^2)$, approximation for the second derivative:[3]

$$y''(x) \approx \frac{y(x+\delta x) - 2y(x) + y(x-\delta x)}{\delta x^2}.$$

Such approximations can be substituted for derivatives to obtain a finite difference method for solving a differential equation. We illustrate with a simple example.

Consider the second order initial value problem

$$y'' = y \quad (\text{initial values} : y(0) = \alpha, y'(0) = \beta).$$

Replacing y'' by its difference approximation leads to the formula:

$$y_{i+2} = (2 - \delta x^2) y_{i+1} - y_i.$$

[3] For brevity of notation we use δx^2 for $(\delta x)^2$.

Thus, if we know y_0, and y_1, we can compute y_2, and then y_3, and so on. We obtain y_1 from the formula

$$y_1 = y_0 + y_0' + \frac{y_0 \delta x^2}{2}.$$

This method is called the *explicit central difference method* in DB (see DB, p. 352-354 for a discussion of this method).

7 Fourier series

This subject is concerned with the representation of functions as linear combinations of sines and cosines. The functions may be continuous or be a finite set of discrete values. These two cases are discussed below. Both CdB and KMN are good references.

7.1 The continuous case

A Fourier series is a sum of sine and cosine terms, often written in the following form:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos(kx) + b_k \sin(kx)). \tag{26}$$

The coefficients a_k and b_k are given by

$$a_k = \frac{1}{\pi}\int_{-\pi}^{+\pi} f(x)\cos(kx)\,dx, \quad b_k = \frac{1}{\pi}\int_{-\pi}^{+\pi} f(x)\sin(kx)\,dx. \tag{27}$$

Typically, a Fourier series is used to represent a given function $f(x)$. The series may be truncated to provide an approximation of $f(x)$: we use $T(x; K)$ to denote the truncated series so that

$$f(x) \approx T(x; K) = \frac{a_0}{2} + \sum_{k=1}^{K}(a_k \cos(kx) + b_k \sin(kx)). \tag{28}$$

A function $f(x)$ can be represented as a Fourier series if it satisfies the following conditions:

1. $f(x)$ is defined at every point in the interval $[-\pi, +\pi]$;

2. $f(x)$ is single-valued, finite, and piecewise continuous with piecewise continuous first-derivatives;

3. At a point of discontinuity x_d of $f(x)$ it is assumed that the value of $f(x_d)$ is the arithmetic mean of the limits $f(x_d - \epsilon)$ and $f(x_d + \epsilon)$, $\epsilon \rightarrow 0$.

While $f(x)$ has been specified only on the interval $[-\pi, +\pi]$, it is evident that equation (26) describes a periodic function, with period 2π, on the entire $x-$axis. A function that is not periodic may also be represented by a Fourier series; this broader application of Fourier series, the Fourier integral, is briefly discussed later.

Here is a simple example of a Fourier series. Consider the function

$$f(x) = \begin{cases} 0 & \text{if } -\pi \leq x \leq -\frac{\pi}{2} \\ \frac{2}{\pi}x + 1 & \text{if } -\frac{\pi}{2} < x \leq 0 \\ 1 - \frac{2}{\pi}x & \text{if } 0 < x \leq \frac{\pi}{2} \\ 0 & \text{if } \frac{\pi}{2} < x \leq \pi \end{cases} \tag{29}$$

It has a triangular shape in the middle of the interval, so we call it the *triangular pulse*. Figure 3 shows this function. It is continuous and has three discontinuities in its first derivative, thus it satisfies the conditions enumerated above.

It is a straightforward matter to compute the Fourier coefficients given by the integrals in equation (27), thus we find

$$a_k = \left\{ \begin{array}{ll} \frac{1}{2} & \text{if } k = 0 \\ \frac{4}{(k\pi)^2} & \text{if } k \text{ is odd} \\ \frac{8}{(k\pi)^2} & \text{if } k/2 \text{ is odd} \\ 0 & \text{if } k/2 \text{ is even} \end{array} \right\}, \quad b_k = 0, \text{ for all } k.$$

Thus, the Fourier series for the triangular pulse is

$$f(x) = \frac{1}{4} + \frac{4}{\pi^2}\cos(x) + \frac{8}{(2\pi)^2}\cos(2x) + \frac{4}{(3\pi)^2}\cos(3x) + \frac{4}{(5\pi)^2}\cos(5x) + \ldots.$$

Note that the $\cos(4x)$ term is missing: $4/2$ is even, hence $a_4 = 0$. This Fourier series contains cosine terms only. We might have anticipated this at

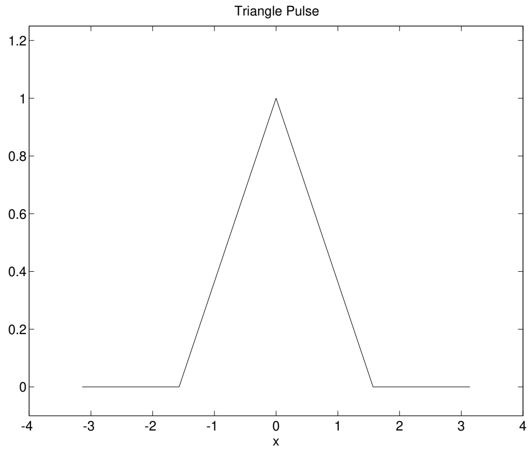

Figure 3: The triangular pulse.

the outset because the triangular pulse is an even function, $f(-x) = f(x)$, and the sine function is odd. Similarly, the Fourier series for an odd function, $f(-x) = -f(x)$, contains only sine terms.

Often a good approximation of $f(x)$ can be obtained from a truncated Fourier series with a small number of terms. If $K = 3$ in equation (28) we obtain the following approximation for the triangular pulse:

$$T(x;3) = \frac{1}{2} + \frac{4}{\pi^2}\cos(x) + \frac{8}{(2\pi)^2}\cos(2x) + \frac{4}{(3\pi)^2}\cos(3x).$$

This approximation is illustrated in figure 4 along with $T(x;7)$, $T(x;11)$, and $T(x;15)$. The illustration shows the improvement in the accuracy of the approximation as K increases. A bound on the error for any value of K is easily found by considering the truncated part of the series:

$$|f(x) - T(x;K)| < \frac{8}{\pi^2}\sum_{k=K+1}^{\infty}\frac{1}{k^2} < \frac{8}{\pi^2}\int_{K+1}^{\infty}\frac{1}{(t-1)^2}dt = \frac{8}{3\pi^2K^3}.$$

Thus, we expect the error to decrease at least as fast as $1/K^3$.

We are not restricted to the interval $[-\pi, +\pi]$. With a change of variables we can move the interval of definition to any other finite interval in the usual

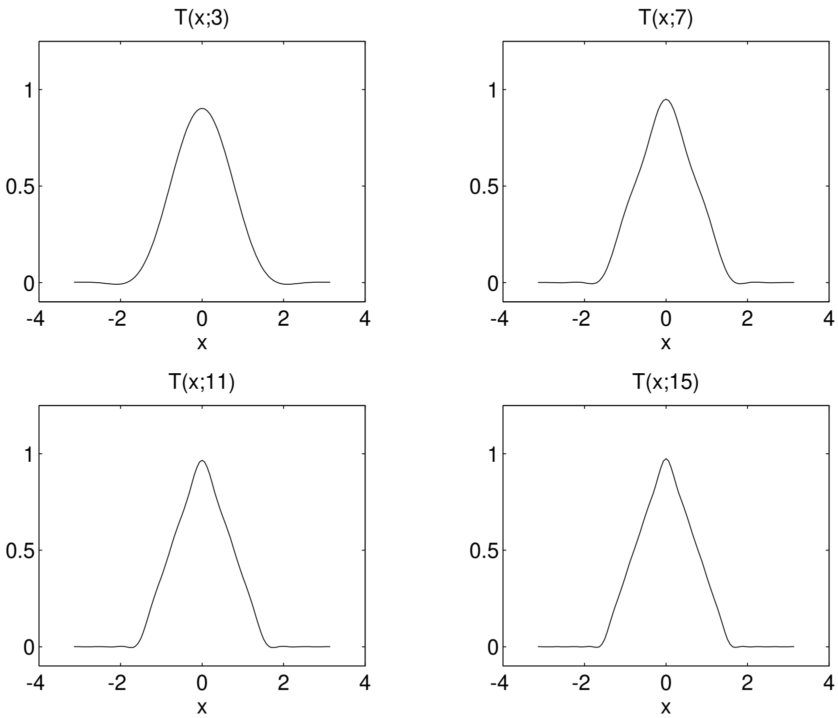

Figure 4: Four approximations for the triangular pulse

way. If $f(x)$ is defined on $[0,1]$, the Fourier series takes the form

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos(2\pi kx) + b_k \sin(2\pi kx)), \tag{30}$$

where

$$a_k = 2\int_0^1 f(x)\cos(2\pi kx)\,dx, \quad b_k = 2\int_0^1 f(x)\sin(2\pi kx)\,dx.$$

Another form of the Fourier series uses the functions e^{ikx}, $i = \sqrt{-1}$, and $k = 0, \pm 1, \pm 2, \ldots$, instead of sines and cosines. Therefore, we have the exponential form of the Fourier series:

$$f(x) = \sum_{k=-\infty}^{+\infty} c_k e^{ikx}, \tag{31}$$

where

$$c_k = \frac{1}{2\pi}\int_{-\pi}^{+\pi} f(x)e^{-ikx}\,dx.$$

This form is easily derived from equation (26) by making use of Euler's formula

$$e^{ikx} = \cos(kx) + i\sin(kx).$$

The exponential form of the Fourier series might seem more difficult to use because it employs complex variables. For numerical computations this may be true, however it is frequently more convenient than the sine-cosine form for algebraic computations. You can appreciate this if you try to simplify the product of two Fourier series.

Notice that if we define

$$z = e^{ix},$$

then the right side of equation (31) becomes

$$\sum_{k=-\infty}^{+\infty} c_k z^k.$$

Thus, we see that the Fourier series has the appearance of a power series; similarly, the truncated Fourier series has the appearance of a polynomial. For this reason a truncated Fourier series is sometimes called a *trigonometric polynomial* or a *Fourier polynomial*.

Fourier series can also be used to represent functions of more than one variable. Suppose that $f(x,y)$ is defined on the square $[-\pi,+\pi]\times[-\pi,+\pi]$. Then the the Fourier series representation for this function is

$$f(x,y) = \sum_{l=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} c_{k,l}e^{i(kx+ly)},$$

where

$$c_{k,l} = \frac{1}{4\pi^2}\int_{-\pi}^{+\pi}\int_{-\pi}^{+\pi} f(x,y)e^{-i(kx+ly)}\,dx\,dy.$$

As you might expect, the region of definition can be changed and approximations can be obtained by truncation, just as in the one-variable case.

7.2 The Fourier integral

A function defined on the interval $(-\infty,+\infty)$, and not periodic, can be represented as a Fourier integral, a kind of limiting form of a Fourier series in which the sum becomes an integral, the *Fourier integral*:

$$f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} c(u)e^{iux}du,$$

where

$$c(u) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} f(x)e^{-iux}dx.$$

Here you can recognize that the index k has become a continuous variable u, and the sum over k has become an integral over u. It is customary to express the Fourier integral in a slightly different way:

$$f(x) = \int_{-\infty}^{+\infty} F(u)e^{i2\pi ux}du,$$

where

$$F(u) = \int_{-\infty}^{+\infty} f(x) e^{-i2\pi ux} dx.$$

This form can be obtained from the first form by a change of variables: $x \to x/\sqrt{2\pi}$ and $u \to u/\sqrt{2\pi}$. The function $F(u)$ is called the *Fourier transform of* $f(x)$; and $f(x)$ is called the *inverse* of $F(u)$.

For this relationship between $f(x)$ and $F(u)$ certain assumptions are made about $f(x)$:

1. $f(x)$ is defined on $(-\infty, +\infty)$;
2. $f(x)$ and its first derivative are piecewise continuous;
3. The integral $\int_{-\infty}^{+\infty} |f(x)|\, dx$ exists and is bounded;
4. At a point of discontinuity x_d of $f(x)$ it is assumed that the value of $f(x_d)$ is the arithmetic mean of the limits $f(x_d - \epsilon)$ and $f(x_d + \epsilon)$, $\epsilon \to 0$.

As with Fourier series there is a Fourier integral for functions of more than one variable. Thus for two variables we have the Fourier integral

$$f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u, v) e^{i2\pi(ux+vy)} du\, dv,$$

and its Fourier transform

$$F(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-i2\pi(ux+vy)} dx\, dy.$$

7.3 The Convolution Theorem

The notion of a convolution arises when we want to smooth a function by taking a weighted local average of its values; that is, replace the value of $f(x)$ at x_a by a weighted average of its values taken over an interval centered on x_a for all x_a. Specifically, a *convolution* of $f(x)$ and $g(x)$, the weighting function, is denoted $f * g$ and defined by

$$f * g = \int_{-\infty}^{+\infty} f(t) g(x - t)\, dt. \tag{32}$$

Note that the convolution is a function of x.

The Convolution Theorem states that the Fourier transform of the convolution of $f(x)$ and $g(x)$ is the product of the Fourier transforms of $f(x)$ and $g(x)$, that is

$$\int_{-\infty}^{+\infty} f * g e^{-i2\pi ux} dx = F(u)G(u).$$

This result has the important consequence of greatly simplifying the computation of a convolution in many cases. If $F(u)$ and $G(u)$ are known then

$$\int_{-\infty}^{+\infty} F(u)G(u)e^{i2\pi ux} du \tag{33}$$

gives the convolution $f * g$. In practice it may be easier to compute the convolution by first computing the Fourier transforms $F(u)$ and $G(u)$, then evaluating the integral equation (33), rather than compute it directly from equation (32).

7.4 The discrete case

We now consider the case in which f denotes a discrete and finite set of values, rather than the infinite set which we denoted with $f(x)$. In particular, let f be the set $\{f_0, f_1, \ldots, f_{K-1}\}$. In applications this set usually consists of measured values of a physical quantity. We will assume that K is even, only to simplify the discussion a little. The *discrete Fourier series representation* of f is a trigonometric polynomial:

$$f_j = \frac{a_0}{K} + \frac{2}{K}\sum_{k=1}^{K/2-1}(a_k \cos(2\pi k\frac{j}{K}) + b_k \sin(2\pi k\frac{j}{K})) + \frac{a_{K/2}}{K}, \ \ j = 0, 1, \ldots, K{-}1,$$

where

$$a_k = \sum_{j=0}^{K-1} f_j \cos(2\pi k\frac{j}{K}), \quad b_k = \sum_{j=0}^{K-1} f_j \sin(2\pi k\frac{j}{K}).$$

Comparison with the Fourier series for the continuous case, equation (30), shows that this corresponds to using the set of points $x_0, x_1, \ldots, x_{K-1}$, where $x_j = j/K$, as the points at which $f(x)$ is evaluated.

The exponential form is

$$f_j = \frac{1}{K} \sum_{k=-K/2}^{K/2-1} F_k e^{i2\pi k \frac{j}{K}}, \quad j = 0, 1, \ldots, K-1,$$

where

$$F_k = \sum_{j=0}^{K-1} f_j e^{-i2\pi k \frac{j}{K}}.$$

It is convenient to think of f and F as vectors of length K, with elements as defined above. With this understanding, f and F can be represented in matrix equations:

$$f = WF, \quad F = W^{-1} f,$$

where W is a square matrix of order K. The matrix W has a very simple structure, easily recognized with the help of a small example.

Suppose $K = 8$, and let

$$w = e^{i\frac{2\pi}{K}},$$

then

$$W = \frac{1}{K} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ w^{-4} & w^{-3} & w^{-2} & w^{-1} & 1 & w^{1} & w^{2} & w^{3} \\ w^{-8} & w^{-6} & w^{-4} & w^{-2} & 1 & w^{2} & w^{4} & w^{6} \\ w^{-12} & w^{-9} & w^{-6} & w^{-3} & 1 & w^{3} & w^{6} & w^{9} \\ w^{-16} & w^{-12} & w^{-8} & w^{-4} & 1 & w^{4} & w^{8} & w^{12} \\ w^{-20} & w^{-15} & w^{-10} & w^{-5} & 1 & w^{5} & w^{10} & w^{15} \\ w^{-24} & w^{-18} & w^{-12} & w^{-6} & 1 & w^{6} & w^{12} & w^{18} \\ w^{-28} & w^{-21} & w^{-14} & w^{-7} & 1 & w^{7} & w^{14} & w^{21} \end{pmatrix}.$$

Thus computation of the Fourier transform of f when $K = 8$ consists in multiplying f by this matrix. The amount of arithmetic required is 64 multiplications and about the same number of additions, in general about $2K^2$ arithmetic operations. This number of operations can be reduced to about $2K \log_2(K)$ by employing a clever idea known as the *Fast Fourier Transform (FFT)*. We do not discuss the FFT here, but it is worth pointing out that it takes advantage of the special structure of W. The elements of W^{-1} should be evident from this example. For more information on the FFT, see the Tomography tutorial [Jessup 95].

F is called the *discrete Fourier transform* of f, and f is called the *inverse Fourier transform* of F. Also, F is sometimes called the representation of f in *frequency space*: deriving from the fact that each element in F corresponds to a distinct value of k, which appears as a frequency in the above equations.

The equations for the discrete Fourier transform are written in other ways. Sometimes they are written

$$f_j = \sum_{k=-K/2}^{K/2-1} F_k e^{i2\pi k \frac{j}{K}}, \quad j = 0, 1, \ldots, K-1,$$

where

$$F_k = \frac{1}{K} \sum_{j=0}^{K-1} f_j e^{-i2\pi k \frac{j}{K}},$$

or they are written

$$f_j = \frac{1}{\sqrt{K}} \sum_{k=-K/2}^{K/2-1} F_k e^{i2\pi k \frac{j}{K}}, \quad j = 0, 1, \ldots, K-1,$$

where

$$F_k = \frac{1}{\sqrt{K}} \sum_{j=0}^{K-1} f_j e^{-i2\pi k \frac{j}{K}}.$$

References

[Conte & de Boor 80] CONTE, S. D. AND CARL de BOOR. [1980]. *Elementary Numerical Analysis: An Algorithmic Approach.* McGraw-Hill, Inc., New York, NY, 3rd edition.

[Dahlquist & Björck 74] DAHLQUIST, GERMUND AND ÅKE BJÖRCK. [1974]. *Numerical Methods.* Prentice-Hall, Inc., Englewood Cliffs, NJ. Translated by Ned Anderson.

[Fosdick 95] FOSDICK, LLOYD D. [1995]. IEEE arithmetic short reference. HPSC Course Notes.

[Golub & Van Loan 89] GOLUB, G.H. AND C.F. Van LOAN. [1989]. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 2nd edition.

[Jessup 95] JESSUP, ELIZABETH R. [1995]. Computerized tomography: An introduction. HPSC Course Notes.

[Kahaner et al 89] KAHANER, DAVID, CLEVE MOLER, AND STEPHEN NASH. [1989]. *Numerical Methods and Software.* Prentice-Hall, Inc., Englewood Cliffs, NJ. Software on disk with book.

[Strang 80] STRANG, G. [1980]. *Linear Algebra and Its Applications.* Academic Press, Inc., New York, NY.