Last updated: 21 July 2015

Chapter 1

INTRODUCTION

1.1 Numerical methods and analysis

When solving a mathematical problem, such as determining a definite integral or solving a differential equation, we attempt to do so *analytically*—we determine an expression for the indefinite integral and then substitute the limits, or we apply an appropriate technique to find an expression that relates the dependent and independent variables of the DE. The precise definition of an analytic expression or solution is one that can be expressed in terms of a bounded number of certain elementary functions: constants (including complex numbers), one variable x, elementary operations of arithmetic ($+ - \times \div$), n-th roots, exponents (which includes trigonometric functions and inverse trigonometric functions) and logarithms. However, we are often confronted with mathematically posed problems that simply cannot be solved analytically, such as the transcendental equation

$$\sin x - 0.625x = 0 \tag{1.1}$$

or the non-linear differential equation

$$\frac{dN}{dt} = aN - k(t)N^{1.7} \tag{1.2}$$

both of which we shall revisit in later chapters. Problems which cannot be solved analytically are generally nonlinear in nature, which is clearly the case with these two equations.

The only hope we have of dealing with such problems is by finding a *numerical solution.* In equation (1.1) this would be a numerical value for x that satisfies the equation; in equation (1.2) the numerical solution is a set of numbers that represents the true solution over the relevant interval of integration. *Numerical methods* are the mathematical tools we use to find such numerical solutions.

1.1.1 Numerical analysis.

As the term suggests, *numerical analysis* is the mathematical study of numerical methods and, in the broader sense, the analysis of the field of numerical methods as a whole. Numerical analysis addresses the following:

(a) The derivation of numerical methods from fundamental mathematical ideas.

(b) Investigating the properties of numerical methods, such as accuracy and stability.

Numerical methods tend to be approximation techniques, i.e. they yield approximate solutions rather than exact solutions. Through appropriate analysis of the method, we may understand the nature of the *approximation error* and we will probably be able to successfully implement the method subject to a desired level of accuracy. If, through appropriate analysis of Consequently, the analysis of numerical methods is an extremely important part of the field.

1.1.2 Types of methods.

In these notes, we study the following methods:

(a) Nonlinear algebraic equations in one variable - chapter 3 - we investigate the bisection method, linear interpolation, Newton's method, and fixed-point iteration.

(b) Systems of linear equations - chapter 4 - we investigate the Jacobi method that can be used to find numerical solutions to large systems of linear equations.

(c) Approximation of functions and data sets - chapter 5 - we investigate polynomial and Lagrange interpolation, least-squares polynomial fitting, and Chebyshev series.

(d) Numerical differentiation - chapter 6 - Approximations to derivatives of various order using Taylor series.

(e) Numerical integration (quadrature) - chapter 7 - we study the Trapezium and Simpson methods, based on Lagrange interpolation, used to approximate definite integrals.

(f) Initial-value problems - chapter 8 - Euler's method, modified Euler method, Runge-Kutta methods of orders two and four, used to approximate initial-value problems arising from ordinary differential equations.

We will devote our efforts to the derivation and error analysis of most of these methods, and we will demonstrate the methods by means of numerical examples. Consequently, to a large extent, these notes are theoretical in nature.

1.2 Number representations and round-off errors

Numbers play an important part of numerical analysis (and mathematics in general with a whole field dedicated to their study). We normally think of numbers as constant-valued entities and use the numerals $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$, and combinations thereof, to represent them, e.g. 2, 42, 5345, etcetera. A k-digit number is represented by using k numerals. It should be clear the k must be a natural number, and we refer to the number as a finite digit number. It is also quite possible for a number to be a ∞-digit number, we simply refer to such a number as an infinitely digit number. The natural numbers, integers, and rational numbers may all be represented as finite digit numbers while the irrational and real numbers are mostly infinite digit numbers. For example, 1 is a 1-digit number, 12 is a 2-digit number and 87543 is a 5-digit number.

1.2.1 Decimal number representation

In the study of numerical analysis, we might want to make use of a finite precision device, e.g. a calculator or computer, to perform computations. Finite precision devices tend to have limited memory and might only be able to store and represent finite digit numbers quite accurately. Due to this memory constraint, there is a very specific way of representing numbers and using these numbers in computations within a finite precision environment.

Consider the mathematical constant π, which we know to be an irrational number and may be represented as the fraction $\frac{22}{7}$. A finite precision device cannot interpret the meaning of the symbol π and we rather make use of $\frac{22}{7}$ in computations.

We know that π can also be written as

$$3.14159265358979323846264338327950288419716939937510\ldots$$

We call this representation of the number the *decimal representation* or *decimal form* of the number. As another example, we know that we can represent $\frac{1}{2}$ as 0.5. Most real numbers are represented this way.

We define the *normalised floating-point form* of a number to be

$$\pm\, 0.d_1 d_2 \ldots d_k \times 10^n, \tag{1.3}$$

where d_i, called a *decimal digit*, is an integer-valued number with $1 \leq d_1 \leq 9$ and $0 \leq d_i \leq 9$, for each $i = 2, 3, \ldots, k$. We call any number represented by (1.3) a *k-digit decimal number.* Thus, π in normalised decimal floating-point form is

$$0.31415926535897932384626433832795028841971693993751 0\ldots$$

and the normalised floating-point form of $\frac{1}{8}$, i.e. 0.125, which is a 3-digit decimal number.

Consider a positive real number y with the normalised floating-point form

$$y = 0.d_1 d_2 \ldots d_k d_{k+1} d_{k+2} \ldots \times 10^n.$$

The finite floating-point form of y, which we denote $fl(y)$, is obtained by terminating y after k decimal digits. This termination is done in one of two ways. The first way, called *truncation* or *chopping*, is done by simply getting rid of the digits $d_{k+1} d_{k+2} \ldots$, which produces

$$fl(y) = 0.d_1 d_2 \ldots d_k \times 10^n.$$

The second way, called *rounding*, is done by adding $5 \times 10^{n-(k+1)}$ to y and then chopping the result to obtain the floating-point form

$$fl(y) = 0.\delta_1 \delta_2 \ldots \delta_k \times 10^n.$$

When rounding, if $d_{k+1} \geq 5$, we add 1 to d_k to obtain $fl(y)$, this is called *rounding up*; if $d_{k+1} < 5$, we simply chop off all the digits after the first k digits, this is called *rounding down.* It should be clear that rounding down yields $\delta_i = d_i$ for each $i = 1, 2, 3, \ldots, k$, but this is not necessarily the case when rounding up.

1.2.2 Error

Round-off error refers to errors in representing numbers in a finite precision environment. For example, we know that

$$\pi = 3.14159265358979323846264338327950288419716939937510...$$

but in most desktop computers we find that

$$\pi = 3.14159265358979.$$

In other words, due to memory constraints as mentioned earlier, the value of π is rounded off to 14 decimal places. In a finite precision device this round-off process is applied to all numerical values, so that round-off error is present most of the time (an obvious exception is the integers, which do not have a fractional part). The digits lost in this rounding-off process constitute the round-off error. By and large, we do not expect round-off error to significantly compromise calculations performed on a computer (round-off error is typically of the order of 10^{-15}), although sometimes round-off errors may be amplified in the course of a computation. We generally find that the approximation errors mentioned previously are far more significant than round-off error, and so we restrict our study of error to approximation errors rather than round-off errors in these notes. Indeed, it is appropriate to regard approximation errors as a consequence of the mathematical nature of the numerical method itself, whereas round-off errors may be seen as a technological limitation—the price we pay for using finite-precision devices.