NUMERICAL ANALYSIS

KENDALL E. ATKINSON*

1. General Introduction. Numerical analysis is the area of mathematics and computer science that creates, analyzes, and implements algorithms for solving numerically the problems of continuous mathematics. Such problems originate generally from real-world applications of algebra, geometry and calculus, and they involve variables which vary continuously; these problems occur throughout the natural sciences, social sciences, engineering, medicine, and business. During the past half-century, the growth in power and availability of digital computers has led to an increasing use of realistic mathematical models in science and engineering, and numerical analysis of increasing sophistication has been needed to solve these more detailed mathematical models of the world. The formal academic area of numerical analysis varies from quite theoretical mathematical studies (e.g. see [5]) to computer science issues (e.g. see [1], [11]).

With the growth in importance of using computers to carry out numerical procedures in solving mathematical models of the world, an area known as *scientific computing* or *computational science* has taken shape during the 1980s and 1990s. This area looks at the use of numerical analysis from a computer science perspective; see [20], [16]. It is concerned with using the most powerful tools of numerical analysis, computer graphics, symbolic mathematical computations, and graphical user interfaces to make it easier for a user to set up, solve, and interpret complicated mathematical models of the real world.

1.1. Historical background. Numerical algorithms are almost as old as human civilization. The Rhind Papyrus (˜1650 BC) of ancient Egypt describes a rootfinding method for solving a simple equation; see [10, p. 88]. Archimedes of Syracuse (287-212 BC) created much new mathematics, including the "method of exhaustion" for calculating lengths, areas, and volumes of geometric figures; see [15, Chap. 2]. When used as a method to find approximations, it is in much the spirit of modern numerical integration; and it was an important precursor to the development of the calculus by Isaac Newton and Gottfried Leibnitz.

A major impetus to developing numerical procedures was the invention of the calculus by Newton and Leibnitz, as this led to accurate mathematical models for physical reality, first in the physical sciences and eventually in the other sciences, engineering, medicine, and business. These mathematical models cannot usually be solved explicitly, and numerical methods to obtain approximate solutions are needed. Another important aspect of the development of numerical methods was the creation of *logarithms* by Napier (1614) and others, giving a much simpler manner of carrying out the arithmetic operations of multiplication, division, and exponentiation.

Newton created a number of numerical methods for solving a variety of problems, and his name is attached today to generalizations of his original ideas. Of special note is his work on rootfinding and polynomial interpolation. Following Newton, many of the giants of mathematics of the 18^{th} and 19^{th} centuries made major contributions to the numerical solution of mathematical problems. Foremost among these are Leonhard Euler (1707-1783), Joseph-Louis Lagrange (1736-1813), and Karl Friedrich Gauss (1777-1855). Up to the late 1800's, it appears that most mathematicians were quite

*Depts of Mathematics and Computer Science, University of Iowa, Iowa City, Iowa

broad in their interests, and many of them were interested in and contributed to numerical analysis. For a general history of numerical analysis up to 1900, see [17].

1.2. An early mathematical model. One of the most important and influential of the early mathematical models in science was that given by Newton to describe the effect of gravity. According to this model, the force of gravity on a body of mass m due to the Earth has magnitude

$$F = \frac{Gmm_e}{r^2}$$

where m_e is the mass of the Earth, r is the distance between the centers of the two bodies, and G is the universal gravitational constant. The force on m is directed towards the center of gravity of the Earth. Newton's model of gravitation has led to many problems that require solution by approximate means, usually involving the numerical solution of ordinary differential equations.

Following the development by Newton of his basic laws of physics, these were applied by many mathematicians and physicists to give mathematical models for solid and fluid mechanics. Civil and Mechanical Engineering use these models as the basis for most modern work on solid structures and the motion of fluids, and numerical analysis has become a basic part of the work of researchers in these areas of engineering. For example, building modern structures makes major use of finite element methods for solving the partial differential equations associated with models of stress; and computational fluid mechanics is now a fundamental tool in designing new airplanes. In the 19^{th} century, phenomena involving heat, electricity, and magnetism were successfully modelled; and in the 20^{th} century, relativistic mechanics, quantum mechanics, and other theoretical constructs were created to extend and improve the applicability of earlier ideas. For a general discussion of modelling, see [26].

2. An Overview of Numerical Analysis. The following is a rough categorization of the mathematical theory underlying numerical analysis, keeping in mind that there is often a great deal of overlap between the listed areas. For a compendium on the current state on research in numerical analysis, see [14].

2.1. Numerical linear and nonlinear algebra. This refers to problems involving the solution of systems of linear and nonlinear equations, possibly with a very large number of variables. Many problems in applied mathematics involve solving systems of linear equations, with the linear system occurring naturally in some cases and as a part of the solution process in other cases. Linear systems are usually written using matrix-vector notation, $A\mathbf{x} = \mathbf{b}$, with A the matrix of coefficients for the system, $\mathbf{x}$ the column vector of the unknown variables $x_1, \ldots, x_n$, and $\mathbf{b}$ a given column vector. Solving linear systems with up to a $n = 1000$ variables is now considered relatively straightforward in most cases. For small to moderate sized linear systems (say $n \leq 1000$), the favorite numerical method is *Gaussian elimination* and its variants; this is simply a precisely stated algorithmic variant of the method of elimination of variables that students first encounter in elementary algebra. For larger linear systems, there are a variety of approaches depending on the structure of the coefficient matrix A. *Direct methods* lead to a theoretically exact solution $\mathbf{x}$ in a finite number of steps, with Gaussian elimination the best known example. In practice, there are errors in the computed value of $\mathbf{x}$ due to rounding errors in the computation, arising from the finite length of numbers in standard computer arithmetic. *Iterative methods* are approximate methods which create a sequence of approximating solutions of increasing accuracy. Linear systems are categorized according to many properties (e.g.

A may be *symmetric* about its main diagonal), and specialized methods have been developed for problems with these special properties; see [1], [18].

Nonlinear problems are often treated numerically by reducing them to a sequence of linear problems. As a simple but important example, consider the problem of solving a nonlinear equation $f(x) = 0$. Approximate the graph of $y = f(x)$ by the tangent line at a point $x^{(0)}$ near the desired root, and use the root of the tangent line to approximate the root of the original nonlinear function $f(x)$. This leads to *Newton's method* for rootfinding:

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}, \qquad k = 0, 1, 2, \ldots$$

This generalizes to handling systems of nonlinear equations. Let $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ denote a system of n nonlinear equations in n unknowns $x_1, \ldots, x_n$. Newton's method for solving this system is given by

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \delta^{(k)} \\ \mathbf{f}'\left(\mathbf{x}^{(\mathbf{k})}\right)\delta^{(k)} &= -\mathbf{f}\left(\mathbf{x}^{(\mathbf{k})}\right), \quad k = 0, 1, \ldots \end{aligned}$$

In this, $\mathbf{f}'(\mathbf{x})$ is the Jacobian matrix of $\mathbf{f}(\mathbf{x})$, and the second equation is a linear system of order n. There are numerous other approaches to solving nonlinear systems, most based on using some type of approximation using linear functions; see [24].

An important related class of problems occur under the heading of *optimization*. Given a real-valued function $f(\mathbf{x})$ with $\mathbf{x}$ a vector of unknowns, we wish to find a value of $\mathbf{x}$ which minimizes $f(\mathbf{x})$. In some cases $\mathbf{x}$ is allowed to vary freely, and in other cases there are constraints on the values of $\mathbf{x}$ which can be considered; see [27]. Such problems occur frequently in business applications.

2.2. Approximation Theory. This category covers the approximation of functions and methods based on using such approximations. When evaluating a function $f(x)$ with x a real or complex number, keep in mind that a computer or calculator can only do a finite number of operations. Moreover, these operations are the basic arithmetic operations of addition, subtraction, multiplication, and division, together with comparison operations such as determining whether $x > y$ is true or false. With the four basic arithmetic operations, we can evaluate polynomials

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n \qquad (*)$$

and rational functions, which are polynomials divided by polynomials. Including the comparison operations, we can evaluate different polynomials or rational functions on different sets of real numbers x. The evaluation of all other functions, e.g. $f(x) = \sqrt{x}$ or 2^x, must be reduced to the evaluation of a polynomial or rational function that approximates the given function with sufficient accuracy. All function evaluations on calculators and computers are accomplished in this manner. This topic is known as approximation theory, and it is a well-developed area of mathematics; for an introduction, see [3, Chap. 3,4].

One method of approximation is called *interpolation*. Consider being given a set of points (x_i, y_i), $i = 0, 1, \ldots, n$, and then finding a polynomial (*) which satisfies $p(x_i) = y_i$, $i = 0, 1, \ldots, n$. The polynomial $p(x)$ is said to interpolate the given data points. Interpolation can be performed with functions other than polynomials (although these are the most popular category of interpolating functions), with

important cases being rational functions, trigonometric polynomials, and *spline functions*. Interpolation has a number of applications. If a function is known only at a discrete set of data points $x_0, \ldots, x_n$, with $y_i = f(x_i)$, then interpolation can be used to extend the definition to nearby points x. If n is at all large, then spline functions are preferable to polynomials for this purpose. Spline functions are smooth piecewise polynomial functions with minimal oscillation, and they are used commonly in computer graphics, statistics, and other applications; see [12].

Most numerical methods for the approximation of integrals and derivatives of a given function $f(x)$ are based on interpolation. Begin by constructing an interpolating function $p(x)$ that approximates $f(x)$, often a polynomial, and then integrate or differentiate $p(x)$ to approximate the corresponding integral or derivative of $f(x)$. For an introduction to this area, see [3, Chap. 5]; and for a more complete view of numerical integration, see [25].

2.3. Solving differential and integral equations. Most mathematical models used in the natural sciences and engineering are based on ordinary differential equations, partial differential equations, and integral equations. The numerical methods for these equations are primarily of two types. The first type approximates the unknown function in the equation by a simpler function, often a polynomial or piecewise polynomial function, choosing it to satisfy the original equation approximately. Among the best known of such methods is the *finite element method* for solving partial differential equations; see [8]. The second type of numerical method approximates the derivatives or integrals in the equation of interest, generally solving approximately for the solution function at a discrete set of points. Most initial value problems for ordinary differential equations and partial differential equations are solved in this way, and the numerical procedures are often called *finite difference methods*, primarily for historical reasons. Most numerical methods for solving differential and integral equations involve both approximation theory and the solution of quite large linear and nonlinear systems. For an introduction to the numerical analysis of differential equations, see [2], [7], [22], [30], [31]; for integral equations, see [4], [9].

2.4. Effects of computer hardware. Virtually all numerical computation is carried out on digital computers, and their structure and properties affect the structure of numerical algorithms, especially when solving large linear systems. First and foremost, the computer arithmetic must be understood. Historically, computer arithmetic varied greatly between different computer manufacturers, and this was a source of many problems when attempting to write software which could be easily ported between different computers. This has been lessoned significantly with the development of the IEEE (Institute for Electrical and Electronic Engineering) standard for computer floating-point arithmetic. All small computers have adopted this standard, and most larger computer manufacturers have done so as well. For a discussion of the standard and of computer floating-point arithmetic in general, see [28].

For large scale problems, especially in numerical linear algebra, it is important to know how the elements of an array A or a vector $\mathbf{x}$ are stored in memory. Knowing this can lead to much faster transfer of numbers from the memory into the arithmetic registers of the computer, thus leading to faster programs. A somewhat related topic is that of *pipelining*. This is a widely used technique whereby the execution of computer operations are overlapped, leading to faster execution. Machines with the same basic clock speed can have very different program execution times due to differences in pipelining and differences in the way memory is accessed.

Most present-day computers are sequential in their operation, but parallel computers are being used ever more widely. Some parallel computers have independent processors that all access the same computer memory (shared memory parallel computers), whereas other parallel computers have separate memory for each processor (distributed memory parallel computers). Another form of parallelism is the use of pipelining of vector arithmetic operations. Some parallel machines are a combination of some or all of these patterns of memory storage and pipelining. With all parallel machines, the form of a numerical algorithm must be changed in order to make best use of the parallelism. For examples of this in numerical linear algebra, see [13].

3. Common Perspectives in Numerical Analysis. Numerical analysis is concerned with all aspects of the numerical solution of a problem, from the theoretical development and understanding of numerical methods to their practical implementation as reliable and efficient computer programs. Most numerical analysts specialize in small sub-areas, but they share some common concerns, perspectives, and mathematical methods of analysis. These include the following.

1. When presented with a problem that cannot be solved directly, then replace it with a "nearby problem" which can be solved more easily. Examples are the use of interpolation in developing numerical integration methods and rootfinding methods; see [3, Chaps 2,5].
2. There is widespread use of the language and results of linear algebra, real analysis, and functional analysis (with its simplifying notation of norms, vector spaces, and operators). See [5].
3. There is a fundamental concern with error, its size, and its analytic form. When approximating a problem, as above in item 1, it is prudent to understand the nature of the error in the computed solution. Moreover, understanding the form of the error allows creation of extrapolation processes to improve the convergence behavior of the numerical method.
4. *Stability* is a concept referring to the sensitivity of the solution of a problem to small changes in the data or the parameters of the problem. Consider the following well-known example. The polynomial

$$\begin{aligned} p(x) &= (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7) \\ &= x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 12132x^2 + 13068x - 5040 \end{aligned}$$

   has roots which are very sensitive to small changes in the coefficients. If the coefficient of x^6 is changed to -28.002, then the original roots 5 and 6 are perturbed to the complex numbers $5.459 \pm 0.540i$, a very significant change in values. Such a polynomial $p(x)$ is called *unstable* or *ill-conditioned* with respect to the rootfinding problem. In developing numerical methods for solving problems, they should be no more sensitive to changes in the data than the original problem to be solved. Moreover, one tries to formulate the original problem to be *stable* or *well-conditioned*. Excellent discussions of this topic are given in [21], with particular emphasis on numerical linear algebra.
5. Numerical analysts are very interested in the effects of using finite precision computer arithmetic. This is especially important in numerical linear algebra, as large problems contain many rounding errors; see [21].
6. Numerical analysts are generally interested in measuring the efficiency of algorithms. What is the cost of a particular algorithm. For example, the use of Gaussian elimination to solve a linear system $A\mathbf{x} = \mathbf{b}$ containing n

equations will require approximately $\frac{2}{3}n^3$ arithmetic operations. How does this compare with other numerical methods for solving this problem?

4. Modern Applications and Computer Software. Numerical analysis and mathematical modelling have become essential in many areas of modern life. Sophisticated numerical analysis software is being embedded in popular software packages, e.g. spreadsheet programs, allowing many people to perform modelling even when they are unaware of the mathematics involved in the process. This requires creating reliable, efficient, and accurate numerical analysis software; and it requires designing *problem solving environments* (PSE) in which it is relatively easy to model a given situation. The PSE for a given problem area is usually based on excellent theoretical mathematical models, made available to the user through a convenient graphical user interface. Such software tools are well-advanced in some areas, e.g. computer aided design of structures, while other areas are still grappling with the more basic problems of creating accurate mathematical models and accompanying tools for their solution, e.g. atmospheric modelling.

4.1. Some application areas. *Computer aided design* (CAD) and *computer aided manufacturing* (CAM) are important areas within engineering, and some quite sophisticated PSEs have been developed for CAD/CAM. A wide variety of numerical analysis is involved in the mathematical models that must be solved. The models are based on the basic Newtonian laws of mechanics; there are a variety of possible models, and research continues on designing such models. An important CAD topic is that of modelling the dynamics of moving mechanical systems. The mathematical model involves systems of both ordinary differential equations and algebraic equations (generally nonlinear); see [19]. The numerical analysis of these mixed systems, called *differential-algebraic systems*, is quite difficult but important to being able to model moving mechanical systems. Building simulators for cars, planes, and other vehicles requires solving differential-algebraic systems in real-time. See [2], [7] for a review of some pertinant numerical analysis literature.

Atmospheric modelling is important for simulating the behavior of the Earth's atmosphere, to understand the possible effect of human activities on our atmosphere. A large number of variables need to be introduced. These include the velocity $\mathbf{v}(x, y, z, t)$ in the atmosphere at position (x, y, z) and time t, the pressure $p(x, y, z, t)$, and the temperature $T(x, y, z, t)$. In addition, we must study various chemicals existing in the atmosphere and their interactions, including ozone, various chemical pollutants, carbon dioxide, and others. The underlying equations for studying $\mathbf{v}(x, y, z, t)$, $p(x, y, z, t)$, and $T(x, y, z, t)$ are partial differential equations; and the chemical kinetic interactions of the various chemicals are described using some quite difficulty ordinary differential equations; see [23]. Many types of numerical analysis procedures are used in atmospheric modelling, including computational fluid mechanics and the numerical solution of differential equations. For the numerical solution of the partial differential equations, see [30], [31].

Modern business makes much use of optimization methods in deciding how to allocate resources most efficiently. These include problems such as inventory control, scheduling, how best to locate manufacturing and storage facilities, investment strategies, and others; see [6], [29]. The numerical analysis of optimization problems was briefly discussed earlier in this article.

4.2. Computer software. Software to implement common numerical analysis procedures is very important. If it is to be shared by many users, it needs to be

reliable, accurate, and efficient. Moreover, it needs to be written so as to be easily portable between different computers. Beginning around 1970, there have been a number of government sponsored research efforts to produce high quality numerical analysis software in particular problem areas. A more recent example of such a project is the LAPACK project [1] which contains state-of-the-art programs for basic problems in numerical linear algebra. Two important online numerical analysis libraries that contain many of these large scale numerical analysis programming projects can be found at the following internet sites:

Name	URL	Location
Netlib	www.netlib.org	Oak Ridge National Laboratory
GAMS	gams.nist.gov	National Institute of Standards and Technology

The most popular programming language for implementing numerical analysis methods continues to be *Fortran*, and it too continues to be updated to meet changing needs, with *Fortran 95* being the most recent standard. Other languages are also important, with the most important ones being *C, C++,* and *Java*. Another approach to supplying numerical analysis programs and programming tools has been to create higher level problem solving environments that contain numerical, programming, and graphical tools, including some quite sophisticated numerical analysis tools to handle many basic problems. Best known of these is *MATLAB* (© The Mathworks, Inc.), a commercial package that has arguably become the most popular way to do numerical computing. For analytical mathematics computing, there are two popular commercial packages, *Maple* (© Waterloo Maple, Inc.) and *Mathematica* (© Wolfram Research, Inc.).

REFERENCES

[1] E. Anderson, et al. (1992) *LAPACK User's Guide*, SIAM Pub.

[2] U. Ascher and L. Petzold (1998) *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM Pub.

[3] K. Atkinson (1989) *An Introduction to Numerical Analysis*, 2^{nd} ed., John Wiley Pub.

[4] K. Atkinson (1997) *The Numerical Solution of Integral Equations of the Second Kind*, Cambridge Univ. Press.

[5] K. Atkinson and W. Han (2001) *Theoretical Numerical Analysis*: *A Functional Analysis Framework*, Springer-Verlag.

[6] D. Bertsimas and R. Freund (2000) *Data, Models, and Decisions*: *The fundamentals of Management Science*, Southwestern College Publishing.

[7] K. Brenan, S. Campbell, and L. Petzold (1996) *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, 2^{nd} ed., SIAM Pub.

[8] S. Brenner and L. Scott (1994) *The Mathematical Theory of Finite Element Methods*, Springer-Verlag.

[9] H. Brunner and P. van der Houwen (1986) *The Numerical Solution of Volterra Equations*, North-Holland Pub.

[10] Jean-Luc Chabert, et al. (1999) *A History of Algorithms*: *From the Pebble to the Microchip*, Springer-Verlag.

[11] W. Cowell, editor (1984) *Sources and Development of Mathematical Software*, Prentice-Hall Inc.

[12] P. Dierckx (1993) *Curve and Surface Fitting with Splines*, Oxford Univ. Press.

[13] J. Dongarra, I. Duff, D. Sorensen, and H. van der Vorst (1998) *Numerical Linear Algebra for High-Performance Computers*, SIAM Pub.

[14] I. Duff and G. Watson, editors (1997) *The State of the Art in Numerical Analysis*, Oxford Univ. Press.

[15] C. Edwards, Jr. (1997) *The Historical Development of the Calculus*, Springer-Verlag.

[16] L. Fosdick, E. Jessup, C. Schauble, and G. Domik (1996) *An Introduction to High-Performance Scientific Computing*, MIT Press.
[17] H. Goldstine (1977) *A History of Numerical Analysis*: *From the 16^{th} Through the 19^{th} Century*, Springer-Verlag.
[18] G. Golub and C. Van Loan (1996) *Matrix Computations*, 3^{rd} ed., John Hopkins Univ. Press.
[19] E. Haug (1989) *Computer Aided Kinematics and Dynamics of Mechanical Systems, Vol. I: Basic Methods*, Allyn and Bacon, Pub.
[20] M. Heath (1997) *Scientific Computing*: *An Introductory Survey*, McGraw-Hill Inc.
[21] N. Higham (1996) *Accuracy and Stability of Numerical Algorithms*, SIAM Pub.
[22] A. Iserles (1996) *A First Course in the Numerical Analysis of Differential Equations*, Cambridge Univ. Press.
[23] D. Jacob (1999) *Introduction to Atmospheric Chemistry*, Princeton Univ. Press.
[24] C. T. Kelley (1995) *Iterative Methods for Linear and Nonlinear Equations*, SIAM Pub.
[25] A. Krommer and C. Ueberhuber (1998) *Computational Integration*, SIAM Pub.
[26] C. Lin and L. Segal (1974) *Mathematics Applied to Deterministic Problems in the Natural Sciences*, MacMillan Pub (and reprinted in 1988 by SIAM Pub.).
[27] J. Nocedal and S. Wright (1999) *Numerical Optimization*, Springer-Verlag.
[28] M. Overton (2001) *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM Pub.
[29] C. Ragsdale (2001) *Spreadsheet Modeling and Decision Analysis*, Southwestern College Publishing.
[30] J. Thomas (1995) *Numerical partial differential equations*: *finite difference methods*, Springer-Verlag.
[31] J. Thomas (1999) *Numerical partial differential equations*: *Conservation laws and elliptic equations*, Springer-Verlag.