COMPUTATIONAL PHYSICS

Steven E. Koonin

53.31
K82

COMPUTATIONAL PHYSICS

Steven E. Koonin
Professor of Theoretical Physics
California Institute of Technology

2ℓc

The Benjamin/Cummings Publishing Company, Inc.
Menlo Park, California • Reading, Massachusetts
Don Mills, Ontario • Wokingham, U.K. • Amsterdam • Sydney
Singapore • Tokyo • Mexico City • Bogota • Santiago • San Juan

Sponsoring Editor: Richard W. Mixter
Production Editor: Karen Gulliver
Cover Designer: Victoria Ann Philp

Library of Congress Cataloging-in-Publication Data

Koonin, Steven E.
  Computational physics.

  Bibliography: p.
  1. Mathematical physics. 2. Numerical analysis.
3. Physics--Computer programs. 4. Differential
equations--Numerical solutions. I. Title.
QC20.K665 1986 530.1'5'0285 85-15052
ISBN 0-8053-5430-1
ABCDEFGHIJKL-HA-89876

The Benjamin/Cummings Publishing Company, Inc.
2727 Sand Hill Road
Menlo Park, CA 94025

This book was typeset by the author using the UNIX-based TROFF text-processing system running on a Digital Equipment Corporation VAX-11/750 computer. Camera-ready output from a Versatec [illegible]-80 printer-plotter was reduced 15% to produce the final copy.

Preface

Computation is an integral part of modern science and the ability to exploit effectively the power offered by computers is therefore essential to a working physicist. The proper application of a computer to modeling physical systems is far more than blind "number crunching", and the successful computational physicist draws on a balanced mix of analytically soluble examples, physical intuition, and numerical work to solve problems which are otherwise intractable.

Unfortunately, the ability "to compute" is seldom cultivated by the standard university-level physics curriculum, as it requires an integration of three disciplines (physics, numerical analysis, and computer programming) covered in disjoint courses. Few physics students finish their undergraduate education knowing how to compute; those that do usually learn a limited set of techniques in the course of independent work, such as a research project or a senior thesis.

The material in this book is aimed at refining computational skills in advanced undergraduate or beginning graduate students by providing direct experience in using a computer to model physical systems. Its scope includes the minimum set of numerical techniques needed to "do physics" on a computer. Each of these is developed in the text, often heuristically, and is then applied to solve non-trivial problems in classical, quantum, and statistical physics. These latter have been chosen to enrich or extend the standard undergraduate physics curriculum, and so have considerable intrinsic interest, quite independent of the computational principles they illustrate.

This book should not be thought of as setting out a rigid or definitive curriculum. I have restricted its scope to calculations which satisfy simultaneously the criteria of illustrating a widely applicable numerical technique, of being tractable on a microcomputer, and of having some particular physics interest. Several important numerical techniques have therefore been omitted, spline interpolation and the Fast Fourier Transform among them. *Computational Physics* is perhaps best thought of as establishing an environment offering opportunities for further exploration. There are many possible extensions and embellishments of the material presented; using one's imagination along these lines is one of the more rewarding parts of working through the book.

Computational Physics is primarily a physics text. For maximum benefit, the student should have taken, or be taking, undergraduate courses in classical mechanics, quantum mechanics, statistical mechanics, and advanced calculus or the mathematical methods of physics. This is *not* a text on numerical analysis, as there has been no attempt at rigor or completeness in any of the expositions of numerical techniques. However, a prior course in that subject is probably not essential; the discussions of numerical techniques should be accessible to a student with the physics background outlined above, perhaps with some reference to any one of the excellent texts on numerical analysis (for example, [Ac70], [Bu81], or [Sh84]). This is also *not* a text on computer programming. Although I have tried to follow the principles of good programming throughout (see Appendix B), there has been no attempt to teach programming *per se*. Indeed, techniques for organizing and writing code are somewhat peripheral to the main goals of the book. Some familiarity with programming, at least to the extent of a one-semester introductory course in any of the standard high-level languages (BASIC, FORTRAN, PASCAL, C), is therefore essential.

The choice of language invariably invokes strong feelings among scientists who use computers. Any language is, after all, only a means of expressing the concepts underlying a program. The contents of this book are therefore relevant no matter what language one works in. However, *some* language had to be chosen to implement the programs, and I have selected the Microsoft dialect of BASIC standard on the IBM PC/XT/AT computers for this purpose. The BASIC language has many well-known deficiencies, foremost among them being a lack of local subroutine variables and an awkwardness in expressing structured code. Nevertheless, I believe that these are more than balanced by the simplicity of the language and the widespread fluency in it, BASIC's almost universal availability on the microcomputers most likely to be used with this book, the existence of both BASIC interpreters convenient for writing and debugging programs and of compilers for producing rapidly executing finished programs, and the powerful graphics and I/O statements in this language. I expect that readers familiar with some other high-level language can learn enough BASIC "on the fly" to be able to use this book. A synopsis of the language is contained in Appendix A to help in this regard, and further information can be found in readily available manuals. The reader may, of course, elect to write the programs suggested in the text in any convenient language.

This book arose out of the Advanced Computational Physics Laboratory taught to third- and fourth-year undergraduate Physics majors at Caltech during the Winter and Spring of 1984. The content and presentation have benefitted greatly from the many inspired suggestions of M.-C. Chu, V. Pönisch, R. Williams, and D.

Meredith. Mrs. Meredith was also of great assistance in producing the final form of the manuscript and programs. I also wish to thank my wife, Laurie, for her extraordinary patience, understanding, and support during my two-year involvement in this project.

Steven E. Koonin
Pasadena
May, 1985

How to use this book

This book is organized into chapters, each containing a text section, an example, and a project. Each text section is a brief discussion of one or several related numerical techniques, often illustrated with simple mathematical examples. Throughout the text are a number of exercises, in which the student's understanding of the material is solidified or extended by an analytical derivation or through the writing and running of a simple program. These exercises are indicated by the symbol □ □ □ in the outer margin.

The example and project in each chapter are applications of the numerical techniques to particular physical problems. Each includes a brief exposition of the physics, followed by a discussion of how the numerical techniques are to be applied. The examples and projects differ only in that the student is expected to use (and perhaps modify) the program which is given for the former in Appendix B, while the book provides guidance in writing programs to treat the latter through a series of steps, also indicated by the symbol □ □ □ in the outer margin. However, programs for the projects have also been included in Appendix C; these can serve as models for the student's own program or as a means of investigating the physics without having to write a major program "from scratch". A number of suggested studies accompany each example and project; these guide the student in exploiting the programs and understanding the physical principles and numerical techniques involved.

The diskette included with this book (360 kB, double-sided, double-density format) also contains the BASIC source codes for the examples and projects; it is suitable for use on any microcomputer system operating under MS-DOS Version 2.0 or higher. Further information about these programs can be found at the beginning of Appendix B and in the file README on the diskette, which can be read by inserting the diskette into the default disk drive and entering the DOS command "TYPE README". Note that it is wise to back up this write-protected diskette before beginning to use the programs.

An attempt has been made to use only the most primitive BASIC statements, so that the codes for the projects and examples should be appropriate for most BASIC dialects. All of the programs will run under a BASIC interpreter, but most require enough computation to make execution speed an important consideration. For serious study, it is therefore recommended that the codes be compiled through the IBM or Microsoft BASIC compiler, after which they will run between five and ten times faster. The programs have also been written in such as way as to make relatively straightforward their transcription into another high-level language, such as FORTRAN.

A "laboratory" format has proved to be one effective mode of presenting this material in a university setting. Students are quite able to work through the text on their own, with the instructor being available for consultation and to monitor progress through brief personal interviews on each chapter. Three chapters in ten weeks (60 hours) of instruction has proved to be a reasonable pace, with students typically writing two of the projects during this time, and using the "canned" codes to work through the physics of the remaining project and the examples. The eight chapters in this book should therefore be more than sufficient for a one-semester course. Alternatively, this book can be used to provide supplementary material for the usual courses in classical, quantum, and statistical mechanics. Many of the examples and projects are vivid illustrations of basic concepts in these subjects and are therefore suitable for classroom demonstrations or independent study.

This book should not be thought of as setting out a rigid or definitive curriculum. I have restricted its scope to calculations which satisfy simultaneously the criteria of illustrating a widely applicable numerical technique, of being tractable on a microcomputer, and of having some particular physics interest. Several important numerical techniques have therefore been omitted, spline interpolation and the Fast Fourier Transform among them. *Computational Physics* is perhaps best thought of as establishing an environment offering opportunities for further exploration. There are many possible extensions and embellishments of the material presented; using one's imagination along these lines is one of the more rewarding parts of working through the book.

Contents

The problem with computers is that they only give answers
-attributed to P. Picasso

Chapter 1

Basic Mathematical Operations

Three numerical operations - differentiation, quadrature, and the finding of roots - are central to most computer modeling of physical systems. Suppose that we have the ability to calculate the value of a function, $f(x)$, at any value of the independent variable x. In differentiation, we seek one of the derivatives of f at a given value of x. Quadrature, roughly the inverse of differentiation, requires us to calculate the definite integral of f between two specified limits (we reserve the term "integration" for the process of solving ordinary differential equations, as discussed in Chapter 2), while in root finding we seek the values of x (there may be several) at which f vanishes.

If f is known analytically, it is almost always possible, with enough fortitude, to derive explicit formulas for the derivatives of f, and it is often possible to do so for its definite integral as well. However, it is often the case that an analytical method cannot be used, even though we can evaluate $f(x)$ itself. This might be either because some very complicated numerical procedure is required to evaluate f and we have no suitable analytical formula upon which to apply the rules of differentiation and quadrature, or, even worse, because the way we can generate f provides us with its values at only a set of discrete abscissae. In these situations, we must employ approximate formulas expressing the derivatives and integral in terms of the values of f we can compute. Moreover, the roots of all but the simplest functions cannot be found analytically, and numerical methods are therefore essential.

This chapter deals with the computer realization of these three basic operations. The central technique is to approximate f by a simple function (such as first- or second-degree polynomial) upon which these operations can be performed easily. We will derive only the simplest and most commonly used formulas; fuller treatments can be found in many textbooks on numerical analysis.

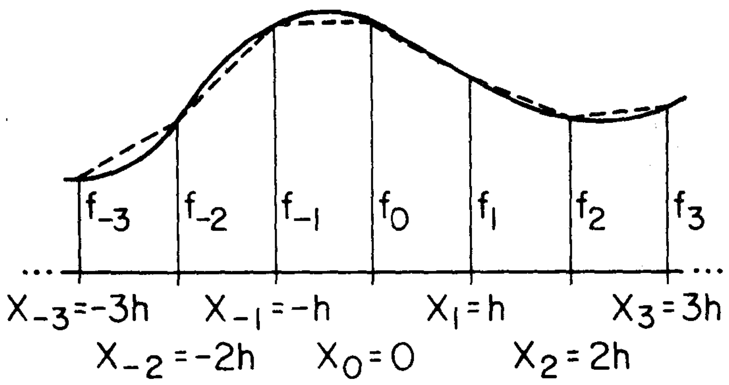

Figure 1.1 Values of f on an equally-spaced lattice. Dashed lines show the linear interpolation.

1.1 Numerical differentiation

Let us suppose that we are interested in the derivative at $x=0$, $f'(0)$. (The formulas we will derive can be generalized simply to arbitrary x by translation.) Let us also suppose that we know f on an equally-spaced lattice of x values,

$$f_n=f(x_n);\ x_n=nh\ \ (n=0,\pm1,\pm2,\cdots),$$

and that our goal is to compute an approximate value of $f'(0)$ in terms of the f_n (see Figure 1.1).

We begin by using a Taylor series to expand f in the neighborhood of $x=0$:

$$f(x)=f_0+xf'+\frac{x^2}{2!}f''+\frac{x^3}{3!}f'''+\cdots, \qquad (1.1)$$

where all derivatives are evaluated at $x=0$. It is then simple to verify that

$$f_{\pm1}\equiv f(x=\pm h)=f_0\pm hf'+\frac{h^2}{2}f''\pm\frac{h^3}{6}f'''+O(h^4), \qquad (1.2a)$$

$$f_{\pm2}\equiv f(x=\pm2h)=f_0\pm2hf'+2h^2f''\pm\frac{4h^3}{3}f'''+O(h^4), \qquad (1.2b)$$

where $O(h^4)$ means terms of order h^4 or higher. To estimate the size of such terms, we can assume that f and its derivatives are all of the same order of magnitude, as is the case for many functions of physical relevance.

Upon subtracting f_{-1} from f_1 as given by (1.2a), we find, after a slight rearrangement,

$$f'=\frac{f_1-f_{-1}}{2h}-\frac{h^2}{6}f'''+O(h^4). \qquad (1.3a)$$

The term involving f''' vanishes as h becomes small and is the dominant error associated with the finite difference approximation that retains only the first term:

$$f' \approx \frac{f_1 - f_{-1}}{2h}. \tag{1.3b}$$

This "3-point" formula would be exact if f were a second-degree polynomial in the 3-point interval $[-h, +h]$, because the third- and all higher-order derivatives would then vanish. Hence, the essence of Eq. (1.3b) is the assumption that a quadratic polynomial interpolation of f through the three points $x=\pm h, 0$ is valid.

Equation (1.3b) is a very natural result, reminiscent of the formulas used to define the derivative in elementary calculus. The error term (of order h^2) can, in principle, be made as small as is desired by using smaller and smaller values of h. Note also that the symmetric difference about $x=0$ is used, as it is more accurate (by one order in h) than the forward or backward difference formulas:

$$f' \approx \frac{f_1 - f_0}{h} + O(h); \tag{1.4a}$$

$$f' \approx \frac{f_0 - f_{-1}}{h} + O(h). \tag{1.4b}$$

These "2-point" formulas are based on the assumption that f is well approximated by a linear function over the intervals between $x=0$ and $x=\pm h$.

As a concrete example, consider evaluating $f'(x=1)$ when $f(x)=\sin x$. The exact answer is, of course, $\cos 1 = 0.540302$. The following BASIC program evaluates Eq. (1.3b) in this case for the value of h input:

```
10 X=1: EXACT=COS(X)
20 INPUT "enter value of h (<=0 to stop)";H
30 IF H<=0 THEN STOP
40 FPRIME = (SIN(X+H)-SIN(X-H))/(2*H)
50 DIFF=EXACT-FPRIME
60 PRINT USING "h=#.######, ERROR=+#.######";H,DIFF
70 GOTO 20
```

(If you are a beginner in BASIC, note the way the value of H is requested from the keyboard, the fact that the code will stop if a non-positive value of H is entered, the natural way in which variable names are chosen and the mathematical formula (1.3b) is transcribed using the SIN function in line 40, the way in which the number of significant digits is specified when the result is to be output to the screen in line 60, and the jump in program control in

Table 1.1 Error in evaluating $d\sin x/dx\,|_{x=1}=0.540302$

h	Symmetric 3-point Eq. (1.3b)	Forward 2-point Eq. (1.4a)	Backward 2-point Eq. (1.4b)	Symmetric 5-point Eq. (1.5)
0.50000	0.022233	0.228254	-0.183789	0.001092
0.20000	0.003595	0.087461	-0.080272	0.000028
0.10000	0.000899	0.042938	-0.041139	0.000001
0.05000	0.000225	0.021258	-0.020808	0.000000
0.02000	0.000037	0.008453	-0.008380	0.000001
0.01000	0.000010	0.004224	-0.004204	0.000002
0.00500	0.000010	0.002108	-0.002088	0.000006
0.00200	-0.000014	0.000820	-0.000848	-0.000017
0.00100	-0.000014	0.000403	-0.000431	-0.000019
0.00050	0.000105	0.000403	-0.000193	0.000115
0.00020	-0.000163	-0.000014	-0.000312	-0.000188
0.00010	-0.000312	-0.000312	-0.000312	-0.000411
0.00005	0.000284	0.001476	-0.000908	0.000681
0.00002	0.000880	0.000880	0.000880	0.000873
0.00001	0.000880	0.003860	-0.002100	0.000880

line 70.)

Results generated with this program, as well as with similar ones evaluating the forward and backward difference formulas Eqs. (1.4a,b), are shown in Table 1.1. Note that the result improves as we decrease h, but only up to a point, after which it becomes worse. This is because arithmetic in the computer is performed with only a limited precision (5-6 decimal digits for a single precision BASIC variable), so that when the difference in the numerator of the approximations is formed, it is subject to large "round-off" errors if h is small and f_1 and f_{-1} differ very little. For example, if $h=10^{-6}$, then

$$f_1=\sin(1.000001)=0.841472;\ f_{-1}=\sin(0.999999)=0.841470,$$

so that $f_1-f_{-1}=0.000002$ to six significant digits. When substituted into (1.3b) we find $f'\approx 1.000000$, a very poor result. However, if we do the arithmetic with 10 significant digits, then

$$f_1=0.8414715251;\ f_{-1}=0.8414704445,$$

which gives a respectable $f'\approx 0.540300$ in Eq. (1.3b). In this sense, numerical differentiation is an intrinsically unstable process (no well-defined limit as $h\to 0$), and so must be carried out with caution.

It is possible to improve on the 3-point formula (1.3b) by relating f' to lattice points further removed from $x=0$. For example, using Eqs. (1.2), it is easy to show that the "5-point" formula

$$f' \approx \frac{1}{12h}[f_{-2}-8f_{-1}+8f_1-f_2]+O(h^4) \tag{1.5}$$

cancels all derivatives in the Taylor series through fourth order. Computing the derivative in this way assumes that f is well-approximated by a fourth-degree polynomial over the 5-point interval $[-2h, 2h]$. Although requiring more computation, this approximation is considerably more accurate, as can be seen from Table 1.1. In fact, an accuracy comparable to Eq. (1.3b) is obtained with a step some 10 times larger. This can be an important consideration when many values of f must be stored in the computer, as the greater accuracy allows a sparser tabulation and so saves storage space. However, because (1.5) requires more mathematical operations than does (1.3b) and there is considerable cancellation among the various terms (they have both positive and negative coefficients), precision problems show up at a larger value of h.

Formulas for higher derivatives can be constructed by taking appropriate combinations of Eqs. (1.2). For example, it is easy to see that

$$f_1-2f_0+f_{-1}=h^2f''+O(h^4), \tag{1.6}$$

so that an approximation to the second derivative accurate to order h^2 is

$$f'' \approx \frac{f_1-2f_0+f_{-1}}{h^2}. \tag{1.7}$$

Difference formulas for the various derivatives of f that are accurate to a higher order in h can be derived straightforwardly. Table 1.2 is a summary of the 4- and 5-point expressions.

Exercise 1.1 Using any function for which you can evaluate the derivatives analytically, investigate the accuracy of the formulas in Table 1.2 for various values of h. □ □ □

1.2 Numerical quadrature

In quadrature, we are interested in calculating the definite integral of f between two limits, $a<b$. We can easily arrange for these values to be points of the lattice separated by an even number of lattice spacings; i.e.,

$$N=\frac{(b-a)}{h}$$