




Xin-She Yang

2nd Edition

# INTRODUCTION TO **Computational Mathematics**

 World Scientific

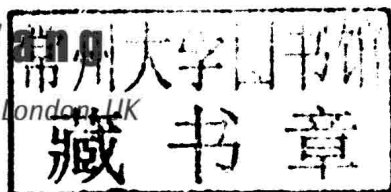


2nd Edition

# INTRODUCTION TO **Computational Mathematics**

**Xin-She Yang**

*Middlesex University London, UK*



 **World Scientific**

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI



*Published by*

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

*USA office:* 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

*UK office:* 57 Shelton Street, Covent Garden, London WC2H 9HE

**Library of Congress Cataloging-in-Publication Data**

Yang, Xin-She.

Introduction to computational mathematics / by Xin-She Yang (Middlesex University London, UK). -- 2nd edition.

pages cm

Includes bibliographical references and index.

ISBN 978-9814635776 (hardcover : alk. paper) -- ISBN 978-9814635783 (pbk. : alk. paper)

1. Numerical analysis. 2. Algorithms. 3. Mathematical analysis--Foundations. 4. Programming (Mathematics) I. Title.

QA297.Y36 2015

518--dc23

2014038711

**British Library Cataloguing-in-Publication Data**

A catalogue record for this book is available from the British Library.

Copyright © 2015 by World Scientific Publishing Co. Pte. Ltd.

*All rights reserved. This book, or parts thereof, may not be reproduced in any form or by any means, electronic or mechanical, including photocopying, recording or any information storage and retrieval system now known or to be invented, without written permission from the publisher.*

For photocopying of material in this volume, please pay a copying fee through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA. In this case permission to photocopy is not required from the publisher.

Printed in Singapore by B & Jo Enterprise Pte Ltd



# Preface

Computational mathematics is essentially the foundation of modern scientific computing. Traditional ways of doing sciences consist of two major paradigms: by theory and by experiment. With the steady increase in computer power, there emerges a third paradigm of doing sciences: by computer simulation. Numerical algorithms are the very essence of any computer simulation, and computational mathematics is just the science of developing and analyzing numerical algorithms.

The science that studies numerical algorithms is numerical analysis or more broadly computational mathematics. Loosely speaking, numerical algorithms and analysis should include four categories of algorithms: numerical linear algebra, numerical optimization, numerical solutions of differential equations (ODEs and PDEs) and stochastic data modelling.

Many numerical algorithms were developed well before the computer was invented. For example, Newton's method for finding roots of nonlinear equations was developed in 1669, and Gauss quadrature for numerical integration was formulated in 1814. However, their true power and efficiency have been demonstrated again and again in modern scientific computing. Since the invention of the modern computer in the 1940s, many numerical algorithms have been developed since the 1950s. As the speed of computers increases, together with the increase in the efficiency of numerical algorithms, a diverse range of complex and challenging problems in mathematics, science and engineering can nowadays be solved numerically to very high accuracy. Numerical algorithms have become more important than ever.

The topics of computational mathematics are broad and the related literature is vast. It is often a daunting task for beginners to find the right book(s) and to learn the right algorithms that are widely used in



computational mathematics. Even for lecturers and educators, it is no trivial task to decide what algorithms to teach and to provide balanced coverage of a wide range of topics, because there are so many algorithms to choose from.

The first edition of this book was published by World Scientific Publishing in 2008 and it was well received. Many universities courses used it as a main reference. Constructive feedbacks and helpful comments have also been received from the readers. This second edition has incorporated all these comments and consequently includes more algorithms and new algorithms to reflect the state-of-the-art developments such as computational intelligence and swarm intelligence.

Therefore, this new edition strives to provide extensive coverage of efficient algorithms commonly used in computational mathematics and modern scientific computing. It covers all the major topics including root-finding algorithms, numerical integration, interpolation, linear algebra, eigenvalues, numerical methods of ordinary differential equations (ODEs) and partial differential equations (PDEs), finite difference methods, finite element methods, finite volume methods, algorithm complexity, optimization, mathematical programming, stochastic models such as least squares and regression, machine learning such as neural networks and support vector machine, computational intelligence and swarm intelligence such as cuckoo search, bat algorithm, firefly algorithm as well as particle swarm optimization.

The book covers both traditional methods and new algorithms with dozens of worked examples to demonstrate how these algorithms work. Thus, this book can be used as a textbook and/or reference book, especially suitable for undergraduates and graduates in computational mathematics, engineering, computer science, computational intelligence, data science and scientific computing.

Xin-She Yang

London, 2014



# Contents

<i>Preface</i>	v
<b>I Mathematical Foundations</b>	<b>1</b>
1. Mathematical Foundations	3
1.1 The Essence of an Algorithm . . . . .	3
1.2 Big- $O$ Notations . . . . .	5
1.3 Differentiation and Integration . . . . .	6
1.4 Vector and Vector Calculus . . . . .	10
1.5 Matrices and Matrix Decomposition . . . . .	15
1.6 Determinant and Inverse . . . . .	20
1.7 Matrix Exponential . . . . .	24
1.8 Hermitian and Quadratic Forms . . . . .	26
1.9 Eigenvalues and Eigenvectors . . . . .	28
1.10 Definiteness of Matrices . . . . .	31
2. Algorithmic Complexity, Norms and Convexity	33
2.1 Computational Complexity . . . . .	33
2.2 NP-Complete Problems . . . . .	34
2.3 Vector and Matrix Norms . . . . .	35
2.4 Distribution of Eigenvalues . . . . .	37
2.5 Spectral Radius of Matrices . . . . .	44
2.6 Hessian Matrix . . . . .	47
2.7 Convexity . . . . .	48



3.	Ordinary Differential Equations	51
3.1	Ordinary Differential Equations . . . . .	51
3.2	First-Order ODEs . . . . .	52
3.3	Higher-Order ODEs . . . . .	53
3.4	Linear System . . . . .	56
3.5	Sturm-Liouville Equation . . . . .	58
4.	Partial Differential Equations	59
4.1	Partial Differential Equations . . . . .	59
4.1.1	First-Order Partial Differential Equation . . . . .	60
4.1.2	Classification of Second-Order Equations . . . . .	61
4.2	Mathematical Models . . . . .	61
4.2.1	Parabolic Equation . . . . .	61
4.2.2	Poisson's Equation . . . . .	61
4.2.3	Wave Equation . . . . .	62
4.3	Solution Techniques . . . . .	64
4.3.1	Separation of Variables . . . . .	65
4.3.2	Laplace Transform . . . . .	67
4.3.3	Similarity Solution . . . . .	68
 <b>II Numerical Algorithms</b>		 <b>71</b>
5.	Roots of Nonlinear Equations	73
5.1	Bisection Method . . . . .	73
5.2	Simple Iterations . . . . .	75
5.3	Newton's Method . . . . .	76
5.4	Iteration Methods . . . . .	78
5.5	Numerical Oscillations and Chaos . . . . .	81
6.	Numerical Integration	85
6.1	Trapezium Rule . . . . .	86
6.2	Simpson's Rule . . . . .	87
6.3	Gaussian Integration . . . . .	89
7.	Computational Linear Algebra	95
7.1	System of Linear Equations . . . . .	95
7.2	Gauss Elimination . . . . .	97



7.3	LU Factorization . . . . .	101
7.4	Iteration Methods . . . . .	103
7.4.1	Jacobi Iteration Method . . . . .	103
7.4.2	Gauss-Seidel Iteration . . . . .	107
7.4.3	Relaxation Method . . . . .	108
7.5	Newton-Raphson Method . . . . .	109
7.6	QR Decomposition . . . . .	110
7.7	Conjugate Gradient Method . . . . .	115
8.	Interpolation . . . . .	117
8.1	Spline Interpolation . . . . .	117
8.1.1	Linear Spline Functions . . . . .	117
8.1.2	Cubic Spline Functions . . . . .	118
8.2	Lagrange Interpolating Polynomials . . . . .	123
8.3	Bézier Curve . . . . .	125
<b>III</b>	<b>Numerical Methods of PDEs</b>	<b>127</b>
9.	Finite Difference Methods for ODEs . . . . .	129
9.1	Integration of ODEs . . . . .	129
9.2	Euler Scheme . . . . .	130
9.3	Leap-Frog Method . . . . .	131
9.4	Runge-Kutta Method . . . . .	132
9.5	Shooting Methods . . . . .	134
10.	Finite Difference Methods for PDEs . . . . .	139
10.1	Hyperbolic Equations . . . . .	139
10.2	Parabolic Equation . . . . .	142
10.3	Elliptical Equation . . . . .	143
10.4	Spectral Methods . . . . .	146
10.5	Pattern Formation . . . . .	148
10.6	Cellular Automata . . . . .	150
11.	Finite Volume Method . . . . .	153
11.1	Concept of the Finite Volume . . . . .	153
11.2	Elliptic Equations . . . . .	154
11.3	Parabolic Equations . . . . .	155
11.4	Hyperbolic Equations . . . . .	156



12. Finite Element Method	157
12.1 Finite Element Formulation	157
12.1.1 Weak Formulation	157
12.1.2 Galerkin Method	158
12.1.3 Shape Functions	159
12.2 Derivatives and Integration	163
12.2.1 Derivatives	163
12.2.2 Gauss Quadrature	164
12.3 Poisson's Equation	165
12.4 Transient Problems	169
 <b>IV Mathematical Programming</b>	 <b>171</b>
13. Mathematical Optimization	173
13.1 Optimization	173
13.2 Optimality Criteria	175
13.3 Unconstrained Optimization	177
13.3.1 Univariate Functions	177
13.3.2 Multivariate Functions	178
13.4 Gradient-Based Methods	180
13.4.1 Newton's Method	181
13.4.2 Steepest Descent Method	182
14. Mathematical Programming	187
14.1 Linear Programming	187
14.2 Simplex Method	189
14.2.1 Basic Procedure	189
14.2.2 Augmented Form	191
14.2.3 A Case Study	192
14.3 Nonlinear Programming	196
14.4 Penalty Method	196
14.5 Lagrange Multipliers	197
14.6 Karush-Kuhn-Tucker Conditions	199
14.7 Sequential Quadratic Programming	200
14.7.1 Quadratic Programming	200
14.7.2 Sequential Quadratic Programming	200
14.8 No Free Lunch Theorems	202



<b>V</b>	<b>Stochastic Methods and Data Modelling</b>	<b>205</b>
15.	Stochastic Models	207
15.1	Random Variables . . . . .	207
15.2	Binomial and Poisson Distributions . . . . .	209
15.3	Gaussian Distribution . . . . .	211
15.4	Other Distributions . . . . .	213
15.5	The Central Limit Theorem . . . . .	215
15.6	Weibull Distribution . . . . .	216
16.	Data Modelling	221
16.1	Sample Mean and Variance . . . . .	221
16.2	Method of Least Squares . . . . .	223
16.2.1	Maximum Likelihood . . . . .	223
16.2.2	Linear Regression . . . . .	223
16.3	Correlation Coefficient . . . . .	226
16.4	Linearization . . . . .	227
16.5	Generalized Linear Regression . . . . .	229
16.6	Nonlinear Regression . . . . .	233
16.7	Hypothesis Testing . . . . .	237
16.7.1	Confidence Interval . . . . .	237
16.7.2	Student's $t$ -Distribution . . . . .	238
16.7.3	Student's $t$ -Test . . . . .	240
17.	Data Mining, Neural Networks and Support Vector Machine	243
17.1	Clustering Methods . . . . .	243
17.1.1	Hierarchy Clustering . . . . .	243
17.1.2	$k$ -Means Clustering Method . . . . .	244
17.2	Artificial Neural Networks . . . . .	247
17.2.1	Artificial Neuron . . . . .	247
17.2.2	Artificial Neural Networks . . . . .	248
17.2.3	Back Propagation Algorithm . . . . .	250
17.3	Support Vector Machine . . . . .	251
17.3.1	Classifications . . . . .	251
17.3.2	Statistical Learning Theory . . . . .	252
17.3.3	Linear Support Vector Machine . . . . .	253
17.3.4	Kernel Functions and Nonlinear SVM . . . . .	256



18. Random Number Generators and Monte Carlo Method	259
18.1 Linear Congruential Algorithms . . . . .	259
18.2 Uniform Distribution . . . . .	260
18.3 Generation of Other Distributions . . . . .	262
18.4 Metropolis Algorithms . . . . .	266
18.5 Monte Carlo Methods . . . . .	267
18.6 Monte Carlo Integration . . . . .	270
18.7 Importance of Sampling . . . . .	273
18.8 Quasi-Monte Carlo Methods . . . . .	275
18.9 Quasi-Random Numbers . . . . .	276
 <b>VI Computational Intelligence</b>	 <b>279</b>
19. Evolutionary Computation	281
19.1 Introduction to Evolutionary Computation . . . . .	281
19.2 Simulated Annealing . . . . .	282
19.3 Genetic Algorithms . . . . .	286
19.3.1 Basic Procedure . . . . .	287
19.3.2 Choice of Parameters . . . . .	289
19.4 Differential Evolution . . . . .	291
20. Swarm Intelligence	295
20.1 Introduction to Swarm Intelligence . . . . .	295
20.2 Ant and Bee Algorithms . . . . .	296
20.3 Particle Swarm Optimization . . . . .	297
20.4 Accelerated PSO . . . . .	299
20.5 Binary PSO . . . . .	301
21. Swarm Intelligence: New Algorithms	303
21.1 Firefly Algorithm . . . . .	303
21.2 Cuckoo Search . . . . .	306
21.3 Bat Algorithm . . . . .	310
21.4 Flower Algorithm . . . . .	313
21.5 Other Algorithms . . . . .	317
 <i>Bibliography</i>	 319
<i>Index</i>	325



# Part I

## Mathematical Foundations







## Chapter 1

# Mathematical Foundations

Computational mathematics concerns a wide range of topics, from basic root-finding algorithms and linear algebra to advanced numerical methods for partial differential equations and nonlinear mathematical programming. In order to introduce various algorithms, we first review some mathematical foundations briefly.

### 1.1 The Essence of an Algorithm

Let us start by asking: what is an algorithm? In essence, an algorithm is a step-by-step procedure of providing calculations or instructions. Many algorithms are iterative. The actual steps and procedures will depend on the algorithm used and the context of interest. However, in this book, we place more emphasis on iterative procedures and ways for constructing algorithms.

For example, a simple algorithm of finding the square root of any positive number  $k > 0$ , or  $x = \sqrt{k}$ , can be written as

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{k}{x_n}\right), \quad (1.1)$$

starting from a guess solution  $x_0 \neq 0$ , say,  $x_0 = 1$ . Here,  $n$  is the iteration counter or index, also called the pseudo-time or generation counter. The above iterative equation comes from the re-arrangement of  $x^2 = k$  in the following form

$$\frac{x}{2} = \frac{k}{2x}, \quad (1.2)$$

which can be rewritten as

$$x = \frac{1}{2}\left(x + \frac{k}{x}\right). \quad (1.3)$$



For example, for  $k = 7$  with  $x_0 = 1$ , we have

$$x_1 = \frac{1}{2}(x_0 + \frac{7}{x_0}) = \frac{1}{2}(1 + \frac{7}{1}) = 4. \quad (1.4)$$

$$x_2 = \frac{1}{2}(x_1 + \frac{7}{x_1}) = 2.875, \quad x_3 \approx 2.654891304, \quad (1.5)$$

$$x_4 \approx 2.645767044, \quad x_5 \approx 2.6457513111. \quad (1.6)$$

We can see that  $x_5$  after just 5 iterations (or generations) is very close to the true value of  $\sqrt{7} = 2.64575131106459\dots$ , which shows that this iteration method is very efficient.

The reason that this iterative process works is that the series  $x_1, x_2, \dots, x_n$  converges to the true value  $\sqrt{k}$  due to the fact that

$$\frac{x_{n+1}}{x_n} = \frac{1}{2}(1 + \frac{k}{x_n^2}) \rightarrow 1, \quad x_n \rightarrow \sqrt{k}, \quad (1.7)$$

as  $n \rightarrow \infty$ . However, a good choice of the initial value  $x_0$  will speed up the convergence. A wrong choice of  $x_0$  could make the iteration fail; for example, we cannot use  $x_0 = 0$  as the initial guess, and we cannot use  $x_0 < 0$  either as  $\sqrt{k} > 0$  (in this case, the iterations will approach another root  $-\sqrt{k}$ ). So a sensible choice should be an educated guess. At the initial step, if  $x_0^2 < k$ ,  $x_0$  is the lower bound and  $k/x_0$  is upper bound. If  $x_0^2 > k$ , then  $x_0$  is the upper bound and  $k/x_0$  is the lower bound. For other iterations, the new bounds will be  $x_n$  and  $k/x_n$ . In fact, the value  $x_{n+1}$  is always between these two bounds  $x_n$  and  $k/x_n$ , and the new estimate  $x_{n+1}$  is thus the mean or average of the two bounds. This guarantees that the series converges to the true value of  $\sqrt{k}$ . This method is similar to the well-known bisection method.

You may have already wondered why  $x^2 = k$  was converted to Eq. (1.1)? Why do not we write it as the following iterative formula:

$$x_n = \frac{k}{x_n}, \quad (1.8)$$

starting from  $x_0 = 1$ ? With this and  $k = 7$ , we have

$$x_1 = \frac{7}{x_0} = 7, \quad x_2 = \frac{7}{x_1} = 1, \quad x_3 = 7, \quad x_4 = 1, \quad x_5 = 7, \quad \dots, \quad (1.9)$$

which leads to an oscillating feature at two distinct stages 1 and 7. You may wonder that it may be the problem of initial value  $x_0$ . In fact, for any initial value  $x_0 \neq 0$ , this above formula will lead to the oscillations between two values:  $x_0$  and  $k$ . This clearly demonstrates that the way to design a good iterative formula is very important.



Mathematically speaking, an algorithm  $A$  is a procedure to generate a new and better solution  $x_{n+1}$  to a given problem from the current solution  $x_n$  at iteration or time  $t$ . That is,

$$x_{n+1} = A(x_n), \quad (1.10)$$

where  $A$  is a mathematical function of  $x_n$ . In fact,  $A$  can be a set of mathematical equations in general. In some literature, especially those in numerical analysis,  $n$  is often used for the iteration index. In many textbooks, the upper index form  $x^{(n+1)}$  or  $x^{n+1}$  is commonly used. Here,  $x^{n+1}$  does not mean  $x$  to the power of  $n+1$ . Such notations will become useful and no confusion will occur when used appropriately. We will use such notations when appropriate in this book.

## 1.2 Big- $O$ Notations

In analyzing the complexity of an algorithm, we usually estimate the order of computational efforts in terms of its problem size. This often requires the order notations, often in terms of big  $O$  and small  $o$ .

Loosely speaking, for two functions  $f(x)$  and  $g(x)$ , if

$$\lim_{x \rightarrow x_0} \frac{f(x)}{g(x)} \rightarrow K, \quad (1.11)$$

where  $K$  is a finite, non-zero limit, we write

$$f = O(g). \quad (1.12)$$

The big  $O$  notation means that  $f$  is asymptotically equivalent to the order of  $g(x)$ . If the limit is unity or  $K = 1$ , we say  $f(x)$  is order of  $g(x)$ . In this special case, we write

$$f \sim g, \quad (1.13)$$

which is equivalent to  $f/g \rightarrow 1$  and  $g/f \rightarrow 1$  as  $x \rightarrow x_0$ . Obviously,  $x_0$  can be any value, including 0 and  $\infty$ . The notation  $\sim$  does not necessarily mean  $\approx$  in general, though it may give the same results, especially in the case when  $x \rightarrow 0$ . For example,  $\sin x \sim x$  and  $\sin x \approx x$  if  $x \rightarrow 0$ .

When we say  $f$  is order of 100 (or  $f \sim 100$ ), this does not mean  $f \approx 100$ , but it can mean that  $f$  could be between about 50 and 150. The small  $o$  notation is often used if the limit tends to 0. That is

$$\lim_{x \rightarrow x_0} \frac{f}{g} \rightarrow 0, \quad (1.14)$$



or

$$f = o(g). \quad (1.15)$$

If  $g > 0$ ,  $f = o(g)$  is equivalent to  $f \ll g$ . For example, for  $\forall x \in \mathcal{R}$ , we have

$$e^x \approx 1 + x + O(x^2) \approx 1 + x + \frac{x^2}{2} + o(x).$$

---

**Example 1.1:** A classic example is Stirling's asymptotic series for factorials

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + \frac{1}{288n^2} - \frac{139}{51480n^3} - \dots\right),$$

which can demonstrate the fundamental difference between an asymptotic series and the standard approximate expansions. For the standard power expansions, the error  $R_k(h^k) \rightarrow 0$ , but for an asymptotic series, the error of the truncated series  $R_k$  decreases compared with the leading term [here  $\sqrt{2\pi n}(n/e)^n$ ]. However,  $R_n$  does not necessarily tend to zero. In fact,

$$R_2 = \frac{1}{12n} \cdot \sqrt{2\pi n}(n/e)^n,$$

is still very large as  $R_2 \rightarrow \infty$  if  $n \gg 1$ . For example, for  $n = 100$ , we have  $n! = 9.3326 \times 10^{157}$ , while the leading approximation is  $\sqrt{2\pi n}(n/e)^n = 9.3248 \times 10^{157}$ . The difference between these two values is  $7.7740 \times 10^{154}$ , which is still very large, though three orders smaller than the leading approximation.

---

### 1.3 Differentiation and Integration

Differentiation is essentially to find the gradient of a function. For any curve  $y = f(x)$ , we define the gradient as

$$f'(x) \equiv \frac{dy}{dx} \equiv \frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}. \quad (1.16)$$

The gradient is also called the first derivative. The three notations  $f'(x)$ ,  $dy/dx$  and  $df(x)/dx$  are interchangeable. Conventionally, the notation  $dy/dx$  is called Leibnitz's notation, while the prime notation  $'$  is called Lagrange's notation. Newton's dot notation  $\dot{y} = dy/dt$  is now exclusively used for time derivatives. The choice of such notations is purely for clarity, convention and/or personal preference.