

## C++ Toolbox for Verified Computing I

R. Hammer M. Hocks  
U. Kulisch D. Ratz

# **C++ Toolbox for Verified Computing I**

Basic Numerical Problems

Theory, Algorithms, and Programs

With 29 Figures



Springer

Prof. Dr. Ulrich Kulisch  
Dr. Rolf Hammer  
Dr. Matthias Hocks  
Dr. Dietmar Ratz  
Institut für Angewandte Mathematik  
Universität Karlsruhe  
D-76128 Karlsruhe

*Cover figure:* Function of Levy (see also page 337)

Mathematics Subject Classification (1991): 65-01, 65-04, 65F, 65G10, 65H, 65K

ISBN-13: 978-3-642-79653-1

Library of Congress Cataloging-in-Publication Data

C++ toolbox for verified computing: theory, algorithms, and programs

R.Hammer [et al.]. p.cm. - Includes bibliographical references and index. Contents:

v. 1. Basic numerical problems.

ISBN-13: 978-3-642-79653-1 e-ISBN-13: 978-3-642-79651-7

DOI: 10.1007/978-3-642-79651-7

1. C++ (Computer program language) I. Hammer, (Rolf), 1961- .

QA76.73.C153C18 1995 519.4'0285'5133--dc20 95-10173 CIP

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on micro-film or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1995

Softcover reprint of the hardcover 1st edition 1995

The copyright for the computer programs in this publication is owned by the authors.

The use of general descriptive names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

*Disclaimer/Legal Matters:* Springer-Verlag and the authors make no warranties with respect to the adequacy of this book or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. We make no warranties, express or implied, that the programs contained in this volume are free of error, or are consistent with any particular standard of merchantability, or that they will meet your requirements for any particular application. They should not be relied on for solving a problem whose incorrect solution could result in injury to a person or loss of property. If you do use the programs in such a manner, it is at your own risk.

In no event shall Springer-Verlag or the authors be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands or claim for lost profits, fees or expenses of any nature or kind.

Typesetting: Camera ready by authors using T<sub>E</sub>X

SPIN 10492623

41/3142 - 5 4 3 2 1 0 - Printed on acid-free paper

# Preface

Our aim in writing this book was to provide an extensive set of C++ programs for solving basic numerical problems with *verification of the results*. This *C++ Toolbox for Verified Computing I* is the C++ edition of the *Numerical Toolbox for Verified Computing I*. The programs of the original edition were written in PASCAL-XSC, a PASCAL eXtension for Scientific Computation. Since we published the first edition we have received many requests from readers and users of our tools for a version in C++.

We take the view that C++ is growing in importance in the field of numerical computing. C++ includes C, but as a typed language and due to its modern concepts, it is superior to C. To obtain the degree of efficiency that PASCAL-XSC provides, we used the C-XSC library. C-XSC is a C++ class library for eXtended Scientific Computing. C++ and the C-XSC library are an adequate alternative to special XSC-languages such as PASCAL-XSC or ACRITH-XSC. A shareware version of the C-XSC library and the sources of the toolbox programs are freely available via anonymous ftp or can be ordered against reimbursement of expenses.

The programs of this book do not require a great deal of insight into the features of C++. Particularly, object oriented programming techniques are not required. However, the reader should be familiar with writing programs in a high-level computer language such as PASCAL, C, or FORTRAN. This book is particularly useful for those programmers who have already worked with the PASCAL-XSC edition but have only little knowledge in the C++ language. For those readers our book may be a source of inspiration when switching from PASCAL to C++.

We want to thank the readers of the original version of this book for their overwhelmingly positive comments. We incorporated some minor modifications in our algorithms which take into account the highly valuable suggestions and comments we received from numerous readers, as well as our own experience while using the toolbox. Some errors and misprints in the original edition have now been corrected. Nevertheless, we encourage readers, especially those of this C++ edition, to keep on communicating error reports to us.

Special thanks to Andreas Wiethoff who supported us in all questions concerning C++ and the C-XSC library. Last but not least, we wish to express again our appreciation to all colleagues whose advice helped produce the original edition, particularly those mentioned in the following preface of the PASCAL-XSC edition.

# Preface to the PASCAL–XSC Edition

As suggested by the title of this book *Numerical Toolbox for Verified Computing*, we present an extensive set of sophisticated tools to solve basic numerical problems with a *verification of the results*. We use the features of the scientific computer language PASCAL–XSC to offer modules that can be combined by the reader to his/her individual needs. Our overriding concern is reliability – the *automatic verification of the result* a computer returns for a given problem. All algorithms we present are influenced by this central concern. We must point out that there is no relationship between our methods of *numerical result verification* and the methods of *program verification* to prove the correctness of an implementation for a given algorithm.

This book is the first to offer a general discussion on

- arithmetic and computational reliability,
- analytical mathematics and verification techniques,
- algorithms, and
- (most importantly) actual implementations in the form of working computer routines.

Our task has been to find the right balance among these ingredients for each topic. For some topics, we have placed a little more emphasis on the algorithms. For other topics, where the mathematical prerequisites are universally held, we have tended towards more in-depth discussion of the nature of the computational algorithms, or towards practical questions of implementation. For all topics, we present examples, exercises, and numerical results demonstrating the application of the routines presented.

The different chapters of this volume require different levels of knowledge in numerical analysis. Most numerical toolboxes have, after all, tools at varying levels of complexity. Chapters 2, 3, 4, 5, 6, and 10 are suitable for an advanced undergraduate course on numerical computation for science or engineering majors. Other chapters range from the level of a graduate course to that of a professional reference. An attractive feature of this approach is that you can use the book at increasing levels of sophistication as your experience grows. Even inexperienced readers can use our most advanced routines as black boxes. Having done so, these readers can go back and learn what secrets are inside.

The central theme in this book is that practical methods of numerical computation can be simultaneously efficient, clever, clear, and (most importantly) reliable.

We firmly reject the alternative viewpoint that such computational methods must necessarily be so obscure and complex as to be useful only in “black box” form where you have to believe in any calculated result.

This book introduces many computational verification techniques. We want to teach you to take apart these black boxes and to put them back together again, modifying them to suit your specific needs. We assume that you are mathematically literate, i.e. that you have the normal mathematical preparation associated with an undergraduate degree in a mathematical, computational, or physical science, or engineering, or economics, or a quantitative social science. We assume that you know how to program a computer and that you have some knowledge of scientific computation, numerical analysis, or numerical methods. We do not assume that you have any prior formal knowledge of numerical verification or any familiarity with interval analysis. The necessary concepts are introduced.

Volume 1 of *Numerical Toolbox for Verified Computing* provides algorithms and programs to solve basic numerical problems using automatic result verification techniques.

Part I contains two introductory chapters on the features of the scientific computer language PASCAL-XSC and on the basics and terminology of interval arithmetic. Within these chapters, the important correlation between the arithmetic capability and computational accuracy and mathematical fixed-point theory is also discussed.

Part II addresses one-dimensional problems: evaluation of polynomials and general arithmetic expressions, nonlinear root-finding, automatic differentiation, and optimization. Even though only one-dimensional problems treated in this part, the verification methods sometimes require multi-dimensional features like vector or matrix operations.

In Part III, we present routines to solve multi-dimensional problems such as linear and nonlinear systems of equations, linear and global optimization, and automatic differentiation for gradients, Hessians, and Jacobians.

Further volumes of *Numerical Toolbox for Verified Computing* are in preparation covering computational methods in the field of linear systems of equations for complex, interval, and complex interval coefficients, sparse linear systems, eigenvalue problems, matrix exponential, quadrature, automatic differentiation for Taylor series, initial value, boundary value and eigenvalue problems of ordinary differential equations, and integral equations. Editions of the program source code of this volume in the C++ computer language are also in preparation.

Some of the subjects that we cover in detail are not usually found in standard numerical analysis texts. Although this book is intended primarily as a reference text for anyone wishing to apply, modify, or develop routines to obtain mathematically certain and reliable results, it could also be used as a textbook for an advanced course in scientific computation with automatic result verification.

We express our appreciation to all our colleagues whose comments on our book were constructive and encouraging, and we thank our students for their help in testing our routines, modules, and programs. We are very grateful to Prof. Dr. George Corliss (Marquette University, Milwaukee, USA) who helped to polish the text and

the contents. His comments and advice based on his numerical and computational experience greatly improved the presentation of our tools for Verified Computing.

*Karlsruhe, September 1993*

*The Authors*

## The computer programs in this book

and a shareware version of the C-XSC library are available in several machine-readable formats. To purchase diskettes in IBM-PC compatible format, use the order form at the end of the book. The programs and the library are also available by anonymous ftp from

`iamk4515.mathematik.uni-karlsruhe.de` (129.13.129.15)

in subdirectory

`pub/toolbox/cxsc.`

Technical questions, corrections, and requests for information on other available formats and software products should be directed to *Numerical Toolbox Software, Institut für Angewandte Mathematik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, e-mail: toolbox@iampc4.mathematik.uni-karlsruhe.de.*

# Table of Contents

<b>1</b>	<b>Introduction</b>	1
1.1	Advice for Quick Reading	1
1.2	Structure of the Book	2
1.3	Typography	3
1.4	Algorithmic Notation	3
1.5	Implementation	4
1.6	Computational Environment	6
1.7	Why Numerical Result Verification?	6
1.7.1	A Brief History of Computing	7
1.7.2	Arithmetic on Computers	8
1.7.3	Extensions of Ordinary Floating-Point Arithmetic	9
1.7.4	Scientific Computation with Automatic Result Verification	11
1.7.5	Program Verification versus Numerical Verification	14
<b>I</b>	<b>Preliminaries</b>	15
<b>2</b>	<b>The Features of C-XSC</b>	17
2.1	Data Types, Predefined Operators, and Functions	18
2.2	Vector and Matrix Handling	21
2.3	Dot Product Expressions	23
2.4	Input and Output	25
2.5	Data Conversion	26
2.6	Predefined Modules	27
2.7	C-XSC or Other Libraries?	28
<b>3</b>	<b>Mathematical Preliminaries</b>	30
3.1	Real Interval Arithmetic	30
3.2	Complex Interval Arithmetic	37
3.3	Extended Interval Arithmetic	39
3.4	Interval Vectors and Matrices	41
3.5	Floating-Point Arithmetic	42
3.6	Floating-Point Interval Arithmetic	44
3.7	The Problem of Data Conversion	46
3.8	Principles of Numerical Verification	50



<b>II One-Dimensional Problems</b>	55
<b>4 Evaluation of Polynomials</b>	57
4.1 Theoretical Background	57
4.1.1 Description of the Problem	57
4.1.2 Iterative Solution	58
4.2 Algorithmic Description	59
4.3 Implementation and Examples	61
4.3.1 C++ Program Code	61
4.3.1.1 Module rpoly	61
4.3.1.2 Module rpeval	63
4.3.2 Examples	65
4.3.3 Restrictions and Hints	68
4.4 Exercises	68
4.5 References and Further Reading	68
<b>5 Automatic Differentiation</b>	70
5.1 Theoretical Background	70
5.2 Algorithmic Description	72
5.3 Implementation and Examples	74
5.3.1 C++ Program Code	74
5.3.1.1 Module ddf_ari	74
5.3.2 Examples	88
5.3.3 Restrictions and Hints	91
5.4 Exercises	91
5.5 References and Further Reading	91
<b>6 Nonlinear Equations in One Variable</b>	93
6.1 Theoretical Background	93
6.2 Algorithmic Description	95
6.3 Implementation and Examples	98
6.3.1 C++ Program Code	98
6.3.1.1 Module xi_ari	99
6.3.1.2 Module nlfzero	103
6.3.2 Example	108
6.3.3 Restrictions and Hints	110
6.4 Exercises	111
6.5 References and Further Reading	111
<b>7 Global Optimization</b>	113
7.1 Theoretical Background	113
7.1.1 Midpoint Test	114
7.1.2 Monotonicity Test	115
7.1.3 Concavity Test	116
7.1.4 Interval Newton Step	116
7.1.5 Verification	117

7.2	Algorithmic Description . . . . .	117
7.3	Implementation and Examples . . . . .	123
7.3.1	C++ Program Code . . . . .	123
7.3.1.1	Module lst1_ari . . . . .	123
7.3.1.2	Module gop1 . . . . .	128
7.3.2	Examples . . . . .	134
7.3.3	Restrictions and Hints . . . . .	137
7.4	Exercises . . . . .	138
7.5	References and Further Reading . . . . .	139
<b>8</b>	<b>Evaluation of Arithmetic Expressions . . . . .</b>	<b>140</b>
8.1	Theoretical Background . . . . .	140
8.1.1	A Nonlinear Approach . . . . .	140
8.2	Algorithmic Description . . . . .	143
8.3	Implementation and Examples . . . . .	146
8.3.1	C++ Program Code . . . . .	146
8.3.1.1	Module expreal . . . . .	147
8.3.2	Examples . . . . .	158
8.3.3	Restrictions, Hints, and Improvements . . . . .	162
8.4	Exercises . . . . .	162
8.5	References and Further Reading . . . . .	163
<b>9</b>	<b>Zeros of Complex Polynomials . . . . .</b>	<b>164</b>
9.1	Theoretical Background . . . . .	164
9.1.1	Description of the Problem . . . . .	164
9.1.2	Iterative Approach . . . . .	165
9.2	Algorithmic Description . . . . .	168
9.3	Implementation and Examples . . . . .	173
9.3.1	C++ Program Code . . . . .	173
9.3.1.1	Module cpoly . . . . .	173
9.3.1.2	Module cipoly . . . . .	174
9.3.1.3	Module cpzero . . . . .	176
9.3.2	Example . . . . .	182
9.3.3	Restrictions and Hints . . . . .	184
9.4	Exercises . . . . .	184
9.5	References and Further Reading . . . . .	185
<b>III</b>	<b>Multi-Dimensional Problems . . . . .</b>	<b>187</b>
<b>10</b>	<b>Linear Systems of Equations . . . . .</b>	<b>189</b>
10.1	Theoretical Background . . . . .	189
10.1.1	A Newton-like Method . . . . .	189
10.1.2	The Residual Iteration Scheme . . . . .	190
10.1.3	How to Compute the Approximate Inverse . . . . .	190
10.2	Algorithmic Description . . . . .	191
10.3	Implementation and Examples . . . . .	195

10.3.1	C++ Program Code . . . . .	195
10.3.1.1	Module <code>matinv</code> . . . . .	195
10.3.1.2	Module <code>linsys</code> . . . . .	199
10.3.2	Example . . . . .	205
10.3.3	Restrictions and Improvements . . . . .	207
10.4	Exercises . . . . .	207
10.5	References and Further Reading . . . . .	209
<b>11</b>	<b>Linear Optimization . . . . .</b>	<b>210</b>
11.1	Theoretical Background . . . . .	210
11.1.1	Description of the Problem . . . . .	210
11.1.2	Verification . . . . .	211
11.2	Algorithmic Description . . . . .	213
11.3	Implementation and Examples . . . . .	219
11.3.1	C++ Program Code . . . . .	219
11.3.1.1	Module <code>set_ari</code> . . . . .	219
11.3.1.2	Module <code>lop_ari</code> . . . . .	223
11.3.1.3	Module <code>rev_simp</code> . . . . .	227
11.3.1.4	Module <code>lop</code> . . . . .	230
11.3.2	Examples . . . . .	238
11.3.3	Restrictions and Hints . . . . .	242
11.4	Exercises . . . . .	242
11.5	References and Further Reading . . . . .	243
<b>12</b>	<b>Automatic Differentiation for Gradients, Jacobians, and Hessians 244</b>	
12.1	Theoretical Background . . . . .	244
12.2	Algorithmic Description . . . . .	247
12.3	Implementation and Examples . . . . .	249
12.3.1	C++ Program Code . . . . .	249
12.3.1.1	Module <code>hess_ari</code> . . . . .	249
12.3.1.2	Module <code>grad_ari</code> . . . . .	269
12.3.2	Examples . . . . .	285
12.3.3	Restrictions and Hints . . . . .	291
12.4	Exercises . . . . .	291
12.5	References and Further Reading . . . . .	292
<b>13</b>	<b>Nonlinear Systems of Equations . . . . . 293</b>	
13.1	Theoretical Background . . . . .	293
13.1.1	Gauss-Seidel Iteration . . . . .	294
13.2	Algorithmic Description . . . . .	296
13.3	Implementation and Examples . . . . .	300
13.3.1	C++ Program Code . . . . .	300
13.3.1.1	Module <code>nlinsys</code> . . . . .	301
13.3.2	Example . . . . .	307
13.3.3	Restrictions, Hints, and Improvements . . . . .	309

13.4 Exercises . . . . .	310
13.5 References and Further Reading . . . . .	311
<b>14 Global Optimization . . . . .</b>	<b>312</b>
14.1 Theoretical Background . . . . .	312
14.1.1 Midpoint Test . . . . .	313
14.1.2 Monotonicity Test . . . . .	313
14.1.3 Concavity Test . . . . .	314
14.1.4 Interval Newton Step . . . . .	314
14.1.5 Verification . . . . .	315
14.2 Algorithmic Description . . . . .	316
14.3 Implementation and Examples . . . . .	323
14.3.1 C++ Program Code . . . . .	323
14.3.1.1 Module lst_ari . . . . .	323
14.3.1.2 Module gop . . . . .	328
14.3.2 Examples . . . . .	336
14.3.3 Restrictions and Hints . . . . .	340
14.4 Exercises . . . . .	341
14.5 References and Further Reading . . . . .	342
<b>Appendix</b>	
<b>A Utility Modules . . . . .</b>	<b>343</b>
A.1 Module r_util . . . . .	343
A.2 Module i_util . . . . .	343
A.3 Module ci_util . . . . .	348
A.4 Module mv_util . . . . .	349
A.5 Module mvi_util . . . . .	351
<b>B Alphabetical List of Modules . . . . .</b>	<b>356</b>
<b>C List of Special Symbols . . . . .</b>	<b>357</b>
<b>Bibliography . . . . .</b>	<b>360</b>
<b>Index . . . . .</b>	<b>366</b>

# List of Figures

1.1	An interval evaluation provides the guarantee for positive $f$ . . . . .	11
2.1	Hierarchy of the data types and header files of the C-XSC library . . .	27
3.1	Attributes of an interval $[x]$ . . . . .	31
3.2	Distance between two intervals $[x]$ and $[y]$ . . . . .	32
3.3	Different interval extensions of a function $f(x)$ . . . . .	36
3.4	Some subset relations for intervals . . . . .	38
3.5	Intersection of complex intervals . . . . .	38
3.6	Wrapping effect caused by multiplication . . . . .	39
3.7	A three-dimensional real interval vector or box . . . . .	41
3.8	The different roundings . . . . .	43
3.9	A floating-point interval . . . . .	45
3.10	<i>A priori</i> method without (left picture) and with intersection . . . . .	52
3.11	<i>A posteriori</i> method without (left picture) and with $\varepsilon$ -inflation . . . . .	52
4.1	Polynomial $p(t)$ for $t \in [1.9995, 2.0005]$ . . . . .	65
4.2	Polynomial $q(t)$ for $t \in [0.99999, 1.00001]$ . . . . .	66
6.1	Interval Newton step with $0 \notin f'([x]^{(k)})$ . . . . .	94
6.2	Extended interval Newton step with $0 \in f'([x]^{(k)})$ . . . . .	94
6.3	Function $f(x) = e^{-3x} - \sin^3 x$ . . . . .	108
7.1	Midpoint test . . . . .	114
7.2	Monotonicity test . . . . .	115
7.3	Concavity test . . . . .	116
7.4	Function $f$ (test function $f_4$ in [93]) . . . . .	134
7.5	Shubert's function $g$ (test function $f_3$ in [93]) . . . . .	135
7.6	Function $h$ (test function $f_1$ in [93]) with several local minima . . . . .	138
7.7	Test function $r$ with a sharp valley . . . . .	138
11.1	Two-dimensional optimization problem . . . . .	238
14.1	Function $f_B(x)$ of Branin . . . . .	337
14.2	Function $f_L(x)$ of Levy with about 700 local minima . . . . .	337
14.3	Six-Hump Camel-Back function . . . . .	341