

Luc Jaulin, Michel Kieffer, Olivier Didrit
and Éric Walter

Applied Interval Analysis

with Examples in
Parameter and State Estimation,
Robust Control and Robotics

June 11, 2001

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Preface

At the core of many engineering problems is the solution of sets of equations and inequalities, and the optimization of cost functions. Unfortunately, except in special cases, such as when a set of equations is linear in its unknowns or when a convex cost function has to be minimized under convex constraints, the results obtained by conventional numerical methods are only local and cannot be guaranteed. This means, for example, that the actual global minimum of a cost function may not be reached, or that some global minimizers of this cost function may escape detection. By contrast, interval analysis makes it possible to obtain guaranteed approximations of the set of *all* the *actual* solutions of the problem being considered. This, together with the lack of books presenting interval techniques in such a way that they could become part of any engineering numerical tool kit, motivated the writing of this book.

The adventure started in 1991 with the preparation by Luc Jaulin of his PhD thesis, under Eric Walter's supervision. It continued with their joint supervision of Olivier Didrit's and Michel Kieffer's PhD theses. More than two years ago, when we presented our book project to Springer, we naively thought that redaction would be a simple matter, given what had already been achieved... Actually, this book is the result of fierce negotiations between its authors about what should be said, and how! At times, we feared that we might never end up with an actual book, but we feel that the result was worth the struggle.

There were at least two ideas on which we easily agreed, though. First, the book should be as simple and understandable as possible, which is why there are so many illustrations and examples. Secondly, readers willing to experiment with interval analysis on their own applications should be given the power to do so.

Many people contributed to our conversion to interval analysis, and it is impossible to quote all of them, but we would like at least to thank Vladik Kreinovich for all the energy that he puts into the Interval Computations WEB site and for all that we learned there.

Special thanks are due to Michel Petitot for his help in exploring the mysteries of ADA and the Stewart-Gough platform, to Dominique Meizel for introducing us to robot localization and tracking, to Olaf Knüppel and Siegfried

M. Rump for making PROFIL/BIAS and INTLAB available, to Isabelle Braems, Martine Ceberio, Ramon Moore, Stefan Ratschan and Nathalie Revol for their constructive remarks when reading earlier versions of the manuscript, and to our editorial assistant Oliver Jackson, whose friendly enquiries were instrumental in the release of this book this millennium.

We would also like to express our gratitude to Guy Demoment, head of the *Laboratoire des Signaux et Systèmes* and to Jean-Louis Ferrier, head of the *Laboratoire d'Ingénierie des Systèmes Automatisés* for their support and the way they managed to shield us from the perturbations of the outside world.

The French *Centre National de la Recherche Scientifique* provided us with ideal working conditions, and partial support by *INTAS* is also gratefully acknowledged.

Contents

Preface	v
Notation	xiii

Part I. Introduction

1. Introduction	3
1.1 What Are the Key Concepts?	4
1.2 How Did the Story Start?	4
1.3 What About Complexity?	5
1.4 How is the Book Organized?	6

Part II. Tools

2. Interval Analysis	11
2.1 Introduction	11
2.2 Operations on Sets	11
2.2.1 Purely set-theoretic operations	11
2.2.2 Extended operations	12
2.2.3 Properties of set operators	13
2.2.4 Wrappers	15
2.3 Interval Analysis	17
2.3.1 Intervals	18
2.3.2 Interval computation	19
2.3.3 Closed intervals	20
2.3.4 Interval vectors	23
2.3.5 Interval matrices	25
2.4 Inclusion Functions	27
2.4.1 Definitions	27
2.4.2 Natural inclusion functions	29
2.4.3 Centred inclusion functions	33
2.4.4 Mixed centred inclusion functions	34

2.4.5	Taylor inclusion functions	35
2.4.6	Comparison	35
2.5	Inclusion Tests	38
2.5.1	Interval Booleans	38
2.5.2	Tests	40
2.5.3	Inclusion tests for sets	42
2.6	Conclusions	42
3.	Subpavings	45
3.1	Introduction	45
3.2	Set Topology	46
3.2.1	Distances between compact sets	46
3.2.2	Enclosure of compact sets between subpavings	48
3.3	Regular Subpavings	49
3.3.1	Pavings and subpavings	50
3.3.2	Representing a regular subpaving as a binary tree	51
3.3.3	Basic operations on regular subpavings	52
3.4	Implementation of Set Computation	54
3.4.1	Set inversion	55
3.4.2	Image evaluation	59
3.5	Conclusions	63
4.	Contractors	65
4.1	Introduction	65
4.2	Basic Contractors	67
4.2.1	Finite subsolvers	67
4.2.2	Intervalization of finite subsolvers	69
4.2.3	Fixed-point methods	72
4.2.4	Forward–backward propagation	77
4.2.5	Linear programming approach	81
4.3	External Approximation	82
4.3.1	Principle	83
4.3.2	Preconditioning	84
4.3.3	Newton contractor	86
4.3.4	Parallel linearization	87
4.3.5	Using formal transformations	88
4.4	Collaboration Between Contractors	90
4.4.1	Principle	90
4.4.2	Contractors and inclusion functions	95
4.5	Contractors for Sets	97
4.5.1	Definitions	97
4.5.2	Sets defined by equality and inequality constraints	99
4.5.3	Improving contractors using local search	99
4.6	Conclusions	100

5. Solvers	103
5.1 Introduction	103
5.2 Solving Square Systems of Non-linear Equations	104
5.3 Characterizing Sets Defined by Inequalities	106
5.4 Interval Hull of a Set Defined by Inequalities	111
5.4.1 First approach	112
5.4.2 Second approach	113
5.5 Global Optimization	117
5.5.1 The Moore–Skelboe algorithm	120
5.5.2 Hansen’s algorithm	121
5.5.3 Using interval constraint propagation	125
5.6 Minimax Optimization	126
5.6.1 Unconstrained case	127
5.6.2 Constrained case	131
5.6.3 Dealing with quantifiers	133
5.7 Cost Contours	135
5.8 Conclusions	136

Part III. Applications

6. Estimation	141
6.1 Introduction	141
6.2 Parameter Estimation Via Optimization	144
6.2.1 Least-square parameter estimation in compartmental modelling	145
6.2.2 Minimax parameter estimation	148
6.3 Parameter Bounding	155
6.3.1 Introduction	155
6.3.2 The values of the independent variables are known	158
6.3.3 Robustification against outliers	160
6.3.4 The values of the independent variables are uncertain ..	164
6.3.5 Computation of the interval hull of the posterior feasible set	167
6.4 State Bounding	168
6.4.1 Introduction	168
6.4.2 Bounding the initial state	171
6.4.3 Bounding all variables	171
6.4.4 Bounding by constraint propagation	174
6.5 Conclusions	184
7. Robust Control	187
7.1 Introduction	187
7.2 Stability of Deterministic Linear Systems	188
7.2.1 Characteristic polynomial	189

7.2.2	Routh criterion	189
7.2.3	Stability degree	190
7.3	Basic Tests for Robust Stability	193
7.3.1	Interval polynomials	195
7.3.2	Polytope polynomials	196
7.3.3	Image-set polynomials	196
7.3.4	Conclusion	198
7.4	Robust Stability Analysis	198
7.4.1	Stability domains	198
7.4.2	Stability degree	201
7.4.3	Value-set approach	205
7.4.4	Robust stability margins	211
7.4.5	Stability radius	216
7.5	Controller Design	220
7.6	Conclusions	223
8.	Robotics	225
8.1	Introduction	225
8.2	Forward Kinematics Problem for Stewart–Gough Platforms	226
8.2.1	Stewart–Gough platforms	226
8.2.2	From the frame of the mobile plate to that of the base	227
8.2.3	Equations to be solved	229
8.2.4	Solution	230
8.3	Path Planning	234
8.3.1	Graph discretization of configuration space	237
8.3.2	Algorithms for finding a feasible path	239
8.3.3	Test case	241
8.4	Localization and Tracking of a Mobile Robot	248
8.4.1	Formulation of the static localization problem	249
8.4.2	Model of the measurement process	253
8.4.3	Set inversion	257
8.4.4	Dealing with outliers	259
8.4.5	Static localization example	260
8.4.6	Tracking	263
8.4.7	Example	264
8.5	Conclusions	267

Part IV. Implementation

9.	Automatic Differentiation	271
9.1	Introduction	271
9.2	Forward and Backward Differentiations	271
9.2.1	Forward differentiation	272
9.2.2	Backward differentiation	273

9.3	Differentiation of Algorithms.....	275
9.3.1	First assumption	275
9.3.2	Second assumption	278
9.3.3	Third assumption.....	279
9.4	Examples	281
9.4.1	Example 1	281
9.4.2	Example 2	284
9.5	Conclusions.....	285
10.	Guaranteed Computation with Floating-point Numbers ..	287
10.1	Introduction	287
10.2	Floating-point Numbers and IEEE 754	287
10.2.1	Representation	288
10.2.2	Rounding.....	289
10.2.3	Special quantities	291
10.3	Intervals and IEEE 754	292
10.3.1	Machine intervals	293
10.3.2	Closed interval arithmetic	294
10.3.3	Handling elementary functions	295
10.3.4	Improvements	297
10.4	Interval Resources	297
10.5	Conclusions.....	299
11.	Do It Yourself	301
11.1	Introduction	301
11.2	Notions of C++	301
11.2.1	Program structure	302
11.2.2	Standard types	303
11.2.3	Pointers	304
11.2.4	Passing parameters to a function	304
11.3	INTERVAL Class	305
11.3.1	Constructors and destructor	307
11.3.2	Other member functions.....	308
11.3.3	Mathematical functions	313
11.4	Intervals with PROFIL/BIAS	315
11.4.1	BIAS	315
11.4.2	PROFIL	316
11.4.3	Getting started.....	317
11.5	Exercises on Intervals	318
11.6	Interval Vectors	319
11.6.1	INTERVAL_VECTOR class	320
11.6.2	Constructors, assignment and function call operators ...	321
11.6.3	Friend functions	323
11.6.4	Utilities	325
11.7	Vectors with PROFIL/BIAS	326

11.8 Exercises on Interval Vectors.....	327
11.9 Interval Matrices	331
11.10 Matrices with PROFIL/BIAS	332
11.11 Exercises on Interval Matrices.....	333
11.12 Regular Subpavings with PROFIL/BIAS	336
11.12.1 NODE class	336
11.12.2 Set inversion with subpavings	339
11.12.3 Image evaluation with subpavings	342
11.12.4 System simulation and state estimation with subpavings	347
11.13 Error Handling.....	349
11.13.1 Using <code>exit</code>	349
11.13.2 Exception handling	350
11.13.3 Mathematical errors	351
References	353
Index	373

Notation

The following tables describe the main typographic conventions and symbols to be used.

Punctual quantities

x	:	punctual scalar
x^*	:	actual value of an uncertain variable x
\tilde{x}	:	prior value of an uncertain variable x
\hat{x}	:	posterior value of an uncertain variable x
\mathbf{x}	:	punctual column vector
\mathbf{x}^T	:	punctual row vector
$\mathbf{0}$:	vector of zeros
$\mathbf{1}$:	vector of ones
\mathbf{X}	:	punctual matrix
$\mathbf{O}, \mathbf{O}_{n \times m}$:	matrix of zeros, $(n \times m)$ matrix of zeros
\mathbf{I}, \mathbf{I}_n	:	identity matrix, $(n \times n)$ identity matrix
$\text{Im}(s)$:	imaginary part of s
$\text{Re}(s)$:	real part of s

Sets

\emptyset	:	empty set
\mathbb{S}	:	set
\mathbb{N}	:	set of all positive integers
\mathbb{Z}	:	set of all integers
\mathbb{R}	:	set of all real numbers
\mathbb{IR}	:	set of all interval real numbers
\mathbb{C}	:	set of all complex numbers
\mathbb{C}^-	:	set of all complex numbers with a strictly negative real part
\mathbb{B}	:	set of all Boolean numbers
\mathbb{IB}	:	set of all interval Boolean numbers
$\partial\mathbb{S}$:	boundary of \mathbb{S}
$[\mathbb{S}]$:	interval hull of \mathbb{S}
$\overline{\mathbb{S}}$:	outer approximation of \mathbb{S}
$\underline{\mathbb{S}}$:	inner approximation of \mathbb{S}
\mathcal{L}	:	list, stack, queue, tree or graph

Intervals

$[x] = [\underline{x}, \overline{x}]$:	interval scalar
$[\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$:	interval vector (or box)
$[\mathbf{X}] = [\underline{\mathbf{X}}, \overline{\mathbf{X}}]$:	interval matrix
$[x_i] = ([\mathbf{x}])_i$:	i th entry of $[\mathbf{x}]$
$[x_{ij}] = ([\mathbf{X}])_{ij}$:	entry of $[\mathbf{X}]$ at i th row and j th column
$\text{lb}([x])$:	lower bound of $[x]$
$\text{ub}([x])$:	upper bound of $[x]$
$w([x])$:	width of $[x]$
$\text{mid}([x])$:	centre of $[x]$

Other symbols

\triangleq	: equal by definition
$:=$: assignment operator
\forall	: universal quantifier (<i>for all</i>)
\exists	: existential quantifier (<i>there exists</i>)
\neg	: logical complementation
\wedge	: logical AND
\vee	: logical OR
$\mathbb{A} \times \mathbb{B}$: Cartesian product of \mathbb{A} and \mathbb{B}
$\mathbb{A} \setminus \mathbb{B}$: $\{x \mid (x \in \mathbb{A}) \wedge (x \notin \mathbb{B})\}$
$\mathbb{A} \sqcup \mathbb{B}$: interval union of \mathbb{A} and \mathbb{B} , equal to $[\mathbb{A} \cup \mathbb{B}]$

Functions

Functions are denoted with the same typographical convention as the elements of their image spaces, thus $[f](\cdot)$ is a scalar interval function and $[\mathbf{f}](\cdot)$ a vector interval function.

If $\mathbf{f}(\cdot)$ is a once-differentiable function from \mathbb{R}^{n_x} to \mathbb{R}^{n_y} , then its *Jacobian matrix* at \mathbf{x} is

$$\mathbf{J}_{\mathbf{f}}(\mathbf{x}) \triangleq \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_{n_x}}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{n_y}}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_{n_y}}{\partial x_{n_x}}(\mathbf{x}) \end{pmatrix}.$$

If $f(\cdot)$ is a once-differentiable function from \mathbb{R}^{n_x} to \mathbb{R} , then its *gradient* at \mathbf{x} is

$$\mathbf{g}_f(\mathbf{x}) \triangleq \begin{pmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_{n_x}}(\mathbf{x}) \end{pmatrix}.$$

If $f(\cdot)$ is twice differentiable, then its *Hessian matrix* at \mathbf{x} is the (symmetric) Jacobian matrix associated with its gradient, *i.e.*,

$$\mathbf{H}_f(\mathbf{x}) \triangleq \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_{n_x} \partial x_1}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_{n_x}}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_{n_x}^2}(\mathbf{x}) \end{pmatrix}.$$

Algorithms

Algorithms are described in a pseudo-code allowing the usual mathematical notation. The most important arguments are listed after the NAME of the algorithm as input arguments (in:), output arguments (out:) or input-output arguments (inout:). To facilitate reading, we take the liberty to omit some of them, such as inclusion functions, gradients, Hessian matrices... Blocks of statements are indicated by indentation. Any return statement causes an immediate return from the current algorithm. Return statements at the end of the algorithms are implicit.

For details about the implementation of these algorithms, see Chapter 11, where C++ code is set in `Typewriter`.