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

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
$\check{x}$	:	prior value of an uncertain variable x
$\widehat{x}$	:	posterior value of an uncertain variable x
$\mathbf{x}$	:	punctual column vector
$\mathbf{x}^{\mathrm{T}}$	:	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
$\mathrm{Im}(s)$	:	imaginary part of s
$\mathrm{Re}(s)$	:	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)
$[\mathbf{X}] = \left[\underline{\mathbf{X}}, \overline{\mathbf{X}}\right]$	:	interval matrix
$[x_i] = ([\mathbf{x}])_i$	:	ith entry of $[\mathbf{x}]$
$[x_{ij}] = ([\mathbf{X}])_{ij}$	:	entry of $[\mathbf{X}]$ at ith row and jth column
$\mathrm{lb}([x])$	:	lower bound of $[x]$
$\mathrm{ub}([x])$	:	upper bound of $[x]$
$w([x])$	:	width of $[x]$
$\mathrm{mid}([x])$	:	centre of $[x]$

Other symbols

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

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 $\mathbf{f}(.)$ is a once-differentiable function from $\mathbb{R}^{n_\mathrm{x}}$ to $\mathbb{R}^{n_\mathrm{y}}$, then its *Jacobian matrix* at $\mathbf{x}$ is

$$\mathbf{J}_\mathbf{f}(\mathbf{x}) \triangleq \begin{pmatrix} \dfrac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \dfrac{\partial f_1}{\partial x_{n_\mathrm{x}}}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_{n_\mathrm{y}}}{\partial x_1}(\mathbf{x}) & \cdots & \dfrac{\partial f_{n_\mathrm{y}}}{\partial x_{n_\mathrm{x}}}(\mathbf{x}) \end{pmatrix}.$$

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

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

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

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

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`.