XML in Scientific Computing

CHAPMAN & HALL/CRC
Numerical Analysis and Scientific Computing

Aims and scope:
Scientific computing and numerical analysis provide invaluable tools for the sciences and engineering. This series aims to capture new developments and summarize state-of-the-art methods over the whole spectrum of these fields. It will include a broad range of textbooks, monographs, and handbooks. Volumes in theory, including discretisation techniques, numerical algorithms, multiscale techniques, parallel and distributed algorithms, as well as applications of these methods in multi-disciplinary fields, are welcome. The inclusion of concrete real-world examples is highly encouraged. This series is meant to appeal to students and researchers in mathematics, engineering, and computational science.

Proposals for the series should be submitted to one of the series editors above or directly to:
CRC Press, Taylor & Francis Group
4th, Floor, Albert House
1-4 Singer Street
London EC2A 4BQ
UK

Published Titles

Classical and Modern Numerical Analysis: Theory, Methods and Practice
Azmy S. Ackleh, Edward James Allen, Ralph Baker Kearfott, and Padmanabhan Seshaiyer

Cloud-Computing: Data-Intensive Computing and Scheduling
Frédéric Magoulès, Jie Pan, and Fei Teng

Computational Fluid Dynamics
Frédéric Magoulès

A Concise Introduction to Image Processing using C++
Meiqing Wang and Choi-Hong Lai

Decomposition Methods for Differential Equations: Theory and Applications
Juergen Geiser

Desktop Grid Computing
Christophe Cérin and Gilles Fedak

Discrete Variational Derivative Method: A Structure-Preserving Numerical Method for Partial Differential Equations
Daisuke Furihata and Takayasu Matsuo

Grid Resource Management: Toward Virtual and Services Compliant Grid Computing
Frédéric Magoulès, Thi-Mai-Huong Nguyen, and Lei Yu

Fundamentals of Grid Computing: Theory, Algorithms and Technologies
Frédéric Magoulès

Handbook of Sinc Numerical Methods
Frank Stenger

Introduction to Grid Computing
Frédéric Magoulès, Jie Pan, Kiat-An Tan, and Abhinit Kumar

Iterative Splitting Methods for Differential Equations
Juergen Geiser

Mathematical Objects in C++: Computational Tools in a Unified Object-Oriented Approach
Yair Shapira

Numerical Linear Approximation in C
Nabih N. Abdelmalek and William A. Malek

Numerical Techniques for Direct and Large-Eddy Simulations
Xi Jiang and Choi-Hong Lai

Parallel Algorithms
Henri Casanova, Arnaud Legrand, and Yves Robert

Parallel Iterative Algorithms: From Sequential to Grid Computing
Jacques M. Bahi, Sylvain Contassot-Vivier, and Raphael Couturier

Particle Swarm Optimisation: Classical and Quantum Perspectives
Jun Sun, Choi-Hong Lai, and Xiao-Jun Wu

XML in Scientific Computing
C. Pozrikidis

XML in Scientific Computing

C. Pozrikidis

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
Version Date: 20120813

International Standard Book Number: 978-1-4665-1227-6 (Hardback)

Contents

Preface

Xml stands for extensible markup language. In fact, *xml* is not a language, but a systematic way of encoding and formatting data and statements contained in an electronic file according to a chosen tagging system. A tag may represent a general entity, a physical, mathematical, or abstract object, an instruction, or a computer language construct. The data can describe cars and trucks in a dealer's lot, the chapters of a book, the input or output of a scientific experiment or calculation, the eigenvalues of a matrix, and anything else that can be described by numbers and words.

Data presentation and description

In the *xml* framework, information is described and presented in the same document, thus circumventing the need for legends and explanations. For example, we may order:

```
<breakfast> toast and eggs </breakfast>
```

Further cooking instructions can be included between the breakfast tag enclosed by the pointy brackets (`<>`) and its closure denoted by the slash (`/`).

Data reuse

Xml data (input) can be read by a person or parsed and processed by a program (application) that produces a new set of data (output). Although the input is the same, the output depends on the interpretation of the tags formatting the data. The inherent polymorphism allows us to materialize the same original data in different ways. For example:

1. An author may write a book inserting formatting tags between words, equations, and figures according to *xml* conventions and grammar. The text (data) file can be processed to produce books with different appearances.
2. A scientist may write a finite-element code that produces output tagged according to *xml* conventions. The elements can be visualized using different graphics programs and the data can be sent to another person or program to serve as input.
3. A conversation could be transcribed using *xml* grammar and then printed on paper or sent to a telephone to be heard by the recipient. It is not necessary to duplicate the data.
4. A computer program could be written according to generic *xml* conventions. The instructions can be interpreted to produce corresponding code in a chosen programming language.

To demonstrate the concept of data sharing and reuse, we deliver the same instructions to a painter and a sculptor, and ask them to produce corresponding pieces of art. The *xml* data encapsulated in these instructions acquire meaning only when the tags describing the data are implemented by the artists to produce physical objects.

Scientific computing

In scientific computing, we are accustomed to compiling and running a code (application) written in a language of our choice, such as C, C++, *fortran*, or *Matlab*®. The code utilizes parameters and input data that are either embedded in the program (monolithic structure) or read from companion input data files (modular structure). Emphasis is placed on the code and the output is generated readily by running the executable. In most applications, the code is more valuable than the output. The opposite is generally true in the *xml* framework where the data play a prominent role and may even serve to launch an application, as in the case of a telephone that rings only when it receives data.

Xml and scientific computing

Xml has received a great deal of attention in the web programming and software engineering disciplines with reference to data encoding and storage, but far less attention in the mainstream computational science and engineering disciplines. Two main issues of interest in scientific computing are: (*a*) producing *xml* formatted output from code and (*b*) reading *xml* input from a data file, converting it into an appropriate data structure. It is revealing that computing environments familiar to scientists and engineers, such as *Matlab*®, *Mathcad*®, and *Mathematica*®, have embraced the *xml* framework and incorporated add-on libraries to facilitate the handling of *xml* input and output.

Goals of this book

Currently available texts and *web* tutorials on *xml* data formatting discuss *xml* in the context of computer science with a clear focus on *web* and database programming.

The first goal of this book is to introduce and describe *xml* to scientists and engineers with some typesetting and programming experience.

The second goal is to introduce the extensible stylesheet language (*xsl*) with applications in *xml* data processing and numerical computation. Strange though it may seem, an *xsl* code is written according to *xml* conventions, that is, *xsl* is an *xml* implementation.

The third and perhaps most important goal of this book is to review possible ways of saving, importing, and sharing *xml* data in code written in programming languages used most frequently by scientists and engineers. Although references

to *latex*, *html*, *fortran* 77 (simply called *fortran*), C++, and *perl* are made, only cursory familiarity with these languages is assumed and necessary explanations are given. Analogies and parallels will be drawn, and contrasts will be made with *xsl* to underline important similarities and differences in programming procedures.

This book is accompanied by a suite of computer programs and other documents arranged in directories corresponding to the book chapters and appendices.* Internet resources and other information pertinent to *xml* are provided as links at the book website.

C. Pozrikidis

Summer, 2012

*http://dehesa.freeshell.org/XML

Notation

Nomenclature and font conventions adopted in the text are defined in the following table:

Symbol or word	Name or meaning
`()`	parentheses
`[]`	square brackets
`{}`	curly brackets
`<>`	angle (pointy) brackets
`->`	*ascii* arrow
filename	name of a file
`sometext`	text typed in a file
language	name of a computer language
`line`	text typed in the keyboard
result	text shown in the screen
ENTER	Enter key in the keyboard

The names of standard computer languages are treated as regular words whose initial letter is capitalized at the beginning of a sentence and printed in lower case otherwise. File contents, typed instructions, and other data appearing on a computer screen are highlighted.