PREFACE

Iterative methods for solving general, large sparse linear systems have been gaining popularity in many areas of scientific computing. Until recently, direct solution methods were often preferred to iterative methods in real applications because of their robustness and predictable behavior. However, a number of efficient iterative solvers were discovered and the increased need for solving very large linear systems triggered a noticeable and rapid shift toward iterative techniques in many applications.

This trend can be traced back to the 1960s and 1970s when two important developments revolutionized solution methods for large linear systems. First was the realization that one can take advantage of "sparsity" to design special direct methods that can be quite economical. Initiated by electrical engineers, these "direct sparse solution methods" led to the development of reliable and efficient general-purpose direct solution software codes over the next three decades. Second was the emergence of preconditioned conjugate gradient-like methods for solving linear systems. It was found that the combination of preconditioning and Krylov subspace iterations could provide efficient and simple "general-purpose" procedures that could compete with direct solvers. Preconditioning involves exploiting ideas from sparse direct solvers. Gradually, iterative methods started to approach the quality of direct solvers. In earlier times, iterative methods were often special-purpose in nature. They were developed with certain applications in mind, and their efficiency relied on many problem-dependent parameters.

Now, three-dimensional models are commonplace and iterative methods are almost mandatory. The memory and the computational requirements for solving three-dimensional Partial Differential Equations, or two-dimensional ones involving many degrees of freedom per point, may seriously challenge the most efficient direct solvers available today. Also, iterative methods are gaining ground because they are easier to implement efficiently on high-performance computers than direct methods.

My intention in writing this volume is to provide up-to-date coverage of iterative methods for solving large sparse linear systems. I focused the book on practical methods that work for general sparse matrices rather than for any specific class of problems. It is indeed becoming important to embrace applications not necessarily governed by Partial Differential Equations, as these applications are on the rise. Apart from two recent volumes by Axelsson [15] and Hackbusch [116], few books on iterative methods have appeared since the excellent ones by Varga [213]. and later Young [232]. Since then, researchers and practitioners have achieved remarkable progress in the development and use of effective iterative methods. Unfortunately, fewer elegant results have been discovered since the 1950s and 1960s. The field has moved in other directions. Methods have gained not only in efficiency but also in robustness and in generality. The traditional techniques which required rather complicated procedures to determine optimal acceleration parameters have yielded to the parameter-free conjugate gradient class of methods.

The primary aim of this book is to describe some of the best techniques available today, from both preconditioners and accelerators. One of the aims of the book is to provide a good mix of theory and practice. It also addresses some of the current research issues such as parallel implementations and robust preconditioners. The emphasis is on Krylov subspace methods, currently the most practical and common group of techniques used in applications. Although there is a tutorial chapter that covers the discretization of Partial Differential Equations, the book is not biased toward any specific application area. Instead, the matrices are assumed to be general sparse, possibly irregularly structured.

The book has been structured in four distinct parts. The first part, Chapters 1 to 4, presents the basic tools. The second part, Chapters 5 to 8, presents projection methods and Krylov subspace techniques. The third part, Chapters 9 and 10, discusses preconditioning. The fourth part, Chapters 11 to 13, discusses parallel implementations and parallel algorithms.

ACKNOWLEDGMENTS

I am grateful to a number of colleagues who proofread or reviewed different versions of the manuscript. Among them are Randy Bramley (University of Indiana at Bloomingtin), Xiao-Chuan Cai (University of Colorado at Boulder), Tony Chan (University of California at Los Angeles), Jane Cullum (IBM, Yorktown Heights), Alan Edelman (Massachussett Institute of Technology), Paul Fischer (Brown University), David Keyes (Old Dominion University), Beresford Parlett (University of California at Berkeley) and Shang-Hua Teng (University of Minnesota). Their numerous comments, corrections, and encouragements were a highly appreciated contribution. In particular, they helped improve the presentation considerably and prompted the addition of a number of topics missing from earlier versions.

This book evolved from several successive improvements of a set of lecture notes for the course "Iterative Methods for Linear Systems" which I taught at the University of Minnesota in the last few years. I apologize to those students who used the earlier error-laden and incomplete manuscripts. Their input and criticism contributed significantly to improving the manuscript. I also wish to thank those students at MIT (with Alan Edelman) and UCLA (with Tony Chan) who used this book in manuscript form and provided helpful feedback. My colleagues at the university of Minnesota, staff and faculty members, have helped in different ways. I wish to thank in particular Ahmed Sameh for his encouragements and for fostering a productive environment in the department. Finally, I am grateful to the National Science Foundation for their continued financial support of my research, part of which is represented in this work.

Yousef Saad

SUGGESTIONS FOR TEACHING

This book can be used as a text to teach a graduate-level course on iterative methods for linear systems. Selecting topics to teach depends on whether the course is taught in a mathematics department or a computer science (or engineering) department, and whether the course is over a semester or a quarter. Here are a few comments on the relevance of the topics in each chapter.

For a graduate course in a mathematics department, much of the material in Chapter 1 should be known already. For non-mathematics majors most of the chapter must be covered or reviewed to acquire a good background for later chapters. The important topics for the rest of the book are in Sections: 1.8.1, 1.8.3, 1.8.4, 1.9, 1.11. Section 1.12 is best treated at the beginning of Chapter 5. Chapter 2 is essentially independent from the rest and could be skipped altogether in a quarter course. One lecture on finite differences and the resulting matrices would be enough for a non-math course. Chapter 3 should make the student familiar with some implementation issues associated with iterative solution procedures for general sparse matrices. In a computer science or engineering department, this can be very relevant. For mathematicians, a mention of the graph theory aspects of sparse matrices and a few storage schemes may be sufficient. Most students at this level should be familiar with a few of the elementary relaxation techniques covered in Chapter 4. The convergence theory can be skipped for non-math majors. These methods are now often used as preconditioners and this may be the only motive for covering them.

Chapter 5 introduces key concepts and presents projection techniques in general terms. Non-mathematicians may wish to skip Section 5.2.3. Otherwise, it is recommended to start the theory section by going back to Section 1.12 on general definitions on projectors. Chapters 6 and 7 represent the heart of the matter. It is recommended to describe the first algorithms carefully and put emphasis on the fact that they generalize the one-dimensional methods covered in Chapter 5. It is also important to stress the optimality properties of those methods in Chapter 6 and the fact that these follow immediately from the properties of projectors seen in Section 1.12. When covering the algorithms in Chapter 7, it is crucial to point out the main differences between them and those seen in Chapter 6. The variants such as CGS, BICGSTAB, and TFQMR can be covered in a short time, omitting details of the algebraic derivations or covering only one of the three. The class of methods based on the normal equation approach, i.e., Chapter 8, can be skipped in a math-oriented course, especially in the case of a quarter system. For a semester course, selected topics may be Sections 8.1, 8.2, and 8.4.

Currently, preconditioning is known to be the critical ingredient in the success of iterative methods in solving real-life problems. Therefore, at least some parts of Chapter 9 and Chapter 10 should be covered. Section 9.2 and (very briefly) 9.3 are recommended. From Chapter 10, discuss the basic ideas in Sections 10.1 through 10.3. The rest could be skipped in a quarter course.

Chapter 11 may be useful to present to computer science majors, but may be skimmed or skipped in a mathematics or an engineering course. Parts of Chapter 12 could be taught primarily to make the students aware of the importance of "alternative" preconditioners. Suggested selections are: 12.2, 12.4, and 12.7.2 (for engineers). Chapter 13 presents an important research area and is primilarily geared to mathematics majors. Computer scientists or engineers may prefer to cover this material in less detail.

To make these suggestions more specific, the following two tables are offered as sample course outlines. Numbers refer to sections in the text. A semester course represents approximately 30 lectures of 75 minutes each whereas a quarter course is approximately 20 lectures of 75 minutes each. Different topics are selected for a mathematics course and a non-mathematics course.

Semester course		
Weeks	Mathematics	Computer Science / Eng.
1 – 3	1.9 –1.13 2.1 – 2.5 3.1 – 3.3, 3.7	1.1 – 1.6 (Read) 1.7 – 1.13, 2.1 – 2.2 3.1 – 3.7
4 – 6	4.1 – 4.3 5. 1 – 5.4 6.1 – 6.3	4.1 – 4.2 5.1 – 5.2.1 6.1 – 6.3
7 – 9	6.4 – 6.7 (Except 6.5.2) 6.9 – 6.11 7.1 – 7.3	6.4 – 6.5 (Except 6.5.5) 6.7.1, 6.8–6.9, 6.11.3. 7.1 – 7.3
10 – 12	7.4.1; 7.4.2 – 7.4.3 (Read) 8.1, 8.2, 8.4; 9.1 – 9.3 10.1 – 10.3	7.4.1; 7.4.2 – 7.4.3 (Read) 8.1 – 8.3; 9.1 – 9.3 10.1 – 10.4
13 – 15	10.5.1 – 10.5.6 10.6 ; 12.2 – 12.4 13.1 – 13.6	10.5.1 – 10.5.4 11.1 – 11.4 (Read); 11.5 – 11.6 12.1 – 12.2; 12.4 – 12.7

Quarter course		
Weeks	Mathematics	Computer Science / Eng.
1 – 2	1.9 – 1.13, 3.1 – 3.2 4.1 – 4.3	1.1 – 1.6 (Read); 3.1 – 3.7 4.1
3 – 4	5.1 – 5.4 6.1 – 6.4	5.1 – 5.2.1 6.1 – 6.3
5 – 6	6.4 – 6.7 (Except 6.5.2) 6.11, 7.1 – 7.3	6.4 – 6.5 (Except 6.5.5) 6.7.1, 6.11.3, 7.1 – 7.3
7 – 8	7.4.1; 7.4.2 – 7.4.3 (Read) 9.1 – 9.3; 10.1 – 10.3	7.4.1; 7.4.2 – 7.4.3 (Read) 9.1 – 9.3; 10.1 – 10.3
9 – 10	10.6 ; 12.2 – 12.4 13.1 – 13.4	11.1 – 11.4 (Read); 11.5 – 11.6 12.1 – 12.2; 12.4 – 12.7