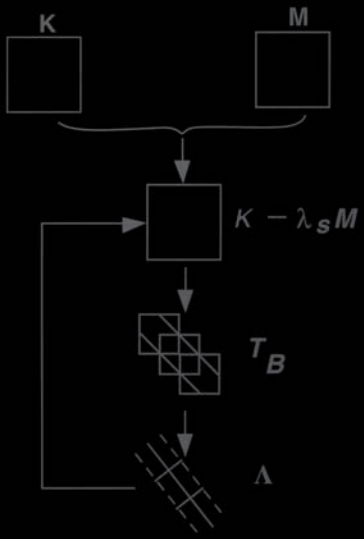
Louis Komzsik

The Lanczos Method

Evolution and Application



SOFTWARE • ENVIRONMENTS • TOOLS

siam

THE LANCZOS METHOD

SOFTWARE • ENVIRONMENTS • TOOLS

The series includes handbooks and software guides as well as monographs on practical implementation of computational methods, environments, and tools. The focus is on making recent developments available in a practical format to researchers and other users of these methods and tools.

Software, Environments, and Tools

Louis Komzsik, *The Lanczos Method: Evolution and Application*

Bard Ermentrout, *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*

V. A. Barker, L. S. Blackford, J. Dongarra, J. Du Croz, S. Hammarling, M. Marinova, J. Waśniewski, and P. Yalamov, *LAPACK95 Users' Guide*

Stefan Goedecker and Adolfy Hoisie, *Performance Optimization of Numerically Intensive Codes*

Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*

Lloyd N. Trefethen, *Spectral Methods in MATLAB*

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide, Third Edition*

Michael W. Berry and Murray Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval*

Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst, *Numerical Linear Algebra for High-Performance Computers*

R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*

Randolph E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0*

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*

Greg Astfalk, editor, *Applications on Advanced Architecture Computers*

Françoise Chaitin-Chatelin and Valérie Frayssé, *Lectures on Finite Precision Computations*

Roger W. Hockney, *The Science of Computer Benchmarking*

Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide, Second Edition*

Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*

J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *Linpack Users' Guide*

Louis Komzsik
Schaeffer Automated Simulation, LLC
Altadena, California

THE LANCZOS METHOD

Evolution and Application

siam
Society for Industrial and Applied Mathematics
Philadelphia

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Komzsik, Louis.
The Lanczos method : evolution and application / Louis Komzsik.
p.cm.
Includes bibliographical references and index.
ISBN 0-89871-537-7 (pbk.)
1. Computer science–Mathematics. 2. Numerical analysis. 3. Computer algorithms. 4. Eigenvalues. I. Title.

QA76.9.M35.K66 2003
004'.01'51–dc21

2002044643

To Stella and Victor

This page intentionally left blank

Contents

This page intentionally left blank

Preface

The subject of this book, the method of Lanczos, is probably one of the most influential methods of computational mathematics in the second half of the 20th century. Since Lanczos's seminal paper [2] in 1950, despite some early setbacks about the applicability of the method in computers with finite precision arithmetic, the method found its way into many aspects of science and engineering. The applications are so widespread that it is practically impossible to describe them in a single book. This book follows the evolution of the method as it became more and more established and understood, and began to solve a wide variety of engineering analysis problems.

My personal involvement with and admiration of the method started in the early 1970s in Budapest as a graduate student at the successor of Lanczos's alma mater. While at that time both Lanczos and his method had somewhat tarnished reputations, for political and numerical reasons, respectively, I was taken by the beauty of the three-member recurrence. The second half of the 1970s saw the restoration of the numerical reputation of the method worldwide, and by the end of the decade Lanczos was also put on his well-deserved pedestal, even in Hungary.

The material in this book comes from seminars and lectures I had given on the topic during the past two decades. The seminars, held by leading corporations of the automobile and aerospace industries in the United States, Europe, and Asia, were attended by engineers and computer scientists and focused on applications of the method in commercial finite element analysis, specifically in structural analysis. The lectures I had given recently as a SIAM Visiting Lecturer at various academic institutions were attended by both faculty and students and centered on practical implementation and computational performance issues. The interest of the audience in both camps and the lack of a text encompassing the evolution of the method contributed to the decision to write this book. I hope that the readers share this interest, enjoy a brief travel of time through the history of the method, and find the book useful in their applications.

The book has two distinct parts. The first part, Chapters 1 through 5, demonstrates the evolution of the method from the review of Lanczos's original method to the state-of-the-art adaptive methods. The second part, Chapters 6 through 10, addresses the practical implementation and industrial application of the method. Specifically, in Chapters 7, 8, and 9 the well-established industrial applications of normal modes and complex eigenvalue analyses, as well as the frequency response analysis, are discussed. The book concludes with the application of the Lanczos method for the solution of linear systems.

While heavy on mathematical content, in order to achieve readability, rigorous statement of theorems and proofs are omitted. Similarly, topics in the linear algebraic foundation

(QR and singular value decomposition, Givens rotations, etc.) are not discussed in detail to keep the focus sharp. Several chapters contain a computational algorithm enabling the reader to implement some of the methods either in a MATLAB environment or in a high-level programming language.

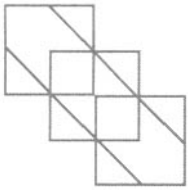
During the past quarter century I have cooperated with many people in various aspects of the Lanczos method. I specifically thank Prof. Beresford Parlett of UC Berkeley and Prof. Gene Golub of Stanford University for their most valuable theoretical influence, which I enjoyed from their books as well as personal contacts. I am also very much indebted to Dr. Horst Simon of Berkeley National Laboratory, Dr. John Lewis of Boeing, and Prof. Zhaojun Bai of UC Davis for their very important cooperation in the practical implementation aspects of the Lanczos method. Finally, special thanks are due to my colleague, Dr. Tom Kowalski, who, besides participating in the implementation of some of the methods mentioned in this book into NASTRAN, has also dutifully proofread the manuscript and provided valuable corrections and suggestions.

Louis Komzsik
2002

Part I

EVOLUTION

This page intentionally left blank



Chapter 1
The classical Lanczos method

At the time of Lanczos's work on the eigenvalue problem during the Second World War, most methods focused on finding the characteristic polynomial of matrices in order to find their eigenvalues. In fact, Lanczos's original paper [2] was also mostly concerned with this problem; however, he was trying to reduce the round-off errors in such calculations. He called his method the method of minimized iterations, which we will now review to lay the foundation.

1.1 The eigenvalue problem

For a real, square matrix A of order n, the product

$$\underline{x}^T A \underline{x} = \sum_{k=1}^{n} \sum_{l=1}^{n} a_{kl} x_k x_l \tag{1.1}$$

defines a quadratic form. This is a continuous function of $\underline{x} = (x_1, x_2, \ldots, x_n)$. When A is symmetric positive definite, the equation

$$\underline{x}^T A \underline{x} = 1 \tag{1.2}$$

defines an n-dimensional ellipsoid in R^n which is in all likelihood rotated. The eigenvalue problem is to find the n principal axes of this ellipsoid which are the eigenvectors $\underline{x}_i$, $i = 1, \ldots, n$. The square roots of the lengths of the principal axes are the eigenvalues λ_i. They satisfy the equation

$$A \underline{x}_i = \lambda_i \underline{x}_i. \tag{1.3}$$

This is easy to verify considering the fact that the directions of the principal axes of the ellipsoid are where the surface normal $\underline{n}$ is colinear with the principal axis location vector pointing to the surface $\underline{x}_i$,

$$\underline{x}_i = c\underline{n}, \tag{1.4}$$

where c is a scalar constant. Since the normal points into the direction of the gradient,

$$\underline{n} = \underline{\nabla}(\underline{x}^T A \underline{x} - 1) = A \underline{x}_i, \tag{1.5}$$

it follows that c is $1/\lambda_i$.

1.2 The method of minimized iterations

Lanczos first considered symmetric matrices ($A^T = A$) and set out to find the characteristic polynomial of

$$G(\mu) = \det(A - \mu I) = 0 \tag{1.6}$$

in order to solve

$$Au = \mu u, \tag{1.7}$$

where u is an eigenvector and μ is the corresponding eigenvalue. In deference to Lanczos, in this section we adhere to his original notation as much as possible. Specifically, the inner products commonly written as $b_0^T b_0$ in today's literature is noted below as b_0^2.

Lanczos sought the characteristic polynomial by generating a sequence of trial vectors, resulting in a successive set of polynomials. Starting from a randomly selected vector b_0, the new vector b_1 is chosen as a certain linear combination of b_0 and Ab_0,

$$b_1 = Ab_0 - \alpha_0 b_0. \tag{1.8}$$

Here the parameter α_0 is found from the condition of b_1 having as small magnitude as possible:

$$b_1^2 = (Ab_0 - \alpha_0 b_0)^2 = \min. \tag{1.9}$$

Differentiation and algebra yields

$$\alpha_0 = \frac{(Ab_0)b_0}{b_0^2}. \tag{1.10}$$

It is important to notice that the new b_1 vector is orthogonal to the original b_0 vector, i.e.,

$$b_1 b_0 = 0. \tag{1.11}$$

Continuing the process, one can find a b_2 vector by choosing the linear combination

$$b_2 = Ab_1 - \alpha_1 b_1 - \beta_0 b_0, \tag{1.12}$$

where once again the constants are defined by the fact that b_2^2 shall be minimal. Some algebraic work yields

$$\alpha_1 = \frac{(Ab_1)b_1}{b_1^2}, \qquad \beta_0 = \frac{(Ab_1)b_0}{b_0^2}. \tag{1.13}$$

Since

$$(Ab_1)b_0 = b_1(Ab_0) = b_1^2, \tag{1.14}$$

the new b_2 vector is orthogonal to both b_1 and b_0. Once more continuing the process, we need

$$b_3 = Ab_2 - \alpha_2 b_2 - \beta_1 b_1 - \gamma_0 b_0. \tag{1.15}$$

However, in view of the orthogonality of b_2 to both previous vectors, we get

$$\gamma_0 = \frac{(Ab_2)b_0}{b_0^2} = \frac{b_2(Ab_0)}{b_0^2} = 0. \tag{1.16}$$

The brilliant observation of Lanczos is that in every step of the iteration we will need only two correction terms: the famous three-member recurrence. The process established by Lanczos is now

$$\begin{aligned} b_0 &= \text{random}, \\ b_1 &= (A - \alpha_0)b_0, \\ b_2 &= (A - \alpha_1)b_1 - \beta_0 b_0, \\ b_3 &= (A - \alpha_2)b_2 - \beta_1 b_1, \\ &\cdot \\ b_m &= (A - \alpha_{m-1})b_{m-1} - \beta_{m-2}b_{m-2} = 0. \end{aligned}$$

The equality to zero on the mth recurrence equation means the end of the process. In Lanczos's view we reached the order of the minimum polynomial, where

$$m \leq n,$$

and n is the order of the matrix A. Unfortunately, in finite precision arithmetic, the process may reach a state where β_k is very small for $k < m$ before the full order of the minimum polynomial is obtained. This phenomenon, at the time not fully understood, contributed to the method's bad numerical reputation in the 1960s.

Lanczos then applied the method to unsymmetric matrices by the simultaneous execution of the process to A and its transpose, A^T. The connection of the two sets of vectors is maintained via inner products when calculating the denominators of the constants. This so-called biorthogonal process starts from random vectors b_0 and b_0^*. Please note that $b_0^* \neq b_0^H$; it is just another starting vector. The first step produces

$$b_1 = Ab_0 - \alpha_0 b_0, \tag{1.17}$$
$$b_1^* = A^T b_0^* - \alpha_0 b_0^*, \tag{1.18}$$

where the value of α_0 satisfying the biorthogonality conditions is

$$\alpha_0 = \frac{(Ab_0)b_0^*}{b_0 b_0^*} = \frac{(A^T b_0^*)b_0}{b_0^* b_0}. \tag{1.19}$$

The second step brings

$$b_2 = Ab_1 - \alpha_1 b_1 - \beta_0 b_0, \tag{1.20}$$
$$b_2^* = A^T b_1^* - \alpha_1 b_1^* - \beta_0 b_0^*, \tag{1.21}$$

where

$$\alpha_1 = \frac{(Ab_1)b_1^*}{b_1 b_1^*} = \frac{(A^T b_1^*)b_1}{b_1^* b_1} \tag{1.22}$$

and

$$\beta_0 = \frac{(Ab_1)b_0^*}{b_1 b_0^*} = \frac{(A^T b_1^*)b_0}{b_0^* b_0} = \frac{b_1^* b_1}{b_0^* b_0}. \tag{1.23}$$

The process now may be continued, leading to the following polynomials:

$$\begin{aligned} p_0 &= 1, \\ p_1(\mu) &= \mu - \alpha_0, \\ p_2(\mu) &= (\mu - \alpha_1)p_1(\mu) - \beta_0 p_0(\mu), \\ &\vdots \\ p_n(\mu) &= (\mu - \alpha_{n-1})p_{n-1}(\mu) - \beta_{n-2}p_{n-2}. \end{aligned}$$

For simplicity, let us assume now that all the polynomials until the nth may be obtained by this process (none of the β_i vanish). Then p_n is the characteristic polynomial of A with roots μ_i, $i = 1, 2, \ldots, n$.

1.3 Calculation of eigenvalues and eigenvectors

Lanczos used the characteristic polynomial developed above and the biorthogonality of the b_i, b_i^* sequence to find an explicit solution for the eigenvectors in terms of the b_i, b_i^* vectors. Assuming that A is of full rank, the b_i vectors may be expressed as linear combinations of the eigenvectors

$$b_i = p_i(\mu_1)u_1 + p_i(\mu_2)u_2 + \cdots + p_i(\mu_n)u_n. \tag{1.24}$$

Taking an inner product with u_k^* which is orthogonal to the u_i vectors, we get

$$b_i u_k^* = p_i(\mu_k)u_k u_k^*, \tag{1.25}$$

since all other inner products vanish. Please note again that $u_k^* \neq u_k^H$; they are just the kth members of the two simultaneous sequences. The reverse process of expressing eigenvectors in terms of the b_i vectors yields

$$u_i = \alpha_{i,0}b_0 + \alpha_{i,1}b_1 + \cdots + \alpha_{i,n-1}b_{n-1}. \tag{1.26}$$

Taking inner products again yields

$$u_i b_k^* = \alpha_{i,k} b_k b_k^* \tag{1.27}$$

or

$$\alpha_{i,k} = \frac{u_i b_k^*}{b_k b_k^*}. \tag{1.28}$$

Using this, the expansion for the eigenvectors becomes

$$u_i = \frac{b_0}{b_0 b_0^*} + p_1(\mu_i)\frac{b_1}{b_1 b_1^*} + \cdots + p_{n-1}(\mu_i)\frac{b_{n-1}}{b_{n-1}b_{n-1}^*}. \tag{1.29}$$

The adjoint, or left-handed, eigenvectors are calculated similarly:

$$u_i^* = \frac{b_0^*}{b_0 b_0^*} + p_1(\mu_i)\frac{b_1^*}{b_1 b_1^*} + \cdots + p_{n-1}(\mu_i)\frac{b_{n-1}^*}{b_{n-1} b_{n-1}^*}. \tag{1.30}$$

In the case of rank deficiency, the expansion is still valid for $m \leq n$. The shortcoming of this method is the repeated calculation and evaluation of the characteristic polynomial. Note that explicit formulation of the polynomial is not necessary. It is easy to see that premultiplying A by B^{*T} and postmultiplying the product by B yields

$$T = B^{*T} A B, \tag{1.31}$$

where

$$B = \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{bmatrix}, \tag{1.32}$$

$$B^* = \begin{bmatrix} b_0^* & b_1^* & \cdots & b_{n-1}^* \end{bmatrix}, \tag{1.33}$$

and

$$T = \begin{bmatrix} \alpha_0 & \beta_0 & & & & \\ \beta_0 & \alpha_1 & \beta_1 & & & \\ & & \cdot & \cdot & \cdot & \\ & & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{bmatrix}. \tag{1.34}$$

This observation allows a more efficient eigenvector calculation scheme. With appropriate pre- and postmultiplications of the problem (1.7),

$$B^{*T} A B B^{*T} u = \mu B^{*T} u, \tag{1.35}$$

and using the biorthogonality property of the b vectors,

$$B B^{*T} = I, \tag{1.36}$$

we get

$$Tv = \mu v, \tag{1.37}$$

where

$$v = B^{*T} u, \qquad u = Bv. \tag{1.38}$$

The latter two equations propose to calculate the v eigenvectors of the T tridiagonal matrix and calculate the eigenvectors of the original matrix with the multiplication by the Lanczos vectors, a process still in principal use.

1.4 Geometric interpretation

For a more insightful interpretation of the method, let us focus again on the symmetric case. Assume that we have found the first two Lanczos vectors b_0, b_1 and that they span the plane

$$S = \text{span}(b_0, b_1). \tag{1.39}$$

We seek the third Lanczos vector. The projection of the Ab_1 vector into the plane spanned by the first two vectors is

$$\text{proj}_S(Ab_1) = \frac{(Ab_1)b_0}{b_0 b_0} b_0 + \frac{(Ab_1)b_1}{b_1 b_1} b_1 \tag{1.40}$$

or

$$\text{proj}_S(Ab_1) = \frac{(Ab_1)b_0}{b_0^2} b_0 + \frac{(Ab_1)b_1}{b_1^2} b_1. \tag{1.41}$$

Based on (1.13)

$$\text{proj}_S(Ab_1) = \beta_0 b_0 + \alpha_1 b_1. \tag{1.42}$$

Choosing the third Lanczos vector as the component of Ab_1 orthogonal to the projection gives

$$b_2 = Ab_1 - \text{proj}_S(Ab_1) = Ab_1 - \alpha_1 b_1 - \beta_0 b_0, \tag{1.43}$$

which is identical to (1.12) and demonstrates the progression of the process.

In general, at every step of the Lanczos method a new b_{j+1} vector is found by projecting the Ab_j vector into the subspace spanned by the previous Lanczos vectors and choosing b_{j+1} to be the component of Ab_j orthogonal to the projection. Specifically

$$B_j = \begin{bmatrix} b_0 & b_1 & . & b_j \end{bmatrix}, \tag{1.44}$$

$$S_j = \text{span}(B_j), \tag{1.45}$$

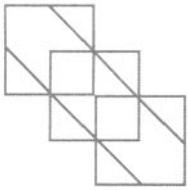
$$a = Ab_j, \tag{1.46}$$

$$p = \text{proj}_{S_j} a, \tag{1.47}$$

and

$$b_{j+1} = a - p. \tag{1.48}$$

The remaining chapters of Part I follow the evolution of the method toward today's state-of-the-art formulation used in the solution of industrial problems. We also abandon Lanczos's original notation, partially for mere convenience and also to adhere to the modern literature.



Chapter 2
The Lanczos method in exact arithmetic

In order to lay the foundation for the evolutionary discussion of the following chapters, let us first review the method in exact arithmetic. For the sake of a unified presentation we now restate the problem and present the method in a form more suitable for computations.

2.1 Computational formulation

The problem is to solve

$$A\underline{x} = \lambda \underline{x}, \qquad \underline{y}^T A = \lambda \underline{y}^T, \tag{2.1}$$

where A is real and unsymmetric. Here and in the remainder of the book $\underline{x}$, $\underline{y}$ are the eigenvectors of the problem; the underlining is used to distinguish them from the x, y Lanczos vectors.

The method generates two sets of biorthogonal vectors (the b_i and the b_i^* vectors of the classical method): X_n and Y_n such that

$$Y_n^T X_n = I, \tag{2.2}$$

where I is the identity matrix of order n and

$$Y_n^T A X_n = T_n, \tag{2.3}$$

where A is the original real, unsymmetric matrix and T_n is a tridiagonal matrix. By appropriate pre- and postmultiplications we get the following two equations:

$$AX_n = X_n T_n, \tag{2.4}$$
$$Y_n^T A = T_n Y_n^T. \tag{2.5}$$

By equating columns in these equations we get

$$Ax_k = \gamma_{k-1} x_{k-1} + \alpha_k x_k + \beta_k x_{k+1}, \tag{2.6}$$
$$y_k^T A = \beta_{k-1} y_{k-1}^T + \alpha_k y_k^T + \gamma_k y_{k+1}^T, \tag{2.7}$$

where $k = 1, 2, \ldots, n-1$, and y_k and x_k are the kth columns of Y_n and X_n. For any $k < n$ the following is also true:

$$AX_k = X_k T_k + \beta_k x_{k+1} e_k^T, \tag{2.8}$$

$$Y_k^T A = T_k Y_k^T + \gamma_k e_k^T y_{k+1}^T, \tag{2.9}$$

where e_k is the kth unit vector containing one in row k and zeros elsewhere. Its presence is needed only to make the matrix addition operation compatible. These equations will be very important in the error bound calculation. Note that in the above equations the transpose of the A matrix is not used, although the equations may also be presented in terms of A^T. The following starting assumptions are made:

$$\gamma_0 x_0 = 0 \tag{2.10}$$

and

$$\beta_0 y_0^T = 0. \tag{2.11}$$

The tridiagonal matrix of equations is then built from the Lanczos coefficients as follows:

$$T_n = \begin{bmatrix} \alpha_1 & \gamma_1 & & & & \\ \beta_1 & \alpha_2 & \gamma_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \beta_{n-2} & \alpha_{n-1} & \gamma_{n-1} & \\ & & & \beta_{n-1} & \alpha_n & \end{bmatrix}. \tag{2.12}$$

By reordering we obtain the following recurrence formulae:

$$\beta_k x_{k+1} = Ax_k - \alpha_k x_k - \gamma_{k-1} x_{k-1}, \tag{2.13}$$

$$\gamma_k y_{k+1}^T = y_k^T A - \alpha_k y_k^T - \beta_{k-1} y_{k-1}^T, \tag{2.14}$$

which is of course identical to the classical method. Unfortunately the coefficients β_k and γ_k are not uniquely defined. Any combination satisfying

$$\beta_k \gamma_k = \overline{y}_{k+1}^T \overline{x}_{k+1} \tag{2.15}$$

is acceptable, where

$$\overline{x}_{k+1} = \beta_k x_{k+1}, \tag{2.16}$$

$$\overline{y}_{k+1}^T = \gamma_k y_{k+1}^T. \tag{2.17}$$

Introducing the biorthogonality parameter

$$\delta_k = \overline{y}_{k+1}^T \overline{x}_{k+1}, \tag{2.18}$$

equation (2.15) can be satisfied with the following choices:

$$\beta_k = \sqrt{|\delta_k|}, \tag{2.19}$$

$$\gamma_k = \beta_k \operatorname{sign}(\delta_k). \tag{2.20}$$

The advantage of these choices is that, except for the sign, the resulting tridiagonal matrix will be symmetric.

In exact arithmetic every pair of Lanczos vectors satisfies

$$y_k^T x_k = 1 \tag{2.21}$$

and

$$y_k^T x_j = 0, \qquad y_j^T x_k = 0 \tag{2.22}$$

for any $j < k$. Based on this orthonormality condition, by premultiplying (2.6), we get

$$\alpha_k = y_k^T A x_k. \tag{2.23}$$

This is the equation for the remaining Lanczos coefficient.

2.2 Solving the tridiagonal problem

In exact arithmetic the Lanczos process produced the T_n matrix of (2.12). Now, we need a way to find the eigenvalues and eigenvectors of this tridiagonal matrix.

$$T_n u_i = \lambda_i u_i \tag{2.24}$$

and

$$v_i^T T_n = v_i^T \lambda_i. \tag{2.25}$$

The i in the above equations is the index of the eigenvalue of the tridiagonal matrix, $i = 1, 2, \ldots, n$.

Since the eigenvalues of A are invariant under the transformation to tridiagonal form, the λ_i eigenvalues of the tridiagonal matrix, the so-called Ritz values, are approximations of the λ eigenvalues of the original problem stated in (2.1). The approximations to the eigenvectors of the original problem (Ritz vectors) are calculated from the eigenvectors of the tridiagonal problem by the procedure originally suggested by Lanczos:

$$\underline{x}_i = X_n u_i, \qquad \underline{y}_i = Y_n v_i, \tag{2.26}$$

where X_n, Y_n are the matrices containing the Lanczos vectors, and u_i, v_i are the ith right and left eigenvectors of the tridiagonal matrix.

To solve these equations Francis's QR iteration [15] may be used, as well as other well-known methods, such as the QL method or the method of bisection in the symmetric case. The QR method is based on a decomposition of the T_n matrix into the form

$$T_n - \omega I = Q^1 R^1, \tag{2.27}$$

where R^1 is an upper triangular matrix and Q^1 contains orthogonal columns

$$Q^{1,T} Q^1 = I. \tag{2.28}$$

The presence of ω accounts for a diagonal shift to aid the stability of this decomposition. The process is performed by iterating as follows:

$$T_n^1 = R^1 Q^1 + \omega I. \tag{2.29}$$

From pre- and postmultiplying we get

$$T_n^2 = Q^{1,T} T^1 Q^1, \tag{2.30}$$

demonstrating the fact that the newly created matrix preserved the eigenvalue spectrum of the old one. By repeatedly applying this procedure, we finally obtain T_n^m in diagonal form, with elements that are the eigenvalues of the original matrix; that is

$$T_n^m = \lambda, \tag{2.31}$$

where m is the number of iterations—hopefully, but not necessarily, much less than n. The computation of Francis's QR iteration takes advantage of the tridiagonal nature of T_n. It creates orthogonal transformation matrices Q^i containing only the four terms of the Givens rotations that reduce the subdiagonal terms of R^i to zero. See [17] for details.

For the eigenvectors, an inverse power iteration procedure originally proposed by Wilkinson [41] will be used. The eigenvectors corresponding to the ith eigenvalue of the T_n tridiagonal matrix may be determined by the following factorization:

$$T_n - \lambda_i I = L_i U_i, \tag{2.32}$$

where L_i is unit lower triangular and U_i is upper triangular. Gaussian elimination with partial pivoting is used; i.e., the pivotal row at each stage is selected to be the equation with the largest coefficient of the variable being eliminated. Since the original matrix is tridiagonal, at each stage there are only two equations containing that variable. Approximate eigenvectors of the ith eigenvalue λ_i will be calculated by the simple (since the U_i also has only 3 codiagonals) iterative procedure

$$U_i u_i^{k+1} = u_i^k, \tag{2.33}$$

where u_i^k is random and k is a counter. Wilkinson proved that the convergence of this procedure is so rapid that k only goes to 2 or 3.

This original method also has some significant extensions for dealing with special cases such as multiple eigenvalues, which we will ignore for now. These last two steps of completing the eigenvalue problem are presented here only for the sake of completeness and will not be discussed in the remainder of the book.

There are several algorithms presented for the Lanczos algorithm in exact arithmetic in [11]. The following is a simple algorithm using only the foundation developed here.

2.3 Exact arithmetic algorithm

The following algorithm describes the method in exact arithmetic.

Algorithm 2.3.1. Exact arithmetic algorithm

$x_0 = y_0 = 0$
$\beta_0 = \gamma_0 = 0$
$y_1^T x_1 = 1$

For $k = 1, 2, \ldots, n$

$z_k = Ax_k$
$w_k^T = y_k^T A$
$\alpha_k = y_k^T z_k$
$z_k = z_k - \alpha_k x_k - \gamma_{k-1} x_{k-1}$
$w_k^T = w_k^T - \alpha_k y_k^T - \beta_{k-1} y_{k-1}^T$
$\delta_k = w_k^T z_k$
$\beta_k = \sqrt{|\delta_k|}$
$\gamma_k = \beta_k \text{sign}(\delta_k)$
$x_{k+1} = z_k / \beta_k$
$y_{k+1}^T = w_k^T / \gamma_k$

End loop k

Calculate u_i, v_i, $i = 1, 2, \ldots, n$, per (2.24) and (2.25)
Calculate $\underline{x}_i$, $\underline{y}_i$, $i = 1, 2, \ldots, n$, per (2.26)

The algorithm is expected to run through the matrix size of n.

2.4 Computational example

One of Lanczos's most important characteristics as a great expositor in his many textbooks was the use of well-selected simple examples. It is probably time now to follow that path and enlighten the reader with a computational example using the mathematics developed so far. Of course the process is tedious when carried out by hand, no matter how simple the example is. It is, however, easy to see that the method produces an orderly scheme amenable to the hand calculation techniques of the 1940s, and it is certainly just as well suited for computer implementations in the 2000s. Let us find the eigenvalues and eigenvectors of the following order 3 unsymmetric matrix:

$$A = \begin{bmatrix} 1/2 & 1/2 & -1/2 \\ 0 & 0 & -2 \\ 3/2 & -1/2 & 9/2 \end{bmatrix}. \tag{2.34}$$

We start the procedure with the following vectors:

$$x_1 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{2.35}$$

and

$$y_1 = \begin{bmatrix} 1/2 \\ -1/2 \\ -1/2 \end{bmatrix}, \tag{2.36}$$

which are not orthogonal, as they may even be random. The first intermediate vectors are

$$z_1 = \begin{bmatrix} 1 \\ 2 \\ -3 \end{bmatrix} = Ax_1 \tag{2.37}$$

and

$$v_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -3/2 \end{bmatrix} = A^T y_1. \tag{2.38}$$

The first Lanczos coefficient is

$$\alpha_1 = y_1^T z_1 = 1. \tag{2.39}$$

Since we are in the $k = 1$ loop and the coefficients with zero indices are zero, the update of the vectors follows as

$$z_1 = z_1 - \alpha_1 x_1 = \begin{bmatrix} 0 \\ 2 \\ -2 \end{bmatrix} \tag{2.40}$$

and

$$v_1 = v_1 - \alpha_1 y_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}. \tag{2.41}$$

The biorthogonality parameter is now calculated:

$$\delta_1 = v_1^T z_1 = 4. \tag{2.42}$$

Since it is positive, its square root becomes the basis for calculating the off-diagonal coefficients:

$$\beta_1 = \sqrt{\delta_1} = 2 = \gamma_1. \tag{2.43}$$

In turn, the normalized Lanczos vectors for the next iteration are

$$x_2 = z_1/\beta_1 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \tag{2.44}$$

and

$$y_2 = v_1/\gamma_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \end{bmatrix}. \tag{2.45}$$

The execution of the above steps for these new vectors is now left to the user. The computational results (without explanations) are

$$\alpha_2 = 3 \tag{2.46}$$

and

$$\beta_2 = 1. \tag{2.47}$$

The normalized Lanczos vectors of the final iteration are

$$x_3 = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} \tag{2.48}$$

and

$$y_3 = \begin{bmatrix} -1/2 \\ -1/2 \\ -1/2 \end{bmatrix}. \tag{2.49}$$

Finally the last diagonal coefficient will be

$$\alpha_3 = 1. \tag{2.50}$$

The resulting tridiagonal matrix is built as

$$T_3 = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{2.51}$$

The reader may verify that the collection of the x_i, y_i Lanczos vectors satisfies the $Y^T X = I$ orthogonality criterion, and the tridiagonal form may also be calculated by the formal process of $Y^T A X = T$.

Applying the methods of section 2.2, one can obtain one of the eigenvalues as an integer $\lambda_1 = 1$ and the corresponding right and left eigenvectors

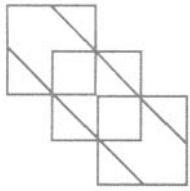
$$\underline{x}_1 = \begin{bmatrix} -3 \\ -2 \\ 1 \end{bmatrix} \tag{2.52}$$

and

$$\underline{y}_1 = \begin{bmatrix} -3 \\ -1 \\ -1 \end{bmatrix}, \tag{2.53}$$

for which the equations $A^T \underline{y}_1 = \lambda_1 \underline{y}_1$ and $A\underline{x}_1 = \lambda_1 \underline{x}_1$ are satisfied. The other two eigenvalues $(2 \pm \sqrt{6})$ and their eigenvectors are irrational.

This page intentionally left blank



Chapter 3

The Lanczos method in finite precision

With the first implementations of the Lanczos method on the computers of the 1950s, an undesirable numerical phenomenon was encountered. Due to the finite precision arithmetic, after some number of steps the orthogonality among the Lanczos vectors was lost. This phenomenon may be avoided by additional work to maintain the orthogonality. The method of monitoring the convergence and the assessment of the accuracy of the solution, as well as the most important issue of the breakdown of the process, are discussed in this chapter.

3.1 Breakdown of the Lanczos process

In finite precision arithmetics the recurrence formulae of (2.13) and (2.14) are subject to round-off error:

$$\beta_k x_{k+1} = Ax_k - \alpha_k x_k - \gamma_{k-1} x_{k-1} + f_x, \tag{3.1}$$

$$\gamma_k y_{k+1}^T = y_k^T A - \alpha_k y_k^T - \beta_{k-1} y_{k-1}^T + f_y, \tag{3.2}$$

where f_x, f_y are the round-off errors accumulated in the step. The round-off errors influence the δ_k biorthogonality parameter introduced in (2.18). This ultimately may result in a breakdown of the process when δ_k becomes or approaches zero. These cases are called exact- and near-breakdown, respectively. When

$$\delta_k = 0 \tag{3.3}$$

for any $k < n$, the cause may be either

$$y_{k+1}^T = 0 \quad \text{or} \quad x_{k+1} = 0. \tag{3.4}$$

This implies that either β_k or γ_k is zero and the process cannot continue (breaks down) due to a division by zero. Either case is called mild breakdown.

It is also possible that neither vector is zero; however, their scalar product is zero:

$$y_{k+1}^T x_{k+1} = 0. \tag{3.5}$$

This case is called the serious breakdown of the procedure by Wilkinson [41]. The interesting question of how far one should go with the Lanczos process is discussed in [23].

3.2 Measuring and maintaining orthogonality

The loss of orthogonality may be measured on various levels. The orthogonality within the recurrence step is called local orthogonality. This may be measured by

$$\frac{|x_{k+1}^T x_k|}{||x_{k+1}||\ ||x_k||} = \epsilon_x \tag{3.6}$$

and

$$\frac{|y_{k+1}^T y_k|}{||y_{k+1}||\ ||y_k||} = \epsilon_y, \tag{3.7}$$

respectively. To maintain orthogonality, Lanczos recommended a rather drastic remedy: orthogonalize each new Lanczos vector with respect to all previous Lanczos vectors. This process is called full orthogonalization.

This may be accomplished by the Gram–Schmidt process [18, pp. 150–152] as

$$x_{k+1}^i = x_{k+1}^{i-1} - \sum_{j=1}^{k} (x_j^T x_{k+1}^{i-1}) x_j, \tag{3.8}$$

$$y_{k+1}^i = y_{k+1}^{i-1} - \sum_{j=1}^{k} (y_j^T y_{k+1}^{i-1}) y_j, \tag{3.9}$$

where i is a repeat counter. In practical circumstances one to three repetitions of the process produce an acceptable level of computational (not exact) orthogonality between the vectors, resulting in $\epsilon_x < \epsilon$ and $\epsilon_y < \epsilon$, where ϵ is an appropriately small number.

The level of biorthogonality between the vectors is now measured by

$$\frac{|y_{k+1}^T x_{k+1}|}{||y_{k+1}||\ ||x_{k+1}||} = \epsilon_{xy}. \tag{3.10}$$

The biorthogonality between the two sets of vectors x_j, y_j may be assured by the following variation of the Gram–Schmidt process:

$$x_{k+1}^i = x_{k+1}^{i-1} - \sum_{j=1}^{k} (y_j^T x_{k+1}^{i-1}) x_j \tag{3.11}$$

and

$$y_{k+1}^i = y_{k+1}^{i-1} - \sum_{j=1}^{k} (x_j^T y_{k+1}^{i-1}) y_j. \tag{3.12}$$

The biorthonormality between the x_k and y_k vectors may be assured by the following normalization:

$$x_{k+1} = \frac{x_{k+1}}{\sqrt{y_{k+1}^T x_{k+1}}} \tag{3.13}$$

and

$$y_{k+1} = \frac{y_{k+1}}{\sqrt{x_{k+1}^T y_{k+1}}}. \tag{3.14}$$

While the full orthogonalization solution is adequate, it has such a high cost that it renders the method impractical for large problems. In essence the cost of every Lanczos step increases with the number of steps. As mentioned in the preface, this fact almost doomed the method until the significant work of Paige [32]. Paige recognized the importance of looking at the Ritz vector basis and specifically the angle between the next Lanczos vector and the previous Ritz vectors.

This view shows that the seemingly gradual loss of orthogonality among all the Lanczos vectors is not the same in the Ritz vector basis. The Ritz vector basis revealed that in a case of an apparent complete loss of orthogonality among the Lanczos vectors, the Ritz vectors remained orthogonal except for the last two Ritz vectors, which were parallel. This meant that the range of the X_k had only dimension $k - 1$.

This may be interpreted as a welcome event, that is, that the X_{k-1} is a decoupled invariant subspace. The eigenvalues extracted from that subspace are exact eigenvalues of the original problem as well. On the other hand, continuing the process requires further considerations, such as implicit deflation or external reorthogonalization [11].

Paige's work originated several techniques of maintaining orthogonality to different levels. These techniques can postpone, prevent, and sometimes even predict the upcoming loss of orthogonality. In the more advanced practical implementations (such as the real symmetric block method discussed in Chapter 4) several of these methods are used.

One of them is called the selective orthogonalization strategy [33], which proposes to orthogonalize against some, but not all, of the Ritz vectors at certain steps. Another successful strategy is the partial orthogonalization method, sometimes also called semiorthogonalization method [39], which executes the orthogonalization of the Lanczos vectors only to the square root of machine epsilon. The machine epsilon, $\epsilon_{\text{machine}}$, is the smallest value on a given machine such that $1 + \epsilon_{\text{machine}}$ is greater than 1.

The most recent of these strategies, the adaptive logic method [9], monitors the loss of orthogonality and adapts (increases or decreases) the number of Lanczos vectors (block size) accordingly. This method will be detailed further in Chapter 5.

There are a couple of other methods present in the literature that are worth mentioning here: the look-ahead algorithm [36] and the s-step method [26]. The first method tries to avoid the breakdown by "looking ahead" and introducing intermediate steps to correct the next regular step in advance. The second method precomputes more than one step at a time and modifies steps prior to the anticipated breakdown step. These methods have not been adopted yet in the industry mainstream and need more research with respect to their applicability to real-life problems.

3.3 Monitoring convergence and estimating accuracy

The approximated residual error in the original solution following Parlett's excellent treatise for symmetric matrices [34] can be calculated as

$$||r_x|| = ||A\underline{x} - \lambda_i \underline{x}|| = ||AX_j u_i - \lambda_i X_j u_i|| = ||(AX_j - \lambda X_j)u_i|| \tag{3.15}$$

where $i = 1, 2, \ldots, j$, and j is the number of successful Lanczos steps executed. Furthermore, following (2.8) and (2.9),

$$||r_x|| = ||(AX_j - X_jT_j)u_i|| = ||(\beta_j x_{j+1} e_j^T)u_i|| = \beta_j e_j^T ||u_i||, \tag{3.16}$$

assuming the norm of the Lanczos vector x_{j+1} is unity. All of the above norms are Euclidean norms. Taking advantage of the structure of the unit vector, we can simplify (3.16) into the following scalar form:

$$||r_x|| = \beta_j |u_{ji}|, \tag{3.17}$$

where u_{ji} is the jth (last) term in the u_i eigenvector. A similar condition can be calculated for the left-handed vectors:

$$||r_y|| = \gamma_j |v_{ji}|, \tag{3.18}$$

where v_{ji} is the jth (last) term in the v_i eigenvector.

The last two equations give a convergence monitoring tool. When these error norms are less than the required tolerance ϵ and the value of j is higher than the number of roots required, the process can be stopped. It is the beauty of this convergence criterion that only the eigenvector of the tridiagonal problem must be found, which is inexpensive compared to finding the eigenvector of the physical (size n) problem.

Let us introduce ρ as the largest of the residual norms calculated above:

$$\rho = \max(||r_x||, ||r_y||). \tag{3.19}$$

This enables a computationally cheap acceptance test of $\rho < \epsilon$ to be executed frequently in evaluating the approximate solutions.

3.4 Finite precision algorithm

This algorithm is similar to the exact arithmetic algorithm with appropriate adjustments to monitor the breakdown of the process. In case the biorthogonality parameter dips below a certain threshold, an orthogonalization step is executed. The algorithm also contains an estimate of the errors in the solution.

Algorithm 3.4.1. Finite precision algorithm

$x_0 = y_0 = 0$
$\beta_0 = \gamma_0 = 0$
$y_1^T x_1 = 1$

For $k = 1, 2, \ldots$ until convergence

$z_k = Ax_k$
$v_k^T = y_k^T A$
$\alpha_k = y_k^T z_k$
$z_k = z_k - \alpha_k x_k - \gamma_{k-1} x_{k-1}$
$v_k^T = v_k^T - \alpha_k y_k^T - \beta_{k-1} y_{k-1}^T$
$\delta_k = v_k^T z_k$

If $|\delta_k| \leq \delta_{\text{threshold}}$ then
Execute orthogonalization per (3.8), (3.9), (3.11), (3.12)
Endif
$\beta_k = \sqrt{|\delta_k|}$
$\gamma_k = \beta_k \text{sign}(\delta_k)$
$x_{k+1} = z_k / \beta_k$
$y_{k+1}^T = v_k^T / \gamma_k$
$j = k$

End loop k

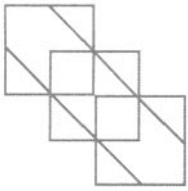
For $i = 1, 2, \ldots, j$

Calculate u_i, v_i
$r_{xi} = \beta_j |u_{ji}|$
$r_{yi} = \gamma_j |u_{ji}|$
$\rho_i = \max(||r_{x_i}||, ||r_{y_j}||)$
If $\rho_i < \epsilon_{\text{acceptance}}$ then
$\underline{x}_i = X_j u_i$
$\underline{y}_i = Y_j v_i$
Endif

End loop i

The $\delta_{\text{threshold}}$ is the biorthogonality tolerance, usually related to the machine epsilon defined earlier. The $\epsilon_{\text{acceptance}}$ criterion is used in the decision whether the computation of the actual eigenvectors is needed or not (the corresponding eigenvalue is accepted or not). The acceptance criterion is less strict than the convergence criterion, and it is usually established based on the physical problem being solved.

This page intentionally left blank



Chapter 4
Block real symmetric Lanczos method

The simple (single-vector) Lanczos method has difficulties in finding multiple eigenvalues. In fact, in exact arithmetic, the algorithm can find only one eigenvector of a multiple eigenvalue. In finite precision arithmetic, due to round-off errors, additional copies of the multiple eigenvalues will converge, usually several steps after the first copy's appearance.

The block methodology introduced by Golub in [19], working with multiple Lanczos vectors at a time, results in accurate calculation of multiple eigenvalues. This frequently occurs in industrial applications due to structural symmetry.

The block idea is also important as a computational efficiency tool, especially with respect to the access of the A matrix when it is held in secondary storage, as is done frequently in practical applications. When the Lanczos steps are executed in blocks, the A matrix needs to be accessed only once for every block, as opposed to once for every vector. If the number of vectors in a block is b, the associated I/O reduction is b-fold.

4.1 The block Lanczos method

The block method carries out the Lanczos recurrence with several vectors simultaneously. In describing this method, let us concentrate on the case of real, symmetric matrices only. First, these matrices are very important in many facets of engineering, most notably in the vibration of structures. Second, some elements of the methodology developed here carry over to unsymmetric and complex matrices also. In this and the following chapter we will use notation more prevalent in the modern literature on block methods [20].

The recurrence algorithm is now executed using blocks of vectors and only up to a certain number of (say j) steps:

$$R_{i+1} = AQ_i - Q_i A_i - Q_{i-1} B_i^T, \quad i = 1, 2, \ldots, j, \tag{4.1}$$

where

$$A_i = Q_i^T A Q_i \tag{4.2}$$

and

$$R_{i+1} = Q_{i+1} B_{i+1}. \tag{4.3}$$

Q_{i+1} is an $n \times b$ matrix with orthonormal columns (the Lanczos vectors earlier noted as x_i) and B_{i+1} is a $b \times b$ upper triangular matrix, n being the problem size and b the block size, obtainable by the QR decomposition (see, for example, [18] for details).

The initial values are $Q_0 = 0$, and R_0 is a collection of b pseudorandom vectors at the start. This process results in a block tridiagonal matrix of the form

$$T_j = \begin{bmatrix} A_1 & B_2^T & & & \\ B_2 & A_2 & B_3^T & & \\ & \cdot & \cdot & \cdot & \\ & & & B_j & A_j \end{bmatrix}. \tag{4.4}$$

Note that, in agreement with the mainstream literature, here and in Chapter 5 the first subdiagonal term is indexed by 2 (B_2) as opposed to the 1 (β_1) used in the scalar methods in earlier chapters. With appropriately chosen Givens transformations, this tridiagonal matrix is reduced into a scalar tridiagonal matrix T_J. The solution of the size $J = jb$ ($J << n$) eigenvalue problem of

$$T_J u = \lambda u$$

may again be found by the previously mentioned QR iteration algorithm. The eigenvalues of the original large problem are again invariant under the transformations, resulting in the tridiagonal form. For the symmetric case considered here, an eigenvalue error estimation [34] is available. The approximated error in the eigenvalue is

$$|\lambda_i - \underline{\lambda}_i| = |r_i|/\delta. \tag{4.5}$$

Here $\underline{\lambda}$ is the "exact" eigenvalue. The r_i is the vector residual norms calculated based on the Jth term of the ith eigenvector as (see section 3.3 for the derivation)

$$r_i = \beta_J |u_{Ji}|, \tag{4.6}$$

and the δ is the minimum gap between the eigenvalues:

$$\delta = \min |\lambda_i - \lambda_k| \tag{4.7}$$

for all $k \neq i$. Again, to find the eigenvectors of the original problem, a back-transformation of the form

$$\underline{x} = Q_J u$$

is required. The Q_J matrix is a collection of the Lanczos vector blocks:

$$Q_J = \begin{bmatrix} Q_1 & Q_2 & \cdot & \cdot & Q_j \end{bmatrix}.$$

This process may then be repeated until all eigenvalues (and corresponding eigenvectors) of the required frequency range are found.

There are several important issues to be addressed in the block method. One is maintaining orthogonality among and within the blocks. Another is the reduction of the block tridiagonal form to a scalar tridiagonal form with the help of Givens rotations. These issues may call for a few words of explanation.

4.2 Measuring and maintaining block orthogonality

As previously stated, the Lanczos algorithm produces an orthonormal set of vectors in exact arithmetic. The finite arithmetic execution of the algorithm will produce round-off errors, which cause the Lanczos vectors to lose their orthogonality. Maintenance of orthogonality is just as crucial for the convergence of the block Lanczos algorithm. Practical use of the Lanczos method on finite precision computers for large problems renders Lanczos's original idea of full reorthogonalization very costly.

The orthogonalization scheme for the block method is based on the recognition of the fact that loss of orthogonality may occur in three different respects:

1. within the current block of Lanczos vectors;
2. with respect to the previous two blocks of Lanczos vectors;
3. with respect to all previously computed Lanczos vectors.

These three classes are called internal, local, and global, respectively.

Internal loss of orthogonality within a block may happen when the vectors of an R_{i+1} block are close to being linearly dependent. Proper implementation of the QR decomposition step may automatically prevent this problem by producing good orthogonality between the Q_{i+1} vectors.

The loss of orthogonality known as local manifests itself among R_{i+1}, Q_i, and Q_{i-1}. These blocks are orthogonal by the nature of the recurrence in exact arithmetic, but they are not in finite precision. This difficulty may easily be overcome by an explicit orthogonalization of R_{i+1} against Q_i and Q_{i-1} to an acceptable precision using a few steps of Gram–Schmidt orthogonalization. This may be done for full machine precision, as this is inexpensive, or by using the semiorthogonalization principle mentioned in section 3.2.

Finally, the global loss of orthogonality between Q_{i+1} and the previous Lanczos blocks is the most serious. In order to assess the global loss of orthogonality, let us reconsider (4.1) in finite precision by incorporating (4.3). Then

$$Q_{i+1}B_{i+1} = AQ_i - Q_iA_i - Q_{i-1}B_i^T + F_i, \tag{4.8}$$

where F_i is the round-off error of this step. Premultiplying this by any block Q_k^T ($k = 1, \ldots, i-2$) that is not involved in the current recurrence step yields

$$Q_k^T Q_{i+1}B_{i+1} = Q_k^T AQ_i - Q_k^T Q_iA_i - Q_k^T Q_{i-1}B_i^T + Q_k^T F_i. \tag{4.9}$$

Switching indices k and i gives

$$Q_i^T Q_{k+1}B_{k+1} = Q_i^T AQ_k - Q_i^T Q_kA_k - Q_i^T Q_{k-1}B_k^T + Q_i^T F_k. \tag{4.10}$$

Subtracting the last two equations and using the fact that for the symmetric case considered here

$$Q_k^T AQ_i = Q_i^T AQ_k, \tag{4.11}$$

we get

$$\begin{aligned} Q_k^T Q_{i+1}B_{i+1} &= Q_i^T Q_{k+1}B_{k+1} \\ &+ Q_i^T Q_kA_k + Q_i^T Q_{k-1}B_k^T - Q_i^T F_k - Q_k^T Q_iA_i - Q_k^T Q_{i-1}B_i^T + Q_k^T F_i. \end{aligned} \tag{4.12}$$

Introducing the notation $W_{j,k} = Q_k^T Q_j$, reordering, and performing some algebra yields

$$W_{i+1,k}B_{i+1} = B_{k+1}^T W_{i,k+1} + A_k W_{i,k} + B_k W_{i,k-1} - W_{i,k}A_i - W_{i-1,k}B_i^T + G_{i,k}, \tag{4.13}$$

where $G_{i,k} = Q_i^T F_k - Q_k^T F_i$ represents the local round-off error. The $W_{i+1,k}$ term represents the global loss of orthogonality dependent on the losses having already occurred at previous steps that are contained in the other $\{W\}$ terms on the right-hand side of the equation. A bound for the global loss is measured by

$$||W_{i+1,k}|| \le ||B_{i+1}^{-1}|| \Big(||B_{k+1}^T|| \, ||W_{i,k+1}|| + ||B_k|| \, ||W_{i,k-1}|| + ||B_i|| \, ||W_{i-1,k}|| + (||A_k|| + ||A_i||)||W_{i,k}|| + ||G_{i,k}|| \Big), \tag{4.14}$$

where the right-hand side terms are available from earlier steps of the recurrence. The above bound is easily computed at each step, and a decision may be made on the need for extra orthogonalization steps. For more details on this issue, including an algorithm to monitor the loss of orthogonality, see [20]; for the unsymmetric generalization, see [12]. In case there is a need for extra (also called external) orthogonalization steps, the selective technique mentioned in section 3.2 is recommended.

4.3 Reduction of block tridiagonal form

Before we can turn the problem over to the tridiagonal solution phase, the block tridiagonal T_j matrix must be transformed to a scalar tridiagonal T_J form. The problem is to eliminate terms in the subdiagonal blocks B_i and in the diagonal blocks A_i below the subdiagonal. The method of Householder [22] or Givens [17] is well suited for this.

The Givens method uses a series of rotation matrices G_i such that

$$G_r G_{r-1} \ldots G_1 T_j G_1^T G_2^T \ldots G_r^T = T_J, \tag{4.15}$$

where $J = jb$, b being the block size, and r is the number of nonzero terms in the blocks below the subdiagonal. In order to eliminate all unwanted terms below the subdiagonal, this process is executed in a systematic fashion. For example, in every subdiagonal block B_i we need $b-1, b-2, \ldots, 1$ transformations starting from the first column to the penultimate column. The last column of B_i has only a subdiagonal term. Similar considerations apply to the A_i blocks. The collection (aggregate) of the Givens rotations is noted as G_H in the algorithm below.

4.4 Block real symmetric algorithm

The execution of the block Lanczos method is presented in this section.

Algorithm 4.4.1. Block symmetric algorithm

1. Initialize:
   a. $R_0 = \text{random}, n \times b$
   b. $Q_0 = 0, n \times b$
   c. $R_1 = AR_0$
   d. $R_1 = Q_1 B_1$

2. For $i = 1, 2, \ldots, j$

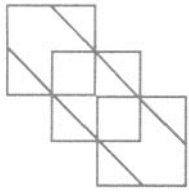
   a. $A_i = Q_i^T Q_i$
   b. $R_{i+1} = AQ_i - Q_i A_i - Q_{i-1} B_i^T$
   c. $R_{i+1} = Q_{i+1} B_{i+1}$
   d. Maintain local orthogonality: $Q_{i+1}^T Q_j \leq \epsilon_{\text{machine}}, j = i, i-1$

End loop i

3. Solve:
   a. Form T_j
   b. $T_J = G_H^T T_j G_H$
   c. $T_J z = \lambda z$
   d. $\underline{x} = Q_J z$

The above algorithm is somewhat simplified, as the details of the global orthogonalization process are omitted.

This page intentionally left blank



Chapter 5
Block unsymmetric Lanczos method

The Lanczos method in finite precision (Chapter 3) and the real, symmetric block version (Chapter 4) are the foundation for the block unsymmetric process discussed here.

5.1 Block biorthogonal Lanczos method

The block biorthogonal Lanczos method (Bai [9]) generates two sets of biorthonormal blocks of vectors P_i and Q_j such that

$$P_i^H Q_j = I \tag{5.1}$$

when $i = j$ $(i, j \leq n)$, and zero otherwise. Note that ^{H} denotes the complex conjugate transpose. These vector sets reduce the A matrix to a T_j block tridiagonal matrix form:

$$T_j = \overline{P}_j^H A \overline{Q}_j, \tag{5.2}$$

where the

$$\overline{P}_j = [P_1, P_2, \ldots, P_j] \tag{5.3}$$

and

$$\overline{Q}_j = [Q_1, Q_2, \ldots, Q_j] \tag{5.4}$$

matrices are the collections of the Lanczos blocks. The structure of the tridiagonal matrix is

$$T_j = \begin{bmatrix} A_1 & B_2 & & & \\ C_2 & A_2 & . & & \\ & . & . & . & \\ & & . & . & B_j \\ & & & C_j & A_j \end{bmatrix}. \tag{5.5}$$

The block Lanczos process is manifested in the following three-term recurrence matrix equations:

$$B_{j+1} P_{j+1}^H = P_j^H A - A_j P_j^H - C_j P_{j-1}^H \tag{5.6}$$

and

$$Q_{j+1}C_{j+1} = AQ_j - Q_jA_j - Q_{j-1}B_j. \tag{5.7}$$

Note that in both of these equations the transpose of the matrix A is again avoided.

5.2 Solution of the block tridiagonal problem

In order to find the mathematical eigenvalues and eigenvectors, we solve the block tridiagonal eigenvalue problems posed as

$$w^H T_j = \theta w^H \tag{5.8}$$

and

$$T_j z = \theta z, \tag{5.9}$$

where in the above equations the size of the reduced tridiagonal matrix T_j is $jp \times jp$, assuming a fixed block size p for now. The θ eigenvalues of the tridiagonal problem are approximations of the λ eigenvalues of the mathematical problem. The approximations to the eigenvectors of the original problem are calculated from the eigenvectors of the tridiagonal problem by

$$\underline{y} = \overline{P}_j w \tag{5.10}$$

and

$$\underline{x} = \overline{Q}_j z, \tag{5.11}$$

where again $\overline{P}_j$, $\overline{Q}_j$ are the matrices containing the first j Lanczos blocks of vectors, and w, z are the left and right eigenvectors of the tridiagonal problem. Finally $\underline{x}$, $\underline{y}$ are the right and left approximated eigenvectors of the original eigenvalue problem.

5.3 Error analysis

A useful aspect of the Lanczos method exploited earlier is that the error norm of the original problem may be calculated from the tridiagonal solution without calculating the eigenvectors. For a similar arrangement, let us introduce a rectangular $n \times p$ matrix E_j having an identity matrix as bottom square. Using this, a residual vector for the left-hand solution is

$$s^H = \underline{y}^H A - \theta \underline{y}^H = (w^H E_j) B_{j+1} P_{j+1}^H, \tag{5.12}$$

which means that only the bottom p (if the current block size is p) terms of the new eigenvector w are required (due to the structure of E_j). Similarly for the right handed vectors,

$$r = A\underline{x} - \theta \underline{x} = Q_{j+1}C_{j+1}(E_j^H z). \tag{5.13}$$

An easy-to-compute acceptance criterion (an extension of the one used in Chapter 3) may be based on the norm of the above residual vectors as

$$\min\left(\frac{||s^H||}{||\underline{y}^H||}, \frac{||r||}{||\underline{y}||}\right) \leq \epsilon_{\text{acceptance}}, \tag{5.14}$$

where the $\epsilon_{\text{acceptance}}$ value to accept convergence is again given based on the physical problem.

Based on a detailed error analysis of these quantities, we can modify this criterion by considering

$$\text{gap}(\theta, T_j) = \min |\theta - \theta_i|, \tag{5.15}$$

where $\theta_i \neq \theta$. With this the recommended criterion is

$$\min\left(||s^H||, ||r||, \frac{||s^H||, ||r||}{\text{gap}(\theta, T_j)}\right) \leq \epsilon_{\text{acceptance}}. \tag{5.16}$$

The $||.||$ denotes the Euclidean norm.

5.4 Block unsymmetric algorithm

The following is a block unsymmetric Lanczos algorithm.

Algorithm 5.4.1. Block unsymmetric algorithm

1. Initialize:
   a. Choose starting vector blocks P_1, Q_1 such that $P_1^H Q_1 = I$
   b. Calculate $R_1 = (P_1^H A)^H$ and $S_1 = A Q_1$

2. For $j = 1, 2, \ldots$

   a. Compute recurrence:
   $A_j = P_j^H S_j$
   $R_j = R_j - P_j A_j^H$
   $S_j = S_j - Q_j A_j$

   b. QR decompose:
   $R_j = P_{j+1} B_{j+1}^H$
   $S_j = Q_{j+1} C_{j+1}$

   c. SV decompose:
   $P_{j+1}^H Q_{j+1} = U_j \Sigma_j V_j^H$

   d. Update:
   $B_{j+1} = B_{j+1} U_j \Sigma_j^{1/2}$
   $C_{j+1} = \Sigma_j^{1/2} V_j^H C_{j+1}$
   $P_{j+1} = P_{j+1} U_j \Sigma_j^{-1/2}$
   $Q_{j+1} = Q_{j+1} V_j \Sigma_j^{-1/2}$

   e. Start recurrence:
   $R_{j+1} = (P_{j+1}^H A - C_{j+1} P_j^H)^H$
   $S_{j+1} = A Q_{j+1} - Q_j B_{j+1}$

End loop j

Details such as when and how to stop this algorithm will be further discussed in the following sections.

5.5 Adapting the block size

We have assumed that the Lanczos blocks have uniform size of p. It is possible to generalize this to allow for the jth iteration to have p_j variable block size. Such flexibility may be advantageous in the case of clustered eigenvalues or to avoid the breakdown of the Lanczos process.

Let us assume that at the $(j+1)$st iteration the block size is increased by k and the $(j+1)$st Lanczos vectors are augmented as

$$\begin{bmatrix} P_{j+1} & \underline{P}_{j+1} \end{bmatrix} \tag{5.17}$$

and

$$\begin{bmatrix} Q_{j+1} & \underline{Q}_{j+1} \end{bmatrix}, \tag{5.18}$$

where the $\underline{P}_{j+1}$, $\underline{Q}_{j+1}$ vectors are the yet-undefined augmentations. It is easy to see that appropriate augmentations will maintain the validity of the three-member recurrence as follows:

$$\begin{bmatrix} B_{j+1} & 0 \end{bmatrix} \begin{bmatrix} P_{j+1}^H \\ \underline{P}_{j+1}^H \end{bmatrix} = P_j^H A - A_j P_j^H - C_j P_{j-1}^H \tag{5.19}$$

and

$$\begin{bmatrix} Q_{j+1} & \underline{Q}_{j+1} \end{bmatrix} \begin{bmatrix} C_{j+1} \\ 0 \end{bmatrix} = AQ_j - Q_j A_j - Q_{j-1} B_j. \tag{5.20}$$

The Lanczos process can therefore formally continue with

$$B_{j+1} = \begin{bmatrix} B_{j+1} & 0 \end{bmatrix}, \tag{5.21}$$

$$C_{j+1} = \begin{bmatrix} C_{j+1} \\ 0 \end{bmatrix}, \tag{5.22}$$

and

$$P_{j+1} = \begin{bmatrix} P_{j+1} & \underline{P}_{j+1} \end{bmatrix}, \tag{5.23}$$

$$Q_{j+1} = \begin{bmatrix} Q_{j+1} & \underline{Q}_{j+1} \end{bmatrix}. \tag{5.24}$$

The conditions of successful continuation with augmented blocks are the orthogonality requirements of

$$\underline{P}_{j+1}^H \overline{Q}_j = 0 \tag{5.25}$$

and

$$\overline{P}_j^H \underline{Q}_{j+1} = 0. \tag{5.26}$$

It is of course necessary that the inner product $P_{j+1}^H Q_{j+1}$ of the newly created augmented pair of Lanczos vector blocks is not singular, since its decomposition will be needed by the algorithm. Specifically, we need the smallest singular values of the inner product matrix to be larger than a certain small number.

A possible choice for the augmentations is to have k pairs of random vectors and orthogonalize them against the earlier vectors by using a modified Gram–Schmidt procedure. The augmentation may be repeated several times to ensure that the smallest singular value is above the threshold.

The most prevalent usage of the block size adaptation is to cover existing clusters of eigenvalues. The multiplicity of a cluster may be found on the fly with the help of the Ritz values. The number of expected multiplicities in a cluster is the number of elements of the set satisfying

$$|\theta_i - \theta_k| \leq \epsilon_{\text{clu}} \max(|\theta_i|, |\theta_k|), \tag{5.27}$$

where ϵ_{clu} is a specified cluster threshold. The order of the largest cluster of the Ritz values may be calculated every time a convergence test is made, and the block size may be appropriately adjusted. This procedure is not very expensive, since it is done on the tridiagonal problem.

5.6 Preventing breakdown

It is easy to see that the algorithm presented in section 5.4 also breaks down in some circumstances:

I. One or both of R_j and S_j are rank deficient.

II. Neither is rank deficient, but $R_j^H S_j$ is rank deficient.

The breakdown of the first kind (mild breakdown) prevents the execution of step 2b of the algorithm, the QR decomposition of the jth blocks. This is fairly easy to overcome by an orthogonalization procedure. Specifically, if S_j is deficient, then we restart the Lanczos process with a Q_{j+1} augmented by random vectors made orthogonal to all previous left Lanczos vectors $\overline{P}_j$ as

$$\overline{P}_j^H Q_{j+1} = 0. \tag{5.28}$$

If S_j is just nearly rank deficient (detected by the QR decomposition of $S_j = Q_{j+1} C_{j+1}$), then we reorthogonalize this Q_{j+1} to the previous left Lanczos vectors, as shown in the above equation.

Rank deficiency (full or near) of R_j is treated similarly with respect to the right Lanczos vectors.

The breakdown of the second kind (serious breakdown) manifests itself in step 2c of Algorithm 5.4.1, the singular value decomposition. In the case of this breakdown some or all of the singular values are zero, as follows:

$$P_{j+1}^H Q_{j+1} = U_j \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V_j^H, \tag{5.29}$$

where Σ is nonsingular if it exists. Using the augmentation techniques shown in section 5.5, we may also overcome this problem. First calculate and partition as follows:

$$P_{j+1}U_j = \begin{bmatrix} P_{(1)} & P_{(2)} \end{bmatrix} \tag{5.30}$$

and

$$Q_{j+1}V_j = \begin{bmatrix} Q_{(1)} & Q_{(2)} \end{bmatrix}, \tag{5.31}$$

where the number of columns in the second partition is the number of zero singular values. Create the following projector matrix:

$$\Pi_j = \overline{Q}_j \overline{P}_j^H, \tag{5.32}$$

where $\overline{Q}_j$ are the eigenvectors that have already been accepted. Bai [9] proves that the vectors of

$$\underline{P}_{(2)} = (I - \Pi_j)^H Q_{(2)} \tag{5.33}$$

and

$$\underline{Q}_{(2)} = (I - \Pi_j)^H P_{(2)} \tag{5.34}$$

in the following augmentation of

$$P_{j+1} = \begin{bmatrix} P_{(1)} & P_{(2)} & \underline{P}_{(2)} \end{bmatrix} \tag{5.35}$$

and

$$Q_{j+1} = \begin{bmatrix} Q_{(1)} & Q_{(2)} & \underline{Q}_{(2)} \end{bmatrix} \tag{5.36}$$

will result in an always nonsingular $P_{j+1}^H Q_{j+1}$ product.

It is possible to extend this procedure to the case of near-breakdown, when the singular values may not be exact zeros, but smaller than desired. In this case it is recommended to increase the block size for those singular values which are below a specified threshold.

5.7 Maintaining biorthonormality

The maintenance of the biorthonormality of the P_j and Q_j vectors is the cornerstone of the Lanczos algorithm. Local biorthonormality, that is, maintaining the condition between the consecutive Lanczos vectors, is fairly simple by executing the following steps after 2b in Algorithm 5.4.1:

$$R_j = R_j - P_j(Q_j^H R_j) \tag{5.37}$$

and

$$S_j = S_j - Q_j(P_j^H S_j). \tag{5.38}$$

These steps use only data available in memory and therefore are cost effective even when executed repeatedly.

This method, however, unfortunately does not ensure that multiple eigenvalues will not reoccur. This may be prevented by a full reorthonormalization scheme using a modified Gram–Schmidt process described earlier. A measure of the orthogonality of the current Lanczos vectors with respect to the already-accepted eigenvectors is

$$d_{j+1} = \max\left(\frac{||\overline{P}_j^H Q_{j+1}||_1}{||\overline{P}_j||_1||Q_{j+1}||_1}, \frac{||\overline{Q}_j^H P_{j+1}||_1}{||\overline{Q}_j||_1||P_{j+1}||_1}\right), \tag{5.39}$$

where $||.||_1$ is the matrix column norm. This quantity is usually compared to a machine computational accuracy indicator as

$$d_{j+1} \leq \sqrt{\epsilon_{\text{machine}}}, \tag{5.40}$$

where $\epsilon_{\text{machine}}$ is the smallest value on a given machine such that $1+\epsilon_{\text{machine}}$ is greater than 1, as defined earlier. The appropriateness of the choice of the square root for the symmetric case was proven by Simon (called partial orthogonality there; see [39]). The generalization of this is presented in [12].

That measure, however, is very expensive with regards to both CPU and I/O. Specifically, the numerator requires the retrieval of the $\overline{P}_j$ and $\overline{Q}_j$ vector blocks from secondary storage and a multiplication by them. A method of maintaining partial orthogonality with a limited access of the $\overline{P}_j$ and $\overline{Q}_j$ matrices uses

$$\overline{d}_{j+1} = \max\left(\frac{||X_{j+1}||_1}{||\overline{P}_j||_1||Q_{j+1}||_1}, \frac{||Y_{j+1}||_\infty}{||\overline{Q}_j||_1||P_{j+1}||_1}\right), \tag{5.41}$$

where $||.||_\infty$ is now the matrix row norm. The norms of the denominator may be updated in every iteration step without retrieving the eigenvectors. This method involves calculating the numerator terms of

$$X_{j+1} = \overline{P}_j Q_{j+1} \tag{5.42}$$

and

$$Y_{j+1} = P_{j+1}\overline{Q}_j. \tag{5.43}$$

It also involves utilizing the fact that these also satisfy the three-term recurrence equations perturbed by rounding errors, as follows:

$$X_{j+1} = T_j \begin{bmatrix} X_j \\ 0 \end{bmatrix} - \begin{bmatrix} X_j \\ 0 \end{bmatrix} A_j - \begin{bmatrix} X_{j-1} \\ W_1^l \end{bmatrix} B_j + \begin{bmatrix} 0 \\ B_{j+1} W_2^l \end{bmatrix}, \tag{5.44}$$

where

$$W_1^l = P_j Q_{j-1} \tag{5.45}$$

and

$$W_2^l = P_{j+1} Q_j. \tag{5.46}$$

The superscript ^{l} refers to the left side. Similarly for the right side, we obtain the following:

$$Y_{j+1} = \begin{bmatrix} Y_j & 0 \end{bmatrix} T_j - A_j \begin{bmatrix} Y_j & 0 \end{bmatrix} - C_j \begin{bmatrix} Y_{j-1} & W_1^r \end{bmatrix} + \begin{bmatrix} 0 \\ W_2^r C_{j+1} \end{bmatrix}, \quad (5.47)$$

where

$$W_1^r = P_j^H Q_{j-1} \quad (5.48)$$

and

$$W_2^r = P_{j+1}^H Q_j, \quad (5.49)$$

with ^{r} referring to the right side. After these steps, the matrices are perturbed to simulate the round-off effect as

$$X_{j+1} = (X_{j+1} + F_j) C_{j+1}^{-1} \quad (5.50)$$

and

$$Y_{j+1} = B_{j+1}^{-1} (Y_{j+1} + F_j^H), \quad (5.51)$$

where F_j is a random matrix having norm of $\epsilon_{\text{machine}}$. If the test

$$\overline{d}_{j+1} \leq \sqrt{\epsilon_{\text{machine}}} \quad (5.52)$$

fails, then a retroactive modified Gram–Schmidt procedure (similar to the local one above) is needed. The dominant operation of this algorithm is calculating the inner products of the Lanczos blocks producing the W vector blocks.

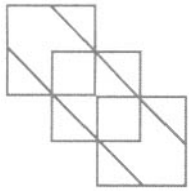
The specialized issues discussed in the last 3 sections (sections 5.5–5.7) and incorporated into the algorithm shown in section 5.4 give the foundation for a very successful industrial implementation in NASTRAN [46]. The details of such an elaborate algorithm are beyond the scope of this book.

It is important to emphasize the fact that the block biorthogonal method presented in this chapter requires that the right and left vector blocks have the same dimension. An interesting and fairly recent alternative, called multiple starting vector method, is presented in [8]. The method allows a different number of left- and right-hand vectors reacting to the possible deflation in one side only. While the approach seems very promising, it is not detailed here further and has not yet been used in industry.

Part II

APPLICATIONS

This page intentionally left blank



Chapter 6

Industrial implementation of the Lanczos method

Industrial implementations of the Lanczos method must deal with a variety of issues arising in practical applications. Examples of these issues of industrial eigenvalue analysis are very large matrices and very wide frequency range of interest. The wide frequency range problem is addressed by a method called spectral transformation. The frequency domain decomposition is in essence the systematic application of the spectral transformation. These methods are discussed in sections 6.1 and 6.2.

To handle the ever-increasing size of matrices occurring in industrial eigenvalue analyses, it is naturally desirable to develop a method which accesses the matrix only in parts. This is especially valid when an a priori partitioning of the matrix is available from the physical problem. The Lanczos method easily facilitates the incorporation of geometric domain decomposition, discussed in section 6.3. For simplicity of presentation, we will concentrate on the real, symmetric case; however, the method is applicable to the other cases as well. The parallel execution of these methods is discussed in section 6.4.

6.1 Spectral transformation

For overcoming some of the numerical problems, the spectral transformation method [14] has become very successful. In industrial applications the eigenvalue spectrum is very nonhomogeneous. Regions with closely spaced eigenvalues are followed by vast empty segments, and vice versa. The convergence of commonly used eigenvalue methods is very slow in that case.

The motivation of the spectral transformation is to modify the spectral distribution to find the eigenvalues more efficiently. This is done in the form of a transformed eigenvalue:

$$\mu = \frac{1}{\lambda - \lambda_s}, \tag{6.1}$$

where λ_s is an appropriately chosen shift. The graphical representation of this is a hyperbola shifted to the right by λ_s in the (μ, λ) coordinate system. As shown in Figure 6.1, this transformation enables one to find closely spaced eigenvalues in the neighborhood of λ_s in a well-separated form on the μ axis.

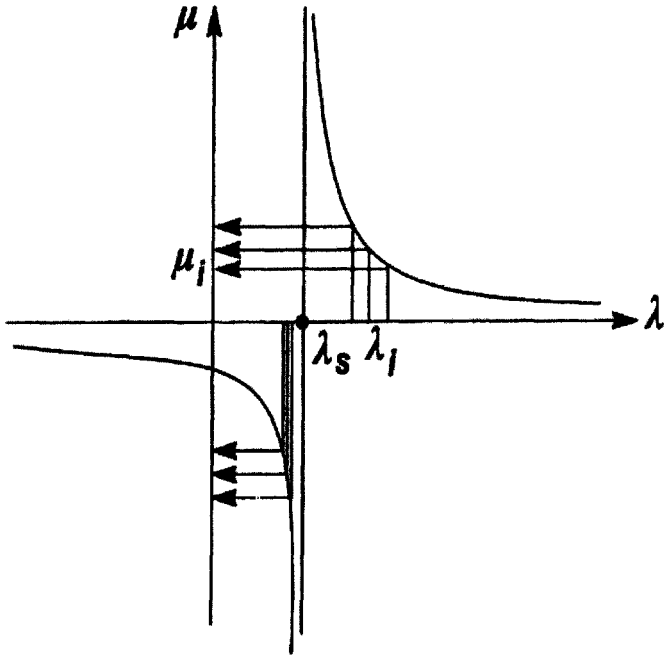


Figure 6.1. *Spectral transformation.*

Applying the spectral transformation to our original problem of

$$A\underline{x} = \lambda \underline{x} \tag{6.2}$$

in the form of

$$\lambda = \frac{1}{\mu} + \lambda_s, \tag{6.3}$$

we get

$$(A - \lambda_s I)^{-1}\underline{x} = \mu \underline{x} \tag{6.4}$$

or

$$\overline{A}\underline{x} = \mu \underline{x}. \tag{6.5}$$

In practical computations the $\overline{A}$ matrix of course is never explicitly formed. The first time that the Lanczos method requires an operator multiplication of

$$z = \overline{A}\underline{x}, \tag{6.6}$$

one executes the

$$(A - \lambda_s I) = LDL^T \tag{6.7}$$

symmetric factorization followed by the

$$LDL^T z = \underline{x} \tag{6.8}$$

forward-backward substitution. In the iterations only the substitution steps are executed until a new spectral transformation is completed. While the cost of the substitution operations may be seemingly too high in the place of the matrix-vector multiplication, the numerical advantages gained clearly outweigh the costs. The spectral transformation step may be executed at multiple, consecutive $\lambda_{s1}, \lambda_{s2}, \ldots$ locations, enabling the traversal of very wide frequency ranges of interest, which are commonplace in the industry.

The possible problem of λ_s being identical (or too close) to an eigenvalue must be resolved by a singularity detection mechanism. The singularity detection is done in the factorization operation. (For more details on the industry standard solution, see [13].) It is usually executed by monitoring the terms of the D(iagonal) matrix relative to the corresponding original terms in $A - \lambda_s I$. If their ratio is too high (λ_s is too close to an eigenvalue), the value of λ_s is perturbed and the factorization is repeated.

Another advantage of the spectral transformation is that the decomposition at the various shifts produces a mechanism to monitor the distribution of eigenvalues. Namely, the number of negative terms on the D(iagonal) factor is equivalent to the so-called Sturm number. The Sturm number is the number of alterations of sign in the so-called Sturm sequence $d_0, d_1, \ldots, d_{n-1}$, where $d_i = \det(A_i - \lambda_s I)$ is the determinant of the leading ith principal submatrix of the shifted matrix with $d_0 = 1$.

It follows that the Sturm number also indicates the number of eigenvalues located to the left of the current shift λ_s in the spectrum. This is a tool exploited by all industrial eigenvalue solution software packages to verify that all eigenvalues in a given region bounded by two λ_s values are found. If the number of eigenvalues found in a region is less than the differences in the Sturm numbers at the boundaries of the region, the region is bisected and additional iterations are executed as needed. If the number of eigenvalues found in the region exceeds the Sturm count difference, then an error has occurred and one or more spurious eigenvalues were found.

6.2 Frequency domain decomposition

The frequency domain decomposition is simply a frequency subdivision approach in which a series of spectral transformations are executed in a systematic fashion. These result in automatically created segments of the user-defined frequency spectrum that may also be statically assigned to different nodes of a parallel computer.

It is important to establish these intermediate spectral transformation locations based on the eigenvalue distribution in order to be able to reach better load balancing. The essence of a frequency domain parallel method is that all processors solve the same global problem in different frequency intervals (the segments).

The decomposition of the frequency domain is advantageous in cases where the problem size is moderately large but the frequency spectrum of interest is very wide.

The frequency domain decomposition is realized by creating p segments in the

frequency domain of interest, as follows:

$$\begin{bmatrix} \textit{Segment} & \textit{Lower Frequency} & \textit{Upper Frequency} \\ 1 & F_{\min} = f_0 & f_1 \\ 2 & f_1 & f_2 \\ . & . & . \\ p & f_{p-1} & f_p = F_{\max} \end{bmatrix}, \tag{6.9}$$

where $F_{\min}$, $F_{\max}$ are the given global frequency range boundaries. The frequency domain principle manifests itself in finding the eigenvalues only in specific segments of the spectrum as

$$\lambda^i = \begin{bmatrix} f_{i-1} & f_i \end{bmatrix}, \tag{6.10}$$

where f_{i-1}, f_i are the boundaries of the ith frequency segment and λ^i are all the eigenvalues in the segment.

6.3 Geometric domain decomposition

The geometric domain decomposition technique is based on decomposing a large problem into smaller pieces and calculating the eigensolution from the pieces. The geometric domain decomposition is executed by an automatic domain decomposition tool. Such tools [24] work from geometric connectivity information and use heuristic algorithms to create partitions. The main criteria in creating partitions are the following: minimizing the boundary between the domains, achieving load balance, and minimizing the cost of the solution of the interior of the subdomains.

The geometric domain decomposition principle is applicable to very large problem sizes but narrower frequency spectrums. It is important to emphasize that this version still solves the global eigenvalue problem as "exactly" as the undivided run. The significant difference is that the global matrix is appropriately partitioned, and only certain partitions need to be available at any given time. Similarly, this method needs only to access the partitions of the Lanczos vectors and the eigenvectors corresponding to the matrix partitions processed at the time. In addition, the portion related to the boundary between the matrix partitions is also needed.

Clearly, there are many advantages of this method from a software engineering point of view. One specific advantage is the execution of this process with parallel computers. While the implementation aspects of this method are also very interesting, especially on distributed memory parallel computers or networks, it is beyond the scope of this book and is omitted.

In the following the main steps of the Lanczos eigensolution method will be considered in the domain decomposition context. The steps of the spectral transformation and the execution of a Lanczos step in partitioned form are addressed in more detail.

6.3.1 Matrix partitioning

Let us assume the shifted matrix is partitioned into submatrices as follows:

$$A - \lambda_s I = \begin{bmatrix} A_{oo}^1 & & & & A_{oa}^1 \\ & A_{oo}^2 & & & A_{oa}^2 \\ & & \cdot & & \cdot \\ & & & A_{oo}^j & A_{oa}^j \\ & & & & \cdot \\ A_{ao}^1 & A_{ao}^2 & \cdot & A_{ao}^j & \cdot \quad A_{aa} \end{bmatrix}, \tag{6.11}$$

where the superscript j refers to the jth partition, subscript a refers to the common boundary of the partitions, and s is the number of partitions, so $j = 1, 2, \ldots, s$. The size of the global matrix as well as the global eigenvectors is N. For the presentation of the algorithm, let us partition the Lanczos vectors accordingly:

$$x = \begin{bmatrix} x_o^1 \\ x_o^2 \\ \cdot \\ x_o^j \\ \cdot \\ x_a \end{bmatrix}. \tag{6.12}$$

At the jth stage of the algorithm (or on the jth processor in a parallel implementation) only the jth partition of

$$A^j = \begin{bmatrix} A_{oo}^j & A_{oa}^j \\ A_{ao}^j & A_{aa}^j \end{bmatrix} \tag{6.13}$$

will be available, where A_{aa}^j is the complete boundary of the jth partition that may be shared by several other partitions. Similarly the local Lanczos vector component is

$$x^j = \begin{bmatrix} x_o^j \\ x_a^j \end{bmatrix}, \tag{6.14}$$

where x_a^j is a partition of x_a.

6.3.2 Partitioned matrix decomposition

Since the A matrix is available only in the partitions shown above, the decomposition will require several steps. First, the decomposition of the interior of the jth partition is as follows:

$$A^j = \begin{bmatrix} A_{oo}^j & A_{oa}^j \\ A_{ao}^j & A_{aa}^j \end{bmatrix} = \begin{bmatrix} L_{oo}^j & 0 \\ L_{ao}^j & I \end{bmatrix} \begin{bmatrix} D_{oo}^j & 0 \\ 0 & \overline{A}_{aa}^j \end{bmatrix} \begin{bmatrix} L_{oo}^{T,j} & L_{ao}^{T,j} \\ 0 & I \end{bmatrix}, \tag{6.15}$$

where the identity matrices are not computed—they are presented only to make the matrix equation algebraically correct—and the $\overline{A}_{aa}^j$ submatrix is the Schur complement of the jth partition:

$$\overline{A}_{aa}^j = A_{aa}^j - L_{ao}^j D_{oo}^j L_{ao}^{T,j}, \tag{6.16}$$

where $A_{aa} = \sum_{j=1}^{s} A_{aa}^{j}$. Next the individual Schur complement matrices are summed up,

$$\overline{A}_{aa} = \sum_{j=1}^{s} \overline{A}_{aa}^{j}, \tag{6.17}$$

to create the global Schur complement.

Finally, the global Schur complement is decomposed as

$$\overline{A}_{aa} = L_{aa} D_{aa} L_{aa}^{T}. \tag{6.18}$$

6.3.3 Partitioned substitution

The forward substitution of (6.8) in the partitioned form is

$$\begin{bmatrix} L_{oo}^{j} D_{oo}^{j} & 0 \\ L_{ao}^{j} D_{oo}^{j} & I \end{bmatrix} \begin{bmatrix} \overline{z}_{o}^{j} \\ \overline{z}_{a} \end{bmatrix} = \begin{bmatrix} x_{o}^{j} \\ x_{a} \end{bmatrix}. \tag{6.19}$$

Please note that the I submatrix is included to ensure compatibility. The forward substitution on the first block may be executed for the interior of all partitions in advance:

$$\overline{z}_{o}^{j} = [L_{oo}^{j} D_{oo}^{j}]^{-1} x_{o}^{j}. \tag{6.20}$$

The second block of the forward solve for each partition yields

$$\overline{z}_{a}^{j} = x_{a} - L_{ao}^{j} D_{oo}^{j} \overline{z}_{o}^{j}, \tag{6.21}$$

which then has to be summed up for all partitions as

$$\overline{z}_{a} = \sum_{j=1}^{s} \overline{z}_{a}^{j}. \tag{6.22}$$

The global boundary solution is a complete forward-backward substitution of

$$L_{aa} D_{aa} L_{aa}^{T} z_{a} = \overline{z}_{a}. \tag{6.23}$$

The partitions' interior solution will be finalized based on the backward part of (6.8) as

$$\begin{bmatrix} L_{oo}^{j\,T} & L_{ao}^{j\,T} \\ 0 & I \end{bmatrix} \begin{bmatrix} z_{o}{}^{j} \\ z_{a} \end{bmatrix} = \begin{bmatrix} \overline{z}_{o}{}^{j} \\ z_{a} \end{bmatrix}, \tag{6.24}$$

which is computationally equivalent to

$$z_{o}^{j} = L_{oo}^{j\,-T} (\overline{z}_{o}^{j} - L_{ao}^{j\,T} z_{a}), \tag{6.25}$$

where z_{o}^{j} is the final result in the interior of the jth partition.

6.3.4 Partitioned Lanczos step

At this point we have calculated both components z_o^j, z_a^j of the local z_k^j vector partitions. Note that in sections 6.3.1–6.3.3 the subscript k of the Lanczos step has been omitted for the sake of clarity. The appropriate partitions of the last two Lanczos vectors x_k^j, x_{k-1}^j are also ready. Hence, the partitioned Lanczos step may be executed as follows:

1. Calculate partition inner product:

$$\alpha_k^j = z_k^{T,j} z_k^j. \tag{6.26}$$

2. Accumulate global inner product:

$$\alpha_k = \sum_{j=1}^{p} \alpha_k^j.$$

3. Execute partitioned step:

$$x_{k+1}^j = z_k^j - \alpha_k x_k^j - \beta_{k-1} x_{k-1}^j. \tag{6.27}$$

The calculation of the biorthogonality parameter follows steps 1–3 above. First

$$\beta_k^j = (x_{k+1}^{T,j} x_{k+1}^j)^{1/2} \tag{6.28}$$

is computed and then

$$\beta_k = \sqrt{\sum_{j=1}^{p} (\beta_k^j)^2}$$

is collected. This leads to the next normalized Lanczos vector

$$x_{k+1}^j = x_{k+1}^j / \beta_k, \tag{6.29}$$

and the partitioned Lanczos step is completed.

It is important to point out that when accumulating the global inner product, special care is needed to consider the shared boundary segments. This issue is beyond the scope of our discussion, but it is of paramount importance in any parallel computer implementation. Results of the practical industrial implementation of this method in connection with the block implementation is discussed in the next chapter.

6.4 Parallel computational strategy

The best parallel computational strategy is to combine the two principles of domain decomposition, sometimes called hierarchic domain decomposition. First, in the outer layer of the hierarchy, the matrix is automatically partitioned and each processor sees only a local

Table 6.1. *Processor allocation scheme.*

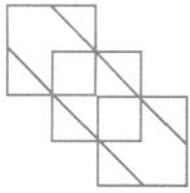
Hierarchy	$Partition^1$	...	$Partition^j$	...	$Partition^s$
$Segment^1$	ϕ^1, λ^1				ϕ^s, λ^1
...					
$Segment^i$			ϕ^j, λ^i		
...					
$Segment^p$	ϕ^1, λ^p				ϕ^s, λ^p

partition. Second, in the inner layer of the hierarchy, the frequency range is again automatically subdivided, and processors who are working on the same geometric partition work on different frequency segments.

The essence of the method [47] is that a subset of processors solves the same global problem in a given frequency interval. Another subset solves the problem in another interval. Within one subset, however, the processors work on their local geometry and communicate with the others on the boundary. Each processor of a given subset computes a (geometrically) local section of the global eigenvectors of the subset's frequency interval and communicates it with its partners in the subset. The process concludes with the final collection of the global eigenvectors of these intervals on a master processor to facilitate follow-on modal analysis techniques or appropriate postprocessing in industrial environments.

Assuming that $p \times s$ processors are available for parallel execution, the following methodology is used. For each frequency segment, s processors will be used for geometric domain partitions. Conversely, for each geometric partition p processors will be used for the various frequency segments. The Lanczos process is then executed with a specific processor allocation scheme, shown in Table 6.1. In this scheme, the numeric superscripts refer to the frequency and geometry domains, respectively. For example, the entry ϕ^j, λ^i means that Lanczos vector components of the jth geometric partition, and the set of eigenvalues belonging to the ith frequency segment are being calculated by the $((i-1) * s + j)$th processor. Naturally, the eigenvectors correspond to the appropriate eigenvalues.

Optimal selection of the s and p values—the number of frequency segments and the number of geometric partitions, respectively—is problem dependent. Generally, wide frequency ranges require high s value, while very large models require large p value. Appropriate compromise may also depend on the computer architecture used to solve the problem. For example, less communication is required in the frequency segment parallelism; hence that paradigm is more amenable to the distributed component of a hierarchic computer environment. On the other hand, the communication needs of the geometric domain parallel technique are more severe, and the method is better suited to the shared or closely coupled component of a hierarchic environment.



Chapter 7
Free undamped vibrations

The free undamped vibration of structures is very important in the automobile, aerospace, and civil engineering industries. These eigenvalue problems appearing in industrial applications generally come from finite element discretization of differential equations; they usually contain two matrices and are called generalized eigenvalue problems. The derivation of these generalized eigenvalue problems will be discussed in section 7.1 and their solution in section 7.2. A practical application example from the automobile industry will conclude the chapter.

7.1 Analysis of mechanical systems

The dynamic analysis of mechanical systems is based on the energy balance defined by

$$L = T - P, \tag{7.1}$$

where T and P are the kinetic and potential energy of the system, respectively. Their difference, L, is called the Lagrangian. The equilibrium is defined by Lagrange's equation of motion,

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0, \tag{7.2}$$

where q is a vector of generalized coordinates describing the motion of the mechanical system. Using the displacements of the system as generalized coordinates, the kinetic energy is of the form

$$T = \frac{1}{2}\int \dot{q}^T \dot{q} \rho dV. \tag{7.3}$$

Here $\dot{q}$ is the generalized velocity at a point in space, and ρ is the mass per unit volume.

The potential energy for the system is

$$P = \frac{1}{2}\int \sigma^T \epsilon dV - \int q^T f_V dV - \int q^T f_s dV. \tag{7.4}$$

Here σ, ϵ are the stresses and strains of the system. The f_V, f_S quantities are active and dissipative forces on the system. The latter forces are usually acting on the surface of the

system and are represented by surface integrals. They may be translated into the volume integral shown above with the help of the divergence theorem. They are usually related to the generalized velocities, in the form of

$$f_s = -\mu \dot{q}, \tag{7.5}$$

where μ is some kind of dissipative force coefficient, such as the coefficient of friction.

The most widely used industrial technique for solving the equilibrium equation of motion is based on the finite element method. The formulation of the technique based on the method of weighted residuals (Galerkin's method) is described briefly here.

Let us assume that the generalized coordinates are described by the following approximation:

$$q = Nu. \tag{7.6}$$

Here N are interpolation (so-called shape) functions, and u is a vector of discrete displacement values at specific spatial locations called nodal displacements. The fixed spatial locations (grid or nodal points) are established by discretizing the V volume into specifically shaped elementary volumes V_e. With these assumptions, the kinetic energy of an element is

$$T_e = \frac{1}{2}\dot{u}^T \int N^T N \rho dV_e \dot{u}. \tag{7.7}$$

The total kinetic energy of the system is the sum of all the elemental kinetic energies:

$$T = \frac{1}{2}\dot{u}^T M \dot{u}, \tag{7.8}$$

where $\dot{u}$ is the vector of nodal point velocities, and M is the mass matrix calculated from

$$M = \sum \int N^T N \rho dV_e. \tag{7.9}$$

Let us now introduce some mechanical considerations. Hooke's law represents the stress-strain relationship as

$$\sigma = \epsilon E, \tag{7.10}$$

where E is Young's modulus. The strains are further related to the displacements via a strain-displacement matrix as

$$\epsilon = Du. \tag{7.11}$$

The first term of the potential energy under the above assumptions is

$$\frac{1}{2}\int \sigma^T \epsilon dV = \frac{1}{2}\sum \int \sigma^T \epsilon dV_e. \tag{7.12}$$

Applying the mechanical considerations, we get

$$\frac{1}{2}\sum \left(u^T \int D^T E D dV_e u \right) = \frac{1}{2} u^T K u, \tag{7.13}$$

where K is the stiffness matrix. The second kinetic energy term is similarly modified:

$$\int q^T f_V dV = \sum \int q^T f_e dV_e = \sum \int u^T N^T f_e dV_e = u^T F. \tag{7.14}$$

F is the vector of active forces. Finally, the third, dissipative energy term yields

$$\int q^T f_S dV = \sum \int q^T f_e dV_e = \sum \int u^T N^T (-\mu) N \dot{u} dV_e = u^T B \dot{u}. \tag{7.15}$$

The damping matrix, B, is clearly identifiable in the above equation.

Substituting into (7.2), differentiating, and reordering yields the dynamic equilibrium equation of

$$M\ddot{u} + B\dot{u} + Ku = F. \tag{7.16}$$

This equation is the most general equilibrium equation, the forced damped vibration of the structure. Lack of damping (B) and loads (F) lead to the simpler undamped free vibration problem presented next.

7.2 Generalized linear eigenvalue problem

The undamped free vibration of structures result in a generalized linear eigenvalue problems of the form

$$M\ddot{u} + Ku = 0, \tag{7.17}$$

where the K and M matrices are the stiffness and mass matrices, respectively, u is the displacement vector, and $\ddot{u}$ is the acceleration. The solutions of these problems are of the form

$$u = \phi \cos(\omega t). \tag{7.18}$$

Introducing $\lambda = \omega^2$, differentiating twice, and reordering results in the form of the typical generalized linear eigenvalue problem:

$$K\phi - \lambda M\phi = 0. \tag{7.19}$$

Since the frequency range of interest in these problems is the lower end of the spectrum, it is advisable to use the spectral transformation introduced in section 6.1:

$$\mu = \frac{1}{\lambda - \lambda_s}. \tag{7.20}$$

This will change the problem into

$$(K - \lambda_s M)\phi = \frac{1}{\mu} M\phi, \tag{7.21}$$

which results in a canonical form amenable to the Lanczos algorithm

$$\mu\phi = (K - \lambda_s M)^{-1} M\phi. \tag{7.22}$$

At this point the block symmetric method introduced in Chapter 4 may be executed. The process scheme, consisting of the spectral transformation, the block tridiagonal reduction, and the eigenvalue solution steps, is depicted in Figure 7.1. The T_B in the figure represents the T_J matrix of Algorithm 4.4.1. The Λ represents both the T_j scalar tridiagonal matrix and diag(λ).

The eigenvectors are invariant under the spectral transformation, and the eigenvalues may be recovered as

$$\lambda = \frac{1}{\mu} + \lambda_s. \tag{7.23}$$

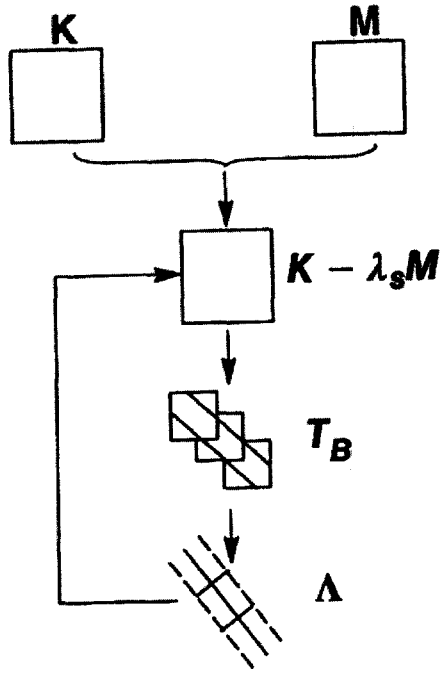


Figure 7.1. *Generalized solution scheme.*

7.3 Normal modes analysis application

The model used to demonstrate this application was an automobile (body in white) model (see Figure 7.2, for example) consisting of 232,649 nodes and 232,404 (mainly quadrilateral) elements resulting in 1,377,677 global degrees of freedom. Normal modes analysis was executed up to 200 Hz with various combinations of the hierarchic parallel execution using NASTRAN [47].

The runs were executed on an IBM SP machine, which is a cluster of 4 workstation nodes. Each node is an IBM 44P Model 270 workstation, with 4 GBytes of memory and 144 GBytes of disk space. Each node contains 4 POWER3-II (375 MHz) processors. The nodes are connected with IBM's SP switch, which enables point-to-point communication between the nodes and has a bandwidth of 150 MBytes per second.

Table 7.1 contains results related to the eigenvalue solution module. Every scenario (a certain number of partitions and segments) has 4 data items. In the first row are the main processor CPU times in seconds and the elapsed times in minutes:seconds. The second row starts with the I/O high-water mark, the maximum disk storage during the analysis, in GBytes. It is followed by a speed-up based on the elapsed time. Analyzing the results, one can make interesting observations. The first line (one segment, various number of partitions) represents the case of geometric domain decomposition only. It is easy to see that along with a good speed-up, the I/O requirements have significantly decreased as the number of

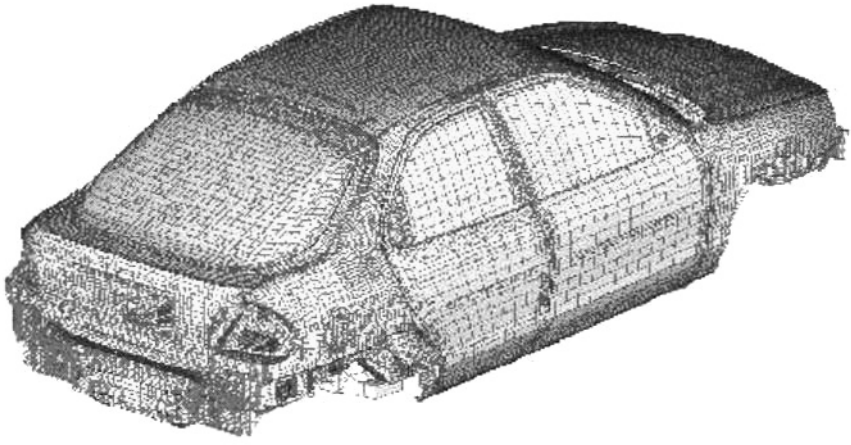


Figure 7.2. *Automobile model.*

Table 7.1. *Normal modes analysis performance results.*

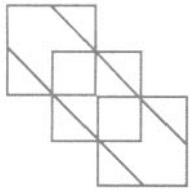
	1 partition		2 partitions		4 partitions	
Times	CPU	Elapsed	CPU	Elapsed	CPU	Elapsed
Statistics	I/O	Speed-up	I/O	Speed-up	I/O	Speed-up
1	21,080	401:34	11,186	204:56	6,905	151:04
Segment	25.5	1.0	15.2	1.96	7.9	2.65
2	11,928	215:13	6,869	132:29	4,002	74:58
Segments	18.7	1.87	11.4	3.03	6.8	5.38
4	8,395	167:55	4,596	84:42	2,688	50:00
Segments	13.2	2.40	7.3	4.75	5.1	8.03

partitions increased.

Similarly good speed-up was obtained when using only the frequency domain decomposition (first column: one partition, various number of segments). The I/O requirements have not decreased as dramatically, since in this case each processor solves the same global problem. Although the number of eigenvectors in a segment is less, ultimately all of them are collected on the master processor.

Results of the hierarchic approach show that the optimal selection of the distribution of the hierarchy (how many processors for frequency vs. geometry) is problem dependent and requires more study. The result utilizing the evenly distributed hierarchic approach with 4 processors (2 partitions and 2 segments: 132:29) is better than any of the 4-processor runs (4 segments: 167:55; 4 partitions: 151:04), utilizing only one domain principle.

This tendency continued on the 8-processor runs. The 4 by 2 and 2 by 4 runs shown in Table 7.1 are better than any of the single principle 8-processor runs (not shown in the table). Finally, the 4-partition and 4-segment run resulted in truly excellent performance. The original task required essentially a work day to execute. The 16-processor hierarchic time is less than an hour, resulting in an approximately eightfold speed-up.



Chapter 8
Free damped vibrations

The damping effect is represented by the presence of the damping matrix. This makes the problem more difficult, as the damped free vibration analysis leads to generalized quadratic eigenvalue problems.

The details of the efficient solution of the quadratic problems are addressed in this chapter, and another industrial example is given.

8.1 Generalized quadratic eigenvalue problem

The quadratic eigenvalue problems result from the differential equation

$$M\ddot{u} + B\dot{u} + Ku = 0, \tag{8.1}$$

where B is the damping matrix, and $\dot{u}$ refers to the velocity. The solution of this homogeneous system (the free, but damped, vibrations) is of the form

$$u = e^{\lambda t}\phi, \tag{8.2}$$

where ϕ is a vector of complex numbers, and the λ eigenvalue is also complex in general. Substitution of the last equation and its time derivatives into the prior one and some reorganization gives the characteristic equation

$$(M\lambda^2 + B\lambda + K)\Phi = 0. \tag{8.3}$$

A detailed analysis on the solution error of these problems is given in [40], and some hints for the linearization of this problem are in [37].

In order to solve the quadratic eigenvalue problem, first a linearization transformation is executed. This transformation converts the original quadratic problem to a linear problem twice the size.

It is obtained by simply rewriting (8.1) as a 2×2 block matrix equation:

$$\lambda \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix} + \begin{bmatrix} B & K \\ -I & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix} = 0, \tag{8.4}$$

where $\dot{\phi} = \lambda\phi$. This equation is now linear; however, there are shortcomings. Specifically, one would need to invert both the mass and the damping matrices and an unsymmetric, indefinite matrix built from the damping and the stiffness matrices in order to reach a solution. Even though the explicit inverses are not needed, the numerical decompositions on either of these matrices may not be very well defined.

An advancement is possible by executing the two steps of the spectral transformation. The first step is introducing an appropriate shift:

$$\lambda = \lambda_0 + \mu. \tag{8.5}$$

With the shift, the linear equation may be rewritten as

$$\begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix} = \mu \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix}. \tag{8.6}$$

The second step of the spectral transformation is to invert the problem via

$$\Lambda = \frac{1}{\mu}. \tag{8.7}$$

By substituting and reordering, we get

$$\Lambda \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix} = \begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix}^{-1} \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix}. \tag{8.8}$$

The latter equation is a canonical form of

$$\Lambda x = Ax, \tag{8.9}$$

where

$$A = \begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix}^{-1} \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}. \tag{8.10}$$

The form allows the singularity of the mass matrix; however, the zero subspaces of K, B, and M may not coincide. This is a much lighter and more practical restriction.

8.2 Recovery of physical solution

The physical eigenvalues may easily be recovered from the backward substitution of the spectral transformation:

$$\lambda = \frac{1}{\Lambda} + \lambda_0. \tag{8.11}$$

In order to find the relationship between the mathematical and physical eigenvectors, let us rewrite (8.4) in the following block notation:

$$(\lambda\overline{M} + \overline{K})x = 0, \tag{8.12}$$

where

$$x = \begin{bmatrix} \dot{\phi} \\ \phi \end{bmatrix}. \tag{8.13}$$

The block matrices are simply

$$\overline{K} = \begin{bmatrix} B & K \\ -I & 0 \end{bmatrix} \tag{8.14}$$

and

$$\overline{M} = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}. \tag{8.15}$$

Substituting and reordering yields

$$[(\overline{K} + \lambda_0 \overline{M})^{-1} \overline{M} + \Lambda I)]x = 0. \tag{8.16}$$

This proves that the right eigenvectors are invariant under the spectral transformation; i.e., the right physical eigenvectors are the same as their mathematical counterparts, apart from the appropriate partitioning.

For the left-hand problem we also use the block notation

$$\underline{y}^H(\lambda \overline{M} + \overline{K}) = 0, \tag{8.17}$$

with a left-hand physical eigenvector of

$$\underline{y}^H = \begin{bmatrix} \dot{\psi}^H & \psi^H \end{bmatrix}. \tag{8.18}$$

Substituting again and introducing an appropriate identity matrix to accommodate the left multiplication yields

$$\underline{y}^H(\overline{K} + \lambda_0 \overline{M})(\overline{K} + \lambda_0 \overline{M})^{-1}[\overline{M} + (\overline{K} + \lambda_0 \overline{M})\Lambda] = 0. \tag{8.19}$$

By multiplying we obtain

$$\underline{y}^H(\overline{K} + \lambda_0 \overline{M})[(\overline{K} + \lambda_0 \overline{M})^{-1} \overline{M} + \Lambda I] = 0, \tag{8.20}$$

which is equivalent to

$$[(\overline{K} + \lambda_0 \overline{M})^H \underline{y}]^H [(\overline{K} + \lambda_0 \overline{M})^{-1} \overline{M} + \Lambda I] = 0. \tag{8.21}$$

Since the original mathematical problem we solve is

$$y^H[(\overline{K} + \lambda_0 \overline{M})^{-1} \overline{M}) + \Lambda I] = 0, \tag{8.22}$$

the left-hand physical eigenvectors are not invariant under the transformation. Comparing gives the relationship

$$y^H = [(\overline{K} + \lambda_0 \overline{M})^H \underline{y}]^H \tag{8.23}$$

or, expanding into the solution terms,

$$y = -\begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix}^H \underline{y}. \tag{8.24}$$

Finally,

$$\underline{y} = -\begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix}^{-1,H} y. \tag{8.25}$$

The cost of this back-transformation is not very large since the factors of the operator matrix are available; we need a forward-backward substitution only.

8.3 Orthogonality diagnostics

From the eigenvalue solution it will be guaranteed that the left and right mathematical eigenvectors are computationally orthonormal:

$$Y^H X = \overline{I}, \tag{8.26}$$

where $\overline{I}$ is an approximate identity matrix with off-diagonals as computational zeros. Monitoring these terms and printing the largest one or the ones above a certain threshold will give a final orthogonality check.

Based on the physical eigenvalues recovered by the shift formulae and the physical eigenvectors, another orthogonality criterion can be formed. Using the left and right solutions, the following equations are true for the problem:

$$(M\lambda_i^2 + B\lambda_i + K)\phi_i = 0 \tag{8.27}$$

and

$$\psi_j^H(M\lambda_j^2 + B\lambda_j + K) = 0. \tag{8.28}$$

By appropriate pre- and postmultiplications we get

$$\psi_j^H(M\lambda_i^2 + B\lambda_i + K)\phi_i = 0 \tag{8.29}$$

and

$$\psi_j^H(M\lambda_j^2 + B\lambda_j + K)\phi_i = 0. \tag{8.30}$$

A subtraction yields

$$\psi_j^H M\phi_i(\lambda_i^2 - \lambda_j^2) + \psi_j^H B\phi_i(\lambda_i - \lambda_j) = 0. \tag{8.31}$$

Assuming $\lambda_j \neq \lambda_i$ we can shorten and obtain a mass orthogonality criterion as

$$O_1 = \psi_j^H M\phi_i(\lambda_i + \lambda_j) + \psi_j^H B\phi_i. \tag{8.32}$$

The O_1 matrix also has computational zeros as off-diagonal terms (when $i \neq j$) and nonzero (containing $2\lambda_i$) diagonal terms.

By premultiplying (8.27) by $\lambda_j \psi_j^H$, postmultiplying (8.28) by $\lambda_i \phi_i$, and subtracting them, we obtain

$$\lambda_j \psi_j^H (M\lambda_i^2 + B\lambda_i + K)\phi_i - \psi_j^H (M\lambda_j^2 + B\lambda_j + K)\lambda_i \phi_i = 0. \tag{8.33}$$

By expanding and canceling, we get

$$(\lambda_i - \lambda_j)\psi_j^H M \phi_i \lambda_i \lambda_j + (\lambda_j - \lambda_i)\psi_j K \phi_i = 0. \tag{8.34}$$

Assuming again that $\lambda_j \neq \lambda_i$, we can shorten and obtain another orthogonality condition recommended mainly for the structural damping option (the case when there is no B matrix, but damping is presented in the K matrix) as

$$O_2 = \lambda_j \psi_j^H M \lambda_i \phi_i - \psi_j^H K \phi_i. \tag{8.35}$$

This orthogonality matrix will also have zero off-diagonal terms, but nonzero (containing λ_i^2) diagonal terms.

8.4 Implicit operator multiplication

From the structure of the A matrix

$$A = \begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix}^{-1} \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}, \tag{8.36}$$

it is clear that it is not needed to be built explicitly. In the following the implicit execution of the operator matrix multiplication in both the transpose and nontranspose case is detailed [29].

In the nontranspose case any $z = Ax$ operation in the recurrence will be identical to the

$$\begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix} z = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} x \tag{8.37}$$

solution of systems of equations. Let us consider the partitioning of the vectors according to the A matrix partitions:

$$\begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}. \tag{8.38}$$

Developing the first row results in

$$(-B - M\lambda_0)z_1 - Kz_2 = Mx_1. \tag{8.39}$$

Developing the second row, we have

$$z_1 = \lambda_0 z_2 + x_2. \tag{8.40}$$

Substituting (8.40) into (8.39), we get

$$(-B - M\lambda_0)(\lambda_0 z_2 + x_2) - Kz_2 = Mx_1. \tag{8.41}$$

Some reordering yields

$$-(K + \lambda_0 B + \lambda_0^2 M)z_2 = Mx_1 + (B + \lambda_0 M)x_2. \tag{8.42}$$

The latter formulation has significant advantages. Besides avoiding the explicit formulation of A, the decomposition of the $2N$ size problem is also avoided.

It is important that the transpose operation be executed without any matrix transpose at all. In this case any $y^T = x^T A$ operation in the recurrence will be identical to the

$$y^T = x^T \begin{bmatrix} -B - M\lambda_0 & -K \\ I & -\lambda_0 I \end{bmatrix}^{-1} \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix} \tag{8.43}$$

operation. Let us introduce an intermediate vector z:

$$z^T = x^T \begin{bmatrix} -(B + M\lambda_0) & -K \\ I & -\lambda_0 I \end{bmatrix}^{-1}. \tag{8.44}$$

Now partitioning these vectors according to the matrix partitions and transposing, we obtain

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -(B + M\lambda_0)^T & I \\ -K^T & -\lambda_0 I \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \tag{8.45}$$

From the first row of (8.45), we get

$$x_1 = (-B - M\lambda_0)^T z_1 + z_2. \tag{8.46}$$

From the second row, we get

$$x_2 = -K^T z_1 - \lambda_0 z_2. \tag{8.47}$$

Expressing z_2 from (8.46), substituting into (8.47), and reordering yields

$$x_2 + \lambda_0 x_1 = -(K^T + \lambda_0 B^T + \lambda_0^2 M^T)z_1 \tag{8.48}$$

or, in the solution form,

$$z_1 = -(K^T + \lambda_0 B^T + \lambda_0^2 M^T)^{-1}(x_2 + \lambda_0 x_1). \tag{8.49}$$

The lower part of the z vector is recovered from (8.46) as

$$z_2 = x_1 + (B + \lambda_0 M)^T z_1. \tag{8.50}$$

The latter two equations will also be used in the recovery of the left-hand physical eigenvectors. Finally,

$$y^T = z^T \begin{bmatrix} M & 0 \\ 0 & I \end{bmatrix}. \tag{8.51}$$

This formulation has even more significant advantages than the nontranspose case. Besides avoiding the explicit formulation of A and the decomposition of the $2N$ size problem, the transpose may also be avoided with left-hand multiplications and forward-backward substitution.

8.5 Implicit operator algorithm

For the nontranspose case (right-hand operation) the following algorithm may be used.

Algorithm 8.5.1. Nontranspose implicit operator algorithm

1. Create shifted matrices:
   $M_1 = K + \lambda_0 B + \lambda_0^2 M$
   $M_2 = B + \lambda_0 M$

2. Multiply right-hand-side vectors:
   $\overline{z}_2 = Mx_1 + M_2 x_2$

3. Decompose M_1:
   $M_1 = LU$

4. Solve for z_2:
   $LUz_2 = \overline{z}_2$

5. Calculate z_1:
   $z_1 = \lambda_0 z_2 + x_2$

For the transpose case (left-hand operation) the algorithm is as follows.

Algorithm 8.5.2. Transpose implicit operator algorithm

1. Same as in nontranspose case.

2. Multiply right-hand-side vectors:
   $\overline{z}_1 = \lambda_0 x_1 + x_2$

3. Same as in nontranspose case.

4. Left-hand solve for z_1:
   $z_1^T LU = \overline{z}_1^T$

5. Left-multiply M_2 with z_1:
   $z_2^T = z_1^T M_2 + x_1^T$

6. Obtain y as:
   $y_1^T = z_1^T M$
   $y_2^T = z_2^T$

These algorithms, in connection with the block unsymmetric algorithm, may solve the difficult industrial problem shown in the next section.

It is important to mention the several computational advantages of the implicit operator technique. They are based on the ability to exploit the characteristics of the participating matrices, such as their symmetry, sparsity, or real (as opposed to complex) nature. The advantage is truly spectacular when out-of-core handling of the original structural matrices can be avoided. Out-of-core handling is a common occurrence in large industrial problems solved by commercial software. As demonstrated by the following industrial application, the implicit operator multiplication enables the practical solution of large quadratic eigenvalue problems in a direct fashion without component modal reduction [49] or domain decomposition.

8.6 Complex eigenvalue analysis application

An important industrial application resulting in the heavy usage of the method developed above is the analysis of automobile brakes. The physical phenomenon occurring at braking is a stick-slip-type vibration between the friction pads on the caliper and the rotor. This coupled vibration of the brake rotor and pad sometimes generates an uncomfortable noise and may also have a negative effect on the braking performance; therein lies the importance of the complex eigenvalue analysis.

The friction between the pads and the rotor is mainly a function of pressure and velocity. The pressure-based friction, sometimes called friction stiffness, creates a relationship between the normal and tangential components of the brake elements. The mathematical effect of this is the asymmetry of the K stiffness matrix of the finite element model. In some occasions a global damping may also be applied to the system, resulting in a complex K matrix.

The velocity-based friction, also known as friction damping, introduces the B damping matrix of the model. It is very difficult to establish the appropriate numeric terms in modeling the damping and is a topic of active research at automobile manufacturers.

The implicit solution technique detailed in this chapter and the block unsymmetric biorthogonal method of Chapter 5 have been implemented in NASTRAN. This technique enables the direct analysis of automobile brakes without invoking modal or substructure reduction. These techniques, used in the past in lieu of a feasible computational solution of the direct problem, required considerable analytical expertise and introduced approximation errors.

A brake model of a test problem (such as shown in Figure 8.1) had about 150,000 degrees of freedom after removing constraints and boundary conditions. Elimination of constraints from finite element models is a fundamental and sometimes very time consuming computation [50], an issue beyond the scope of this book. The direct analysis of this brake model using the implicit method described above took 6,792 CPU (not including I/O) seconds on a Cray C90 computer [45]. Of that total, 6,046 seconds were spent in the complex eigenvalue analysis computation extracting 100 eigenpairs.

The amount of I/O was a staggering 510 GBytes, due to the fact that the structural matrices were residing out of core. Still, the execution of this task proves the feasibility of the method by fitting into the turn-around acceptance span of the typical engineer.

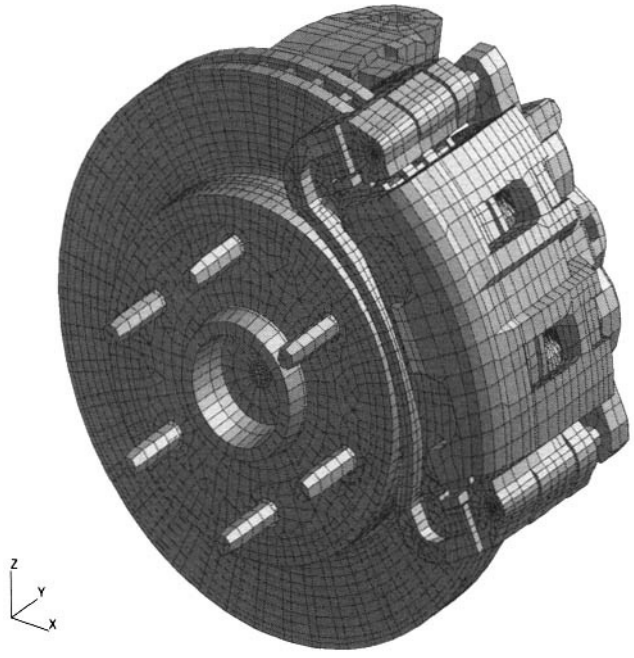
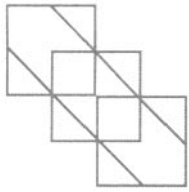


Figure 8.1. *Brake model.*

Regarding numerical accuracy, the mathematical residuals were of $O(10^{-16})$, very close to the machine precision. On the other hand, the normalized (with the input matrix norms) physical state equation residuals were only about $O(10^{-12})$, but still well below the engineering acceptance level.

This page intentionally left blank



Chapter 9
Forced vibration analysis

The goal of this chapter is to demonstrate the applicability of the Lanczos method to a difficult engineering problem which is not an eigenvalue problem. The problem is derived in detail in sections 9.1–9.3. The process known as Padé approximation via the Lanczos method is shown in section 9.4. Some data of a practical application is presented in section 9.5.

A frequently used application in the area of forced vibration analysis is the interior noise calculation of automobiles due to structural excitations. The general interior acoustic problem is a fluid-structure interaction problem where the fluid is partially or fully surrounded by an external structure and the dissipation of acoustic energy into the space is negligible.

Finite element analysis is one of the main methods used for internal acoustics. The physical model is usually discretized by finite elements, resulting in large-scale forced vibration analysis problems.

9.1 The interior acoustic problem

In general, compressible fluid inside a cavity is governed by the Euler equation of the form

$$\rho((v\nabla)v + \dot{v}) = -\nabla P, \tag{9.1}$$

where

$$\begin{aligned} \rho &= \text{density}, \\ v &= \text{velocity}, \\ P &= \text{pressure}. \end{aligned}$$

For acoustic analysis the following assumptions are used:

a. Small motion theory ($v = \dot{u}$) applies, where u is the displacement.

b. Convective momentum terms (the first terms of the Euler equation) are negligible.

These result in a simpler form of Euler equation:

$$\rho \ddot{u} = -\nabla P, \tag{9.2}$$

where $\ddot{u}$ is the acceleration.

The continuity equation of

$$\frac{\partial \rho}{\partial t} + \rho_0 \nabla v = 0 \tag{9.3}$$

is also applicable.

We also assume locally linear pressure-density behavior represented by

$$\frac{dP}{d\rho} = \gamma \frac{P}{\rho}, \tag{9.4}$$

where

$$\gamma = \frac{\rho_0 c^2}{P_0}, \tag{9.5}$$

and c is the speed of sound. Substituting into the continuity equation and integrating yields

$$P = -B \nabla u, \tag{9.6}$$

where

$$B = c^2 \rho_0 \tag{9.7}$$

is the bulk modulus. Finally, differentiating twice and substituting the simplified form of the Euler equation, we get the wave equation describing the fluid:

$$\frac{1}{B} \ddot{P} = \nabla \left(\frac{1}{\rho} \nabla P \right). \tag{9.8}$$

The connection with the surrounding structure is based on the structural-acoustic analogy which connects the structural displacements with the pressure in the fluid. It also parallels the elastic constants of the structure with the inverse of the density of the fluid. The stresses of the structure correspond to the particle acceleration in the fluid.

The following boundary conditions are also applied. At a structure-fluid interface based on momentum and continuity considerations,

$$\frac{\partial P}{\partial n} = -\rho \ddot{u}_n, \tag{9.9}$$

where n is the direction of the outward normal. At free surfaces,

$$u = P = 0. \tag{9.10}$$

9.2 Fluid-structure interaction

The derivation of the equilibrium equation for this problem will be based on the variational principle. Following another Lanczos classic, his book on variational principles [1], the virtual work statement of the above wave equation may be written as

$$\int\int\int \left[\frac{1}{B}\ddot{P} - \frac{1}{\rho}\nabla\cdot\nabla P\right] dV\delta P = 0. \tag{9.11}$$

From vector calculus, we have

$$\nabla\cdot(f\overline{F}) = \nabla f\cdot\overline{F} + f\nabla\cdot\overline{F}, \tag{9.12}$$

where f is a scalar function and $\overline{F}$ is a vector field. Applying this to the second term of the virtual work statement, letting $f = \delta P$ and $\overline{F} = \nabla P$, we get

$$\nabla\cdot(\delta P\nabla P) = \nabla\delta P\cdot\nabla P + \delta P\nabla\cdot\nabla P. \tag{9.13}$$

Since

$$\nabla\delta P\cdot\nabla P = \frac{1}{2}\delta\nabla P\cdot\nabla P, \tag{9.14}$$

substituting and reordering yields

$$\delta P\nabla\cdot\nabla P = -\frac{1}{2}\delta\nabla P\cdot\nabla P + \nabla\cdot\delta P\nabla P. \tag{9.15}$$

Finally, using the divergence theorem of the form

$$\int\int\int \nabla\cdot\overline{F}dV = \int\int \overline{F}\cdot\overline{dS}, \tag{9.16}$$

where $\overline{dS}$ is the outward-oriented surface normal vector, and using the fact

$$\ddot{P}\delta P = \frac{1}{2}\delta(\dot{P}^2), \tag{9.17}$$

we get the governing equilibrium equation

$$\int\int\int \frac{\delta}{2}\left[\frac{1}{B}\dot{P}^2 + \frac{1}{\rho}\nabla P\cdot\nabla P\right] dV - \int\int \frac{1}{\rho}\nabla P\cdot\overline{dS}\delta P = 0. \tag{9.18}$$

The finite element discretization of this can be carried out based on Galerkin's principle, assuming

$$P = \sum P_i N_i, \tag{9.19}$$

where P_i and N_i are the pressure and shape functions associated with the ith node of the fluid finite element mesh.

The first integral of the governing equation represents the virtual work and the second (surface) integral represents the boundary condition. In order for the problem to be solved, these two components have to be zero for all admissible δP.

Concentrating on the second integral and substituting the earlier boundary condition,

$$-\int\int \frac{1}{\rho}\nabla P \cdot \overline{dS}\delta P = \int\int \ddot{u}\overline{dS}\delta P = [A]\ddot{u}, \tag{9.20}$$

where the A coupling matrix describing the boundary connection between the fluid and the surrounding structure is built as

$$A(i, j) = \int\int N_j N_i \overline{dS}. \tag{9.21}$$

For the two components of the first integral, we are introducing

$$M_f(i, j) = \int\int\int \frac{1}{B} N_i N_j dV \tag{9.22}$$

and

$$K_f(i, j) = \int\int\int \frac{1}{\rho}\nabla N_i \nabla N_j dV, \tag{9.23}$$

where the f subscript stands for fluid. Finally, the connection between the forces on the structure and the pressure in the fluid is established via the coupling matrix A as

$$F = -A^T P. \tag{9.24}$$

9.3 Coupled forced vibration problem

The corresponding structural mass and stiffness matrices are derived as in section 7.1. With these matrices and the above equations, one obtains the (fluid-structure) coupled forced vibration problem of

$$M_c\ddot{u} + K_c u = P_c, \tag{9.25}$$

where the coupled matrix coefficients are

$$M_c = \begin{bmatrix} M_s & 0 \\ A & M_f \end{bmatrix} \tag{9.26}$$

and

$$K_c = \begin{bmatrix} K_s & -A^T \\ 0 & K_f \end{bmatrix}, \tag{9.27}$$

where the subscripts s and f refer to the structure and fluid, respectively, M is the mass, K is the stiffness, and A is the coupling matrix. The unknowns of the system are the structural displacements and the pressure values at the fluid node locations:

$$u = \begin{bmatrix} u_s \\ P \end{bmatrix}. \tag{9.28}$$

The loads are given as structural forces and pressure values as

$$P_c = \begin{bmatrix} F_s \\ P_f \end{bmatrix}. \tag{9.29}$$

The above forced vibration problem is called a frequency response problem in the industry. Usually the right-hand-side load vectors are frequency dependent. The F_s structural load is frequency dependent as $F_s(\omega)$, and the P_f term is an acoustic excitation.

In the case in which there is zero external excitation, the forced vibration problem reduces to a coupled unsymmetric eigenvalue problem of

$$K_c u + M_c \ddot{u} = 0. \tag{9.30}$$

This problem may be solved by the block biorthogonal version of the Lanczos method presented in Chapter 5 and will not be discussed further here.

9.4 Padé approximation via the Lanczos method

In the case of excitation, the responses of the system are usually calculated directly by the transfer function. An interesting and efficient solution is gaining ground utilizing the Lanczos process to calculate an approximate transfer function. As the approximation is of Padé type, the method is called Padé approximation via the Lanczos method.

9.4.1 Transfer function calculation

Let us execute a Laplace transform on the frequency response equation (9.25). Ignoring the c subscript for the sake of simplicity, we get

$$M(z^2 u(z) - zu(0) - \dot{u}(0)) + Ku(z) = P(z). \tag{9.31}$$

Reordering yields

$$u(z)(Mz^2 + K) = M(zu(0) + \dot{u}(0)) + P(z), \tag{9.32}$$

from which the direct solution is easy to obtain:

$$u(z) = (Mz^2 + K)^{-1}(u(0)Mz + M\dot{u}(0) + P(z)). \tag{9.33}$$

The usual partitioning into the response to the initial conditions and to the input data results in

$$u(z) = (Mz^2 + K)^{-1}(u(0)Mz + M\dot{u}(0)) + (Mz^2 + K)^{-1}P(z). \tag{9.34}$$

Introducing initial conditions $u(0) = 0$ and $\dot{u}(0) = 0$, the equation simplifies to

$$u(z) = (Mz^2 + K)^{-1}P(z) \tag{9.35}$$

or

$$u(z) = H(z)P(z), \tag{9.36}$$

where the $H(z)$ transfer function is

$$H(z) = (Mz^2 + K)^{-1}. \tag{9.37}$$

This is a rational polynomial producing the computationally exact, direct solution.

9.4.2 Approximate transfer function

Introducing $s = z^2$ we can rewrite $H(z)$ as

$$H(s) = (Ms + K)^{-1}. \tag{9.38}$$

Let us introduce a shift of s_0 where the unsymmetric factorization of

$$Ms_0 + K = LU \tag{9.39}$$

is practical. With some algebraic manipulations, we get

$$H(s) = (LU + Ms - Ms_0)^{-1}. \tag{9.40}$$

Furthermore

$$H(s) = (L(I + L^{-1}MU^{-1}s - L^{-1}MU^{-1}s_0)U)^{-1}. \tag{9.41}$$

Executing the inverse, we obtain

$$H(s) = U^{-1}(I + (s - s_0)L^{-1}MU^{-1})^{-1}L^{-1}. \tag{9.42}$$

Introducing $s_0 - s = \sigma$ and

$$A = L^{-1}MU^{-1}, \tag{9.43}$$

we arrive at

$$H(s_0 + \sigma) = U^{-1}(I - \sigma A)^{-1}L^{-1}. \tag{9.44}$$

Using a Neumann series expansion for the middle inverse,

$$(I - \sigma A)^{-1} = \sum_{j=0}^{\infty} A^j \sigma^j, \tag{9.45}$$

and its truncation gives an approximation of the transfer function as

$$H(s + \sigma) = \sum_{j=0}^{n} m_j \sigma^j, \tag{9.46}$$

where n is smaller than the problem size N of the matrices. Here the m_j are the so-called moments

$$m_j = U^{-1}A^jL^{-1}. \tag{9.47}$$

Let us express specifically the $2j$th moment,

$$m_{2j} = U^{-1}A^{2j}L^{-1} = ((A^T)^jU^{-1})^T(A^jL^{-1}), \tag{9.48}$$

and $2j + 1$st moment,

$$m_{2j+1} = U^{-1}A^{2j+1}L^{-1} = ((A^T)^jU^{-1})^TA(A^jL^{-1}). \tag{9.49}$$

They reveal that they are obtained from the members of Krylov subspaces,

$$K(A, L^{-1}) = \text{span}(L^{-1}, AL^{-1}, A^2L^{-1}, \ldots, A^{n-1}L^{-1}) \tag{9.50}$$

and

$$K(A^T, U^{-1}) = \text{span}(U^{-1}, A^TU^{-1}, (A^T)^2U^{-1}, \ldots, (A^T)^{n-1}U^{-1}). \tag{9.51}$$

The Lanczos process is an efficient way of producing these Krylov subspaces. More on the intriguing relationship between the Krylov subspaces and the Lanczos process may be found in [25]. Let us assume that n steps of a biorthogonal Lanczos process were executed, resulting in Lanczos vectors x_j, y_j, $j = 1, \ldots, n$. They are by definition biorthogonal:

$$y_j^T z_k = 0, \quad j \neq k, \quad j, k = 1, \ldots, n. \tag{9.52}$$

They also reduce the A matrix to a tridiagonal matrix of size n,

$$T_n = Y^T AX, \tag{9.53}$$

where Y and X are the collection of the y and z vectors, respectively. With the help of T_n, we can obtain an approximation of the transfer function as

$$H_n(s + \sigma) = U^{-1,T}L^{-1}e_1^T(I - \sigma T_n)^{-1}e_1. \tag{9.54}$$

The response now may be calculated by substituting H_n into (9.36).

It is possible to prove that

$$H(s + \sigma) = H_n(s + \sigma) + O(\sigma^{2n}); \tag{9.55}$$

therefore H_n is an nth order Padé approximation, since the power series expansion of H_n around σ agrees with that of $H(\sigma)$ in the first $2n$ terms.

The higher the value of n in relation to the problem size N, the better the approximation. On the other hand, the solution is more efficient when n is small compared to N. Practical implementations must find a suitable compromise.

In practice, the solution is usually required only at some components of the displacement vector $u(z)$. These components may be defined as $v(z) = T^Tu(z)$, where T is a transformation matrix of size $N \times p$. That would allow the size reduction of the transfer function, resulting in further efficiency improvements. These details are usually industry dependent (see, for example, [10]) and beyond the scope of this book.

9.5 Acoustic optimization application

The aim of acoustic optimization is to reduce the noise at a certain location inside an automobile—for instance, at the driver's ear. This means that a series of forced vibration problems are solved, between which the structural model is changed to modify the acoustic response (lower the noise). These computations are very time consuming; therefore usage of advanced methods, such as the Padé-via-Lanczos technique, is of paramount importance.

A practical application from a leading auto manufacturer had 654,560 structural and 1,252 fluid degrees of freedom. The big discrepancy is due to the fact that the fluid model

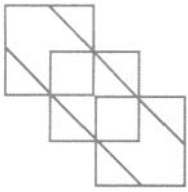
of the interior is usually much coarser than the structural model. The excitation originated from the front tires and the goal was reached in 5 optimization steps.

The overall analysis required 37,354.7 CPU seconds and 19,646.6 I/O seconds on a Cray C90 supercomputer using one processor [44]. One optimization step required 4,181.4 CPU and 3,199.4 I/O seconds. Each forced vibration analysis in these steps (for a frequency range of 100 Hz) took most (4,153.1 and 3,120.4 CPU and I/O seconds respectively) of the time of the optimization steps.

Additional speed-up in the solution of this problem was obtained by the use of multiple processors; details of this will not be discussed here.

Note that it is also possible to organize the internal acoustics problem using the pressure derivatives (and corresponding structural velocities) as unknowns, yielding a more convenient symmetric formulation. In this case, however, to obtain the results interesting to engineers, we need an extra integration step. Further details on this method are beyond our scope.

Furthermore, this physical problem may also contain damping and could result in a generalized quadratic eigenvalue problem (zero excitation) or forced damped vibration (nonzero excitation), but the details of the derivation are more difficult and less enlightening. Since the goal of this example was to briefly demonstrate another real-life problem solvable by Lanczos's method, this avenue is not explored further.



Chapter 10
Linear systems and the Lanczos method

To end this book on a lighter note as well as to step back once again to the origin of the method, we review the application of the Lanczos method for solving linear systems.

10.1 Exact solution

Let us assume that A is symmetric, positive definite, and of order n and that the system we wish to solve is

$$Ax = b. \tag{10.1}$$

In theory the solution is

$$x = A^{-1}b; \tag{10.2}$$

however, calculating the inverse operator may be impractical. Again, we exploit the fact that the Lanczos method produces

$$Q_n^T A Q_n = T_n, \tag{10.3}$$

where Q_n is an orthogonal matrix. Therefore we can evaluate the inverse of A as

$$A^{-1} = Q_n T_n^{-1} Q_n^T \tag{10.4}$$

and find the exact solution

$$x = A^{-1}b = Q_n T_n^{-1} Q_n^T b. \tag{10.5}$$

This certainly seems more practical now, as we need to invert only a tridiagonal matrix. Let us forget for now the minor inconvenience of calculating the tridiagonal form and all the Lanczos vectors.

10.2 Approximate solution

Assuming that we seek only an approximate solution to the linear system (satisfying some error criterion), the Lanczos method enables the formulation of such a solution [3], [31], [35], [38].

We know that the stepwise nature of the Lanczos process results in a tridiagonal matrix T_j and in Lanczos vector block Q_j, $j = 1, \ldots, n$, after j steps of the process, which are related as

$$AQ_j = Q_j T_j + \beta_j q_{j+1} e_j^T. \tag{10.6}$$

Based on this we can have an approximate inverse and consequently an approximate solution:

$$x_j = Q_j T_j^{-1} Q_j^T b. \tag{10.7}$$

The issue now is the efficient calculation of the inverse of the tridiagonal matrix, for which Lanczos himself produced a clever algorithm. Let us assume the factorization of

$$T_j = L_j D_j L_j^T, \tag{10.8}$$

where

$$T_j = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \cdot & \cdot & \cdot & \\ & & & \beta_{j-1} & \alpha_j \end{bmatrix}, \tag{10.9}$$

and its factors are

$$L_j = \begin{bmatrix} 1 & & & & \\ \gamma_1 & 1 & & & \\ & \cdot & \cdot & \cdot & \\ & & & \gamma_{j-1} & 1 \end{bmatrix} \tag{10.10}$$

and

$$D_j = \begin{bmatrix} \delta_1 & & & \\ & \delta_2 & & \\ & & \cdot \cdot \cdot & \\ & & & \delta_j \end{bmatrix}. \tag{10.11}$$

The terms may be calculated by the following algorithm.

Algorithm 10.2.1. Tridiagonal matrix inverse algorithm

$\delta_1 = \alpha_1$

For $i = 2, \ldots, j$
  $\gamma_{i-1} = \beta_{i-1}/\delta_{i-1}$
  $\delta_i = \alpha_i - \beta_{i-1}\gamma_{i-1}$
End loop i

The algorithm is recursive in nature, and when going from Lanczos step j to $j+1$, one needs only to calculate the 2 terms in the loop for $i = j+1$. The algorithm as shown is for exact arithmetic and assumes that δ_{i-1} never becomes zero.

10.3 Recursive approximate solution

This idea of recursion is certainly appealing, and we attempt a formulation in this sense. Clearly (10.7) may be rewritten as

$$x_j = Q_j(L_j D_j L_j^T)^{-1} Q_j^T b. \tag{10.12}$$

Appropriately grouping, we get

$$x_j = \overline{Q}_j p_j, \tag{10.13}$$

where the scaled $\overline{Q}$ version of the Lanczos vectors is defined by

$$\overline{Q}_j L_j^T = Q_j. \tag{10.14}$$

The vector $p_j \in R^j$ satisfies

$$L_j D_j p_j = Q_j^T b. \tag{10.15}$$

Equation (10.14) in a columnwise fashion reads

$$\left[\begin{array}{cccc} \overline{q}_1 & \overline{q}_2 + \gamma_1 \overline{q}_1 & \cdot & \overline{q}_j + \gamma_{j-1}\overline{q}_{j-1} \end{array}\right] = \left[\begin{array}{cccc} q_1 & q_2 & \cdot & q_j \end{array}\right]. \tag{10.16}$$

This results in a recursive equation useful for our purposes:

$$\overline{q}_j = q_j - \gamma_{j-1}\overline{q}_{j-1}. \tag{10.17}$$

Assume that the p_j vector has components c_i, $i = 1, 2, \ldots, j$. Equation (10.15) in detail reads as

$$\left[\begin{array}{cccc} \delta_1 & & & \\ \delta_1\gamma_1 & \delta_2 & & \\ & \cdot & \cdot & \\ & & \delta_{j-1}\gamma_{j-1} & \delta_j \end{array}\right] \left[\begin{array}{c} c_1 \\ c_2 \\ \cdot \\ c_j \end{array}\right] = \left[\begin{array}{c} q_1^T b \\ q_2^T b \\ \cdot \\ q_j^T b \end{array}\right]. \tag{10.18}$$

Expressing the last row, we get

$$\delta_{j-1}\gamma_{j-1}c_{j-1} + \delta_j c_j = q_j^T b. \tag{10.19}$$

This leads to a recursive form of

$$c_j = \frac{q_j^T b - \gamma_{j-1}\delta_{j-1}c_{j-1}}{\delta_j}. \tag{10.20}$$

Finally, with the

$$p_j = \left[\begin{array}{c} p_{j-1} \\ c_j \end{array}\right] \tag{10.21}$$

partitioning, the recursive formula for the approximate solution is

$$x_j = \overline{Q}_j p_j = \overline{Q}_{j-1} p_{j-1} + c_j \overline{q}_j = x_{j-1} + c_j \overline{q}_j. \tag{10.22}$$

This formula has the clear advantage in computing the additional approximations in successive steps for only the cost of an inner product and some scalar operations.

10.4 Lanczos linear solver algorithm

The recursive formulae above and the preceding discussion result in the algorithm below.

Algorithm 10.4.1. Lanczos linear solver algorithm

$x_0 = \text{random}$
$r_0 = b - Ax_0$
$\beta_0 = ||r_0||$
$q_0 = 0$

For $j = 1, 2, \ldots$ until convergence

$q_j = r_{j-1}/\beta_{j-1}$
$\alpha_j = q_j^T A q_j$
$r_j = Aq_j - \alpha_j q_j - \beta_{j-1} q_{j-1}$
$\beta_j = ||r_j||$
If $\beta_j < \epsilon$ quit

If $j = 1$ then
$\delta_1 = \alpha_1$
$\overline{q}_1 = q_1$
$c_1 = \beta_0/\alpha_1$
$x_1 = c_1 \overline{q}_1$
Else
$\gamma_{j-1} = \beta_{j-1}/\delta_{j-1}$
$\delta_j = \alpha_j - \beta_{j-1}\gamma_{j-1}$
$\overline{q}_j = q_j - \gamma_{j-1}\overline{q}_{j-1}$
$c_j = q_j^T b - \gamma_{j-1}\delta_{j-1}c_{j-1}/\delta_j$
$x_j = x_{j-1} + c_j \overline{q}_j$
If $||Ax_j - b|| < \epsilon$ quit
Endif

End loop j

One of the two exit criteria of the algorithm is based on the $||Ax_j - b||$ error norm of the approximate solution. The other one is based on β_j. They are related to each other. In fact when β_j is zero (the Lanczos process breaks down), we have found an exact solution $Ax_j = b$.

The reader familiar with iterative solution methods will recognize that this algorithm is essentially the conjugate gradient method. The conjugate gradient method was published a couple of years later than Lanczos's seminal paper by two of Lanczos's coworkers at the National Bureau of Standards [21]. Popular belief says that their work was greatly influenced (to say the least) by Lanczos's method, and this is a fitting final testament to the influence of Lanczos's contribution.

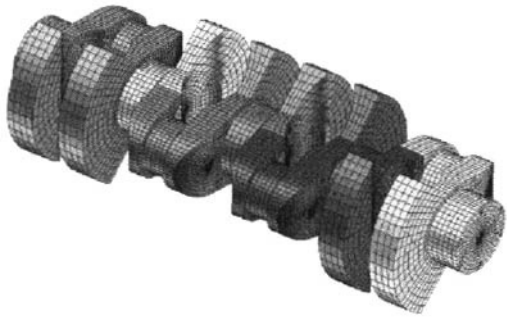


Figure 10.1. *Automobile crankshaft.*

10.5 Linear static analysis application

This application, most common in the industry, has the simplest equilibrium equation in the form of

$$Ku = F. \tag{10.23}$$

Its solution provides the engineer with the deformations of a structure, u, in response to static (neither time- nor frequency-dependent) loads, F. Here the K matrix represents the stiffness of the structure.

Such analyses are mainly executed in the auto and aerospace industries when analyzing the structural integrity of mechanical components. Today, these components are mainly modeled by CAD systems using various solid-modeling techniques. The finite element discretization of such solid models via automatic mesh generators results in a very high number of three-dimensional elements and high connectivity in the finite element models.

This fact leads to somewhat denser matrices, in contrast to the sparse matrices of the global shell models of a car body or an airplane fuselage. Direct solution of such systems requires enormous memory and disk resources due to the size of the factor matrices. On the other hand, iterative solutions such as the one described above are well suited for solving such problems. An example is the analysis of automobile crankshafts, as shown in Figure 10.1. The model consisted of 218,570 tetrahedral elements (a usual result of automatic mesh generators), resulting in 449,866 global degrees of freedom (rows in the matrix). Linear static analysis was executed with various load conditions using NASTRAN [48].

The direct solution of the analysis was not executed due to lack of disk space. The estimated maximum front size of the matrix was 10,112, and the estimated number of nonzero terms in the factor matrix was 1,495,312,000. This would have required over 10 GBytes of disk space, as every term is 8 bytes, and some additional index information that is also needed for the storage of factor matrices. While 10 GBytes isn't a very big disk storage these days, at times any particular user of a multiuser environment may not have this much space available.

The successful iterative solution was executed on an IBM RS/6000 Model 590 workstation using 1 GByte of memory and 8 GBytes of disk space. The solution was obtained with a block implementation of a preconditioned variant of the conjugate gradient algorithm presented above. It required 5,653.1 CPU seconds, or slightly more than one and a half hours. 2,952 iteration steps were executed and only 120 MBytes of memory was used, and the disk requirement was 4.3 GBytes, a rather respectable performance.

It is important to mention that the iterative method is also well adaptable to parallel computers. Due to the fact that the Schur complement calculation, a known bottleneck of the parallel direct solution approach, is omitted, the scalability is even better. In [43] a smaller version of the crankshaft example, containing about 150,000 degrees of freedom, is reported. The single-processor analysis took 38 minutes and 42 seconds, and the 8-processor run only 8 minutes and 35 seconds, a 4.6-fold speed-up in elapsed time. The disk space requirement was 1.4 GBytes and 0.26 GBytes, respectively, due to the fact that only a part of the global matrix is stored on any processor.

Closing Remarks

I hope that the reader enjoyed the journey through half a century of the evolution of a most influential method that is widely accepted in many industries. The major variations of the Lanczos method and the chapters dealing with applications were presented to help the reader find one method applicable to his or her problem.

Several interesting variations of the Lanczos method were not discussed in detail to keep the intended fluency. Such variations—not really new, and yet still not accepted industrially—are the look-ahead method [36] and the s-step method [26]. Several more recently published techniques like the rational Lanczos method [30], the multiple starting vector unsymmetric Lanczos method [8], and the so-called band Lanczos method [16], while very promising, have not yet been evaluated in the industry. Finally, some methods, like the thick-restart Lanczos method [42], are successful in some specific industrial segments (in this case, electronic), but do not work out in areas of highly demanding structural analysis, which was the application focus of this book. In regards to these issues the reader is referred to the references.

The references are organized into three categories. The first category [1]–[7] contains, in chronological order, the two original classical papers and several of the most popular books by Lanczos. Lanczos had about 135 publications in several countries (the United States, Ireland, Hungary, and Germany) on the two continents.

The second alphabetical category [8]–[42] mainly comprises the most influential publications related to the further development of the Lanczos method during the last three decades. The third section [43]–[50] contains publications related to the industrial implementation and commercial applications of the Lanczos method, demonstrating the method's success story almost unparalleled in engineering.

This page intentionally left blank

A Brief Biography of Cornelius Lanczos

Cornelius (Kornel) Lanczos was born on February 2, 1893 in Szekesfehervar, Hungary. After attending elementary and secondary schools in his hometown, he moved to Budapest to enroll in the University of Budapest (today Eotvos University of Sciences) to study mathematics, physics, and chemistry. He was mainly interested in physics in his early years and received his doctorate in 1921 for a dissertation related to Maxwell's equations and their function theoretical aspects.

Due to the worsening political atmosphere in Hungary, and to follow his professional interests, Lanczos left for Germany in the early 1920s. He spent the decade at various universities in Germany and had a deep collaboration with Einstein. He would later look back on those times with Einstein as the happiest hours of his life. Unfortunately the late 1920s and early 1930s brought political turmoil to Germany, and Lanczos again decided to leave.

He arrived in the United States and became a visiting professor at Purdue University for a year, which ultimately extended into a stay of almost 15 years. By the early 1940s Lanczos held three professorial titles at Purdue, among them professor of mathematical physics and professor of aeronautical engineering. The latter title led him to The Boeing Company during the war, where he gave lectures to engineers on various applied mathematics topics. In 1949 he joined the Institute for Numerical Analysis at the National Bureau of Standards, located at UCLA. During this time he wrote his first (now classical) textbook on variational principles of mechanics [1].

This was also the time when his method, the topic of this book, was published. Lanczos's eigenvalue computation paper is considered to be one of the most important papers ever written on matrix computations. While he was not able to fully alleviate the orthogonality problem of his method, even this aspect of it is also important. This loss of orthogonality issue inspired many excellent numerical analysts to further analyze his method, resulting in significant improvements and corollary discoveries.

Ironically, political circumstances (McCarthy investigations, etc.) again forced Lanczos to walk away from a well-established environment, and in the early 1950s he accepted the position of senior professor of the School of Theoretical Physics at the Dublin Institute for Advanced Studies in Ireland. It seems he found a well-deserved home for deep thinking and family life. Except for some visiting appointments at North American Aviation and at North Carolina State University, he remained in Ireland for almost two decades. During his

Dublin years Lanczos was devoted to full-time research and wrote several other great books [4], [5], [6], [7].

Finally, it seemed that fate would bring him back to his homeland when in June of 1974 he arrived in Budapest as a visiting professor at the Eotvos University of Sciences, the successor of his alma mater. However, only about a week after arriving, he was struck with a heart attack and died in a hospital on the morning of June 25. He was 81 years old. Under the still raging communist regime, his death was not national news, and on July 5 he was laid to rest quietly, in much the manner he had lived.

Bibliography

[1] Lanczos, C.; *The Variational Principles of Mechanics*, University of Toronto Press, 1949.

[2] Lanczos, C.; An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *Journal of the National Bureau of Standards*, Vol. 45, pp. 255–282, 1950.

[3] Lanczos, C.; Solution of systems of linear equations by minimized iteration, *Journal of the National Bureau of Standards*, Vol. 49, pp. 409–436, 1952.

[4] Lanczos, C.; *Applied Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1956.

[5] Lanczos, C.; *Linear Differential Operators*, Van Nostrand, New York, 1961; reprinted by SIAM, Philadelphia, 1996.

[6] Lanczos, C.; *Discourse on Fourier Series*, Oliver and Boyd, Edinburgh, 1966.

[7] Lanczos, C.; *Space through the Ages*, Academic Press, New York, 1970.

---

[8] Aliaga, J. I., Boley, D. L., Freund, R. W., and Hernández, V.; A Lanczos-type method for multiple starting vectors, *Mathematics of Computation*, Vol. 69, pp. 1577–1601, 2000.

[9] Bai, Z., Day, D., and Ye, Q.; ABLE: An adaptive block Lanczos method for non-Hermitian eigenvalue problems, *SIAM Journal on Matrix Analysis and Applications*, Vol. 20, pp. 1060–1082, 1999.

[10] Bai, Z. and Freund, R.; A partial Padé-via-Lanczos method for reduced order modeling, *Linear Algebra and Its Applications*, Vols. 332–334, pp. 139–164, 2001.

[11] Cullum, J. K. and Willoughby, R. A.; *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.* 1: *Theory*, SIAM, Philadelphia, 2002.

[12] Day, D.; An efficient implementation of the nonsymmetric Lanczos algorithm, *SIAM Journal on Matrix Analysis and Applications*, Vol. 18, pp. 566–589, 1997.

[13] Duff, I. and Reed, J.; The multifrontal solution of indefinite sparse symmetric linear equations, *ACM Transactions on Mathematical Software*, Vol. 9, pp. 302–325, 1983.

[14] Ericsson, T. and Ruhe, A.; The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems, *Mathematics of Computation*, Vol. 35, pp. 1251–1268, 1980.

[15] Francis, J. G. F.; The QR transformation, I, II, *Computer Journal*, Vol. 4, pp. 265–271, 332–345, 1961.

[16] Freund, R.; Band Lanczos method, in *Templates for the Solution of Algebraic Eigenvalue Problems*, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, 2000, pp. 80–88.

[17] Givens, W.; Computation of plane unitary rotations transforming a general matrix to triangular form, *Journal of the Society for Industrial and Applied Mathematics*, Vol. 6, pp. 26–50, 1958.

[18] Golub, G. H. and Van Loan, C. F.; *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.

[19] Golub, G. H. and Underwood, R.; The block Lanczos method for computing eigenvalues, in *Mathematical Software* III, J. Rice, ed., Academic Press, New York, 1977, pp. 364–377.

[20] Grimes, R. G., Lewis, J. G., and Simon, H. D.; A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems, *SIAM Journal on Matrix Analysis and Applications*, Vol. 15, pp. 228–272, 1994.

[21] Hestenes, M. R. and Stiefel, E.; Method of conjugate gradients for solving linear systems, *Journal of the National Bureau of Standards*, Vol. 49, pp. 409–436, 1952.

[22] Householder, A. S.; Unitary triangularization of a nonsymmetric matrix, *Journal of the Association for Computing Machinery*, Vol. 5, pp. 339–342, 1958.

[23] Kahan, W. and Parlett, B. N.; *How Far Should You Go with the Lanczos Process?*, Research Report, University of California at Berkeley, 1987.

[24] Karypis, L. and Kumar, V.; *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*, Tech. report, TR 95-035, Department of Computer Science, University of Minnesota, Minneapolis, 1995.

[25] Kent, M.; *Chebyshev, Krylov, Lanczos: Matrix Relationships and Computations*, Ph.D. thesis, STAN-CS-89-1271, Stanford University, Stanford, CA, 1989.

[26] Kim, S. K. and Chronoloulos, A. T.; *The s-Step Lanczos and Arnoldi Method on Parallel*, Tech. report, TR 89-83, IT University of Minnesota, Minneapolis, 1989.

[27] Komzsik, L.; Optimization of finite element software systems on supercomputers, in *Supercomputing in Engineering Analysis*, H. Adeli, ed., Marcel Dekker, New York, pp. 261–288, 1992.

[28] Komzsik, L. and MacNeal, R. H.; Speeding up the Lanczos algorithm, in *Dynamic Behavior of Concrete Structures*, RILEM, Bratislava, Slovakia, pp. 23–30, 1995.

[29] Komzsik, L.; Implicit computational solution of quadratic eigenvalue problems, *Finite Elements in Analysis and Design*, Vol. 37, pp. 799–810, 2001.

[30] Meerbergen, K.; Changing poles in the rational Lanczos method for the Hermitian eigenvalue problem, *Numerical Linear Algebra with Applications*, Vol. 8, pp. 33–52, 2001.

[31] Nour-Omid, B., Parlett, B. N., and Raefsky, A.; *Comparison of Lanczos with Conjugate Gradient Using Element Preconditioning*, Lockheed Palo Alto Research Labs, 1985.

[32] Paige, C.; Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix, *Journal of the Institute of Mathematics and Its Applications*, Vol. 18, pp. 341–349, 1976.

[33] Parlett, B. N. and Scott, D. S.; The Lanczos algorithm with selective orthogonalization, *Mathematics of Computation*, Vol. 33, pp. 217–238, 1979.

[34] Parlett, B. N.; *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980; reprinted by SIAM, Philadelphia, 1998.

[35] Parlett, B. N.; A new look at the Lanczos algorithm for solving symmetric systems of linear equations, *Linear Algebra and Its Applications*, Vol. 29, pp. 323–346, 1980.

[36] Parlett, B. N. and Taylor, D.; *A Look Ahead Lanczos Algorithm for Unsymmetric Matrices*, Department of Mathematics, University of California at Berkeley, 1983.

[37] Parlett, B. N. and Chen, H.; Use of indefinite pencils for computing damped natural modes, *Linear Algebra and Its Applications*, Vol. 140, pp. 53–88, 1990.

[38] Saad, Y.; On the Lanczos method for solving symmetric linear systems with several right-hand sides, *Mathematics of Computation*, Vol. 48, pp. 651–662, 1987.

[39] Simon, H.; The Lanczos algorithm with partial reorthogonalization, *Mathematics of Computation*, Vol. 42, pp. 115–142, 1984.

[40] Tisseur, F.; Backward error analysis and condition of polynomial eigenvalue problems, *Linear Algebra and Its Applications*, Vol. 309, pp. 339–361, 2000.

[41] Wilkinson, J. H.; The calculation of eigenvectors of codiagonal matrices, *Computer Journal*, Vol. 1, pp. 90–96, 1958.

[42] Wu, K., Canning, A., Simon, H. D., and Wang, L.-W.; Thick-restart Lanczos method for electronic structure calculations, *Journal of Computational Physics*, Vol. 154, pp. 156–173, 1999.

---

[43] Komzsik, L. et al.; A breakthrough in parallel performance in MSC.NASTRAN V70.7, in *Proceedings of the 1st MSC World Wide Automobile Users' Conference*, Munich, Germany, 1999.

[44] Komzsik, L.; Computational acoustic analysis in the automobile industry, in *Automotive Technology and Automation*, ISATA, Florence, Italy, pp. 11–22, 1996.

[45] Komzsik, L.; Computational analysis of automobile brakes, in *Vehicle Simulation and Supercomputing*, ISATA, Düsseldorf, Germany, pp. 17–24, 1998.

[46] Komzsik, L.; *MSC/NASTRAN Numerical Methods User's Guide*, 6th ed., The MacNeal-Schwendler Corporation, Los Angeles, 2000.

[47] Komzsik, L.; High performance normal modes analysis, in *Proceedings of the European Conference on Computational Mechanics*, Cracow, Poland, 2001.

[48] Komzsik, L., Poschmann, P., and Sharapov, I.; A preconditioning technique for indefinite linear systems, *Finite Elements in Analysis and Design*, Vol. 26, pp. 253–258, 1997.

[49] Komzsik, L. and Rose, T.; Substructuring in MSC/NASTRAN for large scale parallel applications, *Computing Systems in Engineering*, Vol. 2, pp. 167–173, 1991.

[50] Komzsik, L. and Chiang, K. N.; The effect of a Lagrange multiplier approach in MSC/NASTRAN on large-scale parallel applications, *Computing Systems in Engineering*, Vol. 4, pp. 399–403, 1993.

Index