Progress in the Numerical Solution of the Nonsymmetric Eigenvalue Problem*

Dedicated to W. Kahan and B. N. Parlett on the occasion of their 60th birthdays

Zhaojun Bai †

October 17, 1992

Abstract

With the growing demands from disciplinary and interdisciplinary fields of science and engineering for the numerical solution of the nonsymmetric eigenvalue problem, competitive new techniques have been developed for solving the problem. In this paper, we examine the start-of-the-art of the algorithmic techniques and the software scene for the problem. Some current developments are also outlined.

1 Introduction

Over several years working on the LAPACK project [2], and on algorithm and software development of the nonsymmetric eigenvalue problem and communication with a variety of users who work in diverse fields involving scientific computing, the author have seen a growing demand for the numerical solution of the nonsymmetric eigenvalue problems. Meanwhile, in numerical analysis community, since Parlett's exploratory review paper entitled "The Software Scene in the Extraction of Eigenvalues from Sparse Matrices" [34] nearly one decade ago, and with the successful development of the symmetric eigenvalue problem, many new numerical methods and analysis have been developed for the nonsymmetric eigenproblem. The aim of this essay is to review the origins of the problem and the progress of the numerical techniques over the past decade, and to share our view and expertise within scientific computing community.

The survey is by no means complete. One reason for this is that relevant articles may be found scattered throughout the scientific and engineering literature, and the task of tracking them all down is impossibly large. The author apologizes for the ignorance of some important contributions to the problem that are not mentioned here. A new book by Saad [44] is an elegant source for studying the start-of-the-art in large eigenproblem techniques. This review will only focus on the nonsymmetric eigenvalue problem in the aspects of its origins, *algorithmic* techniques, software scene and work in progress.

As defined by Parlett [34] one decade ago, there are two different user groups for the eigenproblem. One is called *intensive* user group and the other called *sporadic* user group.

*This work was supported in part by NSF grant ASC-9102963 and in part by the Applied and Computational Mathematics Program, Defense Advanced Research Projects Agency, under contract DM28E04120.

†Department of Mathematics, University of Kentucky, Lexington, KY 40506, U.S.A, e-mail address: bai@ms.uky.edu

For the former group, spectral analysis is imperative to their entire work; they have been spending tremendous efforts in terms of times and funding for extracting the desired spectral information. But for the latter group, the need to compute eigenvalues arises occasionally and the user wants to obtain them with minimal fuss.

With the rapid advances of computer facilities, in particular the massively parallel computers, and the new engagement of interdisciplinary scientific computing activities, as proposed by J. W. Demmel, there are two different camps in each user group according to their different desired priorities, in terms of computation details, reliability and execution time, of a program. The first camp is made up of traditional library users, and the second camp is made up of high performance computing researchers. For the first camp, the desiderata can be characterized as follows:

1. easy user interface with hidden computation details,
2. reliability; the code should fail as rarely as possible,
3. execution time.

However for the second camp, the desiderata are

1. execution time,
2. be able to access to internal details to tune data structures to their applications,
3. reliability; A program should expend only a negligible amount of time, space or code in checking or taking precautions against rare eventualities that the user knows may never arise for his or her particular applications.

These different desiderata give an extra dimension to numerical algorithm development and analysis. To what extent can we satisfy both camps? In this essay, we will try to address this interesting question with respect to the nonsymmetric eigenvalue problem.

There are no clear territories for *small* and *large* problem, *dense* and *sparse* matrices. All of them are relative and are changing rapidly with the advances of algorithms and computer technology. A matrix of order a couple of hundred used to be considered large, but on today's high performance workstations and supercomputers it is small. The list of today's performance and benchmarks for an algorithm on high performance computers often starts with the sizes of matrices over a couple of hundreds and up to thousands. A matrix is considered to be sparse if only a small percentage of its entries are nonzero. However, concerning the tradeoff between the sparsity and algorithm complexity and performance, we do not necessarily exploit the zeros. Following Parlett's suggestion in his 1984 paper [34], "it is vital to science that one not be more precise than is necessary for the purpose in hand", we will leave user to decide whether his or her problem is small or large, dense or sparse.

Finally, to numerical analysts, it is well known that the techniques for treating dense and sparse matrices, small and large problems are closely related. An inner loop of a method for large and sparse eigenproblem often constitutes a small and dense problem. The small problem solver could be a bottleneck for the large problem.

The rest of the paper is organized as follows: Section 2 reviews the main features of the nonsymmetric eigenvalue problem solvers in the LAPACK. Sections 3 and 4 discuss the cases where the users want to have more than what the LAPACK can offer. Section 5 describes algorithms and software scene for the sparse nonsymmetric eigenvalue problem.

Section 6 discusses some contributions in the last decade which have become more widely appreciated. Sections 7 and 8 describe some work in progress and conclusions.

2 The nonsymmetric eigenproblem in LAPACK

The LAPACK [2] is a transportable public domain library of Fortran 77 subroutines for solving the most common matrix computation problems, such as solving linear systems, least squares problems, eigenproblems and so on. The library was mainly supported by NSF beginning in 1987. It is a further development of the successful LINPACK and EISPACK libraries [15, 46] that were developed in the 1970s. A great deal of effort has been expended to improve accuracy and robustness, to add new functionality, and to incorporate design methodologies and algorithms that make LAPACK appropriate for today's high performance shared memory vector and parallel computer architectures. The design, implementation and performance evaluation of LAPACK has attracted significant interest from user communities and from computer manufacturers.

LAPACK is said to be "transportable" rather than "portable", because the fastest performance requires that highly optimized block matrix operation already be implemented on each machine by the manufacturer or someone else. Nevertheless, the algorithms used in LAPACK are so well specified and understood that they have been turned into "black box" programs demanding no preknowledge of where the eigenvalues are *likely* to lie, and no tricky choices of tolerances or accuracy.

There are three levels of subroutine in LAPACK[1] to fully address the standard nonsymmetric eigenvalue problem:

1. **Driver routines** for nonexpert and expert: These drivers solve a complete problem. User can choose to compute the eigenvalues and left and/or right eigenvectors, or Schur decomposition. In addition, with the expert drivers, a balancing transformation can be used to improve the conditioning of the eigenvalues and eigenvectors, the user can choose to compute condition numbers for the eigenvalues and the right eigenvectors, to order the eigenvalues on the diagonal of the Schur form, and to compute a condition number for the average of the selected eigenvalues and for the right invariant subspace corresponding to the selected eigenvalues.

2. **Computational routines**: These subroutines perform distinct computational tasks, such as Hessenberg reduction, and the Schur decomposition of Hessenberg form, inverse iteration for selected eigenvectors, condition number estimation.

3. **Auxiliary routines**: Auxiliary routines in this level perform relatively low level operations, such as unblocked Hessenberg reduction, swapping the adjacent diagonal (block) entries in Schur form, solving 2 by 2 eigenproblems and so on.

These LAPACK subroutines are not just more efficient updates of their popular predecessor EISPACK. In order to obtain the better performance of LAPACK subroutines on a wide range of shared memory vector and parallel processors, some new algorithms have been designed to fit the memory access patterns with respect to the multilayered memory hierarchies of the machines. For example, a block Hessenberg reduction algorithm has been

[1]Individual routines from LAPACK can be obtained by electronic mail through *netlib*: netlib@ornl.gov or netlib@research.att.com. The complete package can be obtained on magnetic media from NAG at Jordan Hill Road, Oxford OX2 8DR, England.

adapted in LAPACK for reducing the original matrix a condensed form by orthogonal transformation. Following the reduction to condensed form, a multishift QR algorithm has been developed in order to using block matrix operations. The traditional EISPACK routine HQR uses a double shift; the multishift strategy uses block shifts of higher order and can perform matrix-vector operations instead of vector-vector operations, and obtains modest speed up.

Along with the new goals of LAPACK in terms of stability, efficiency, accuracy and robustness, LAPACK also provides much wider functionality for the nonsymmetric eigenvalue problem, including computing the Schur form and Schur vectors, ordering selected eigenvalues in Schur form, and computing right and/or left eigenvectors. Last, but not least, with the new condition number estimation techniques, we can produce condition numbers along with the basic solutions. This is a valuable by-product of the successful perturbation theory conducted by many numerical analysts over three decades.

All LAPACK subroutines have been through massive testing and performance evaluation on a large class of modern computers with help of over 40 test sites from USA to Europe. The critiques and comments from the potential user communities and computer vendors have improved the programs in many aspects.

With the large and growing fast storage capacity of today's high performance computers, the domain of the LAPACK grows. The author has used a LAPACK driver routine to find all the eigenvalues of a 2000 by 2000 matrix, a computation that would have been difficult or impossible 10 years ago. It took about 575 seconds CPU time on a CRAY-2 machine.

Does LAPACK satisfy all user groups and camps? Who will then remain unsatisfied with LAPACK for their applications? What has been missed in LAPACK? We will try to address these questions in the following sections.

3 Why we remain unsatisfied with LAPACK

Without doubt, LAPACK subroutines will be used extensively in the coming years. For those sporadic and traditional library users of eigenvalue solvers, whose matrices can be stored in-core, who think reliability is more important than execution time, LAPACK subroutines are their desired answers. However, for those intensive users and high performance competitors, LAPACK does not provide all they are looking for. The following are the main complaints on the nonsymmetric eigenproblem routines.

The penalty by exceptional handling: The speed of some subroutines is penalized by spending a considerable amount of time, space, or code checking for rare events. If the eigenvalue computation is the inner loop of large computation, such a penalty could significantly slow the computation, even though the users (who are more interested in speed) know that their problems are very well-conditioned. For example, in order to resist the possible overflow, we cannot use the standard Level 2 BLAS triangular solver in computing the eigenvectors of a triangular matrix and in inverse iteration for the selected eigenvectors, because there is no protection against overflow in the BLAS routine. Instead, we had to develop another much complicated triangular solver including sophisticated scaling in the inner loop: it has 300 lines of Fortran versus the standard Level 2 BLAS code with 159 lines of Fortran (not including comments). This gives us a double performance penalty, since we cannot use optimized BLAS, and since we must do many more floating point operations and logical tests.

Without partial eigenvalue handling: There are eigenvalue problem applications where the users are only interested in the eigenvalues and their corresponding the invariant subspace in a specified region on complex plane, such as in a vertical strip including the imaginary axis. However, the LAPACK routines always compute all eigenvalues of the matrix. The users have to sort them out by themselves.

Single data structure and no sparsity handling: The main drawback of LAPACK is that it does not try to exploit sparsity structure in the original matrix. A square matrix is stored in essentially one standard conventional two dimensional data structure. Many matrices arising in practice are too large to be stored in-core. In this case, in order to be able to tackle larger and larger problems, it is necessary to take advantage of the sparsity of the original matrix to reduce memory requirements and computation time. In the next section, we will examine the origins of large scale eigenvalue problems in science and engineering to illustrate the need for exploiting the sparsity of the matrix.

4 Where are the origins of large eigenvalue problems?

For small matrix eigenproblems, the algorithm and software are in such good standing that we can solve the problem efficiently without need to know where the problems come from. However, for the large problem, it is still a research subject, and is the frontier of our expertise. Therefore, before we consider an algorithm for the problem, it does help to know where the problem comes from and what the users really want.

In [34], Parlett discusses eigenproblems from structural engineering, quantum chemistry, and plasma physics. Most of them are the symmetric eigenproblems. In addition, the application problems from Markov chain modeling and numerical simulation of certain chemical reactions from the literature of Stewart [50] and Saad [40] have become the standard test problems for a numerical method. In this section, we will examine some newly arising nonsymmetric eigenvalue problems from the scientific computing community.

Power system stability study [33]: Low damped electromechanical oscillations have become a common phenomenon in modern electric power systems. The damping of these oscillations is dependent on system structure, operating conditions, and the effects of automatic-controller action. An efficient way to combat these problems is through the installation of additional signals to the generator excitation systems. In recent years, this area of work has led to the development of numerical methods to determine the source of these problems and obtain solutions via excitation control.

The power system stability problem is represented by a set of algebraic differential equations. If we write the Jacobian matrix of the entire set of equations evaluated at an operating point as

$$J = \begin{pmatrix} J_1 & J_2 \\ J_3 & J_4 \end{pmatrix},$$

then the power system state matrix is then defined as

$$A = J_1 - J_2 J_4^{-1} J_3.$$

The dominant eigenvalues of the matrix A determine the stability of the nonlinear system. In practice, the matrix J is very large and sparse, but the state matrix A is not sparse. The problem is to calculate the desired eigenvalues of A without forming it explicitly. The author is informed that a group of researchers under the leadership of Martins is developing an efficient algorithm on an Intel distributed memory machine.

Numerical analysis of optical waveguides [45]: In recent years, an accurate numerical analysis for optical waveguides is of practical interest for optimum design of optical devices and optical integrated circuits. Under certain conditions, the propagation constant β and electric field $\phi(x, y)$ in the waveguide are described by the scalar Helmholtz equation of the form

$$(\nabla^2 + k^2 n^2(x, y) - \beta^2)\phi(x, y) = 0,$$

where k is the wave number and $n(x, y)$ is the refractive index. The discretization with the finite difference method on a nonuniform mesh converts the equation to the eigenvalue problem

$$A\phi = \beta^2 \phi,$$

where A is a banded matrix, and ϕ is a column vector whose components represent the electric field at each cell. The eigenvalues corresponding to the bound modes range between the refractive index of the channel layer and the refractive index of the cladding layer. In other words, only those eigenvalues $\Re(\beta^2)$ in a certain interval are of interest.

A recent report by Galick *et al* [21] presents a numerical approach to the simulation of dielectric channel waveguide and is based on the solution of an eigenproblem for two transverse components of the magnetic field.

The stability analysis of viscous free surface flow [11]: Slide coating is an industrial precision coating operation. A liquid film is formed on an inclined plane, flows down and off the plane and onto a moving sheet displacing air as it wets the smooth solid surface of the sheet. The desired liquid flow is steady and uniform across the width of the sheet, and governed by the Navier-Stokes system for two-dimensional, incompressible flow. Under the momentum conservation, mass conversation and kinematic boundary condition, by means of Galerkin's method and finite element basis functions, one has a set of nonlinear algebraic differential equations of the form

$$f(y, \dot{y}, p) = 0,$$

where y is the vector of all nodal unknowns, i.e., the coefficients of the finite basis element basis functions, p is the vector of physical and discretization parameters. The prototype problem examined in [11] is the asymptotic stability of the steady to infinitesimal disturbances,. Small disturbances y_1 imposed on a steady state y_0 are governed by the linearized equation (up to the first order)

$$f_{\dot{y}}(y_0, 0, p)\frac{\partial y_1}{\partial t} + f_y(y_0, 0, p)y_1 = 0.$$

The fate of y_1 is then given by

$$y_1(t) = \sum_{i=1}^{N} \epsilon_i \exp(\lambda_i t) x_i,$$

where x_i is a generalized right eigenvector associated with eigenvalue λ_i, i.e.,

$$f_y(y_0, 0, p)x_i = -\lambda_i f_{\dot{y}}(y_0, 0, p)x_i, \qquad (1)$$

and ϵ_i is the component of y_1 along x_i at $t = 0$. Let $J = f_y(y_0, 0, p)$ and $M = -f_{\dot{y}}(y_0, 0, p)$; then (1) is a generalized eigenproblem of the form

$$Jx = \lambda Mx$$

where both J and M are of order 2000 to 4000, nonsymmetric, banded, sparse and singular. A steady state is stable if all eigenvalues have negative real parts. The computational task is to find the "dangerous eigenmode", i.e., the eigenvalue with the largest real part.

Recently, [8, 9] have reported the stability analysis of 3-dimensional patterned states in the normal field instability of ferrofluids. The size of the related eigenvalue problem, when all possible patterns are "allowed" (by the boundary conditions) to compete, is of the order of 100,000!

Navier-Stokes Solver [19, 32] The eigensystem analysis is associated with small perturbation of a finite different representation of the Navier-Stokes equations for analyzing complex flows such as unsteady, transonic, viscous flow over airfoils. Such eigensystem information is used to reduce the number of time steps required to reach a steady solution and accelerate the convergence [19], to determine the modal behavior of fluid in a fluid-structure interaction problem [32]. The matrices involved in such problems are very large (up to $24,000 \times 24,000$), real, banded sparse and unsymmetric. The interesting eigenvalues are the ones with small negative real parts, or the dominant ones.

In [22], an eigenproblem problem solver is required to the numerical detection of Hopf bifurcation of parameter dependent nonlinear system modeling a tubular reactor. Chatelin *et al* [10] shows an eigenproblem from aeronautical industries. The interesting eigenvalues are those whose imaginary part lies in a frequence range chosen by the engineer. The matrix is sparse with a block structure and has order 10^3 to 10^4. In [35], an interesting eigenproblem of a transfer matrix from the Ising model, used in chemistry and biology, is discussed. The matrix is nonsymmetric, very sparse (only have two entries at each column and row) and huge (at order of 2^n). The desired eigenvalues are the two largest ones.

5 Software scene for sparse nonsymmetric eigenproblem

There has been a significant progress for the large symmetric eigenvalue problem over the last two decades. The Lanczos algorithm has become the the most favored algorithm. Today you can get the software through public domain or commercial package. However, the large nonsymmetric eigenvalue problem is still a research topic. We are still trying to better understand the problem, the existing algorithms and their numerical behaviors.

In this section, we review the progress in the numerical solution of the problem in the past decade. We focus on three popular algorithms which are mostly studied in the numerical analysis community. We also list information on the software production which is developed mostly by numerical analysts, but it should be noted that massive testing phase has not applied for the software. Some of the software are still research codes. The experts are still searching for the best way to implement the algorithms.

In sparse matrix computations, it is widely accepted that it is important to reference the matrix in question only through a user provided subroutine to form matrix-vector products. All algorithms and software discussed in this section have such a feature. In some applications this essential operation could be the major cost of the overall computation. The exploitation of the structure and sparsity of the matrix are passed to the users. The performance of these operations on supercomputers can differ significantly from one problem data structure to the others.

The author should note that beside the three popular methods discussed in this section, there are some other methods, such as Goldhirsch-Orszag-Maulik's method [23] proposed for solving the hydrodynamic and mechanical stability analysis, and the generalization of the popular Davidson's method for a general matrix in computational chemistry [13], have not been reviewed. It is sad to say that not much attention has been paid to these algorithms in numerical analysis community.

5.1 Algorithms

This section gives a brief description of the main ideas of the three popular methods. There is an extensive literature on these methods, too many to list in this essay. The reader is referred to [24, 44].

Simultaneous iteration method, whose prototype is Bauer's *Treppeniteration*, are the extension of the power method. It is still one of the most popular methods used for computing the dominant eigenvalues of large matrices, in particular, in the structure engineering. The iteration starts with an $n \times m$ matrix Q_0 and generates a sequence of $n \times m$ matrices Q_k according to the formula

$$AQ_{k-1} = Q_k R_k,$$

where R_k is a nonsingular matrix chosen variously by the different implementations. Since the column space of Q_k and $Q_k R_k$ are the same, R_k can be regarded as a scaling factor. Then one forms a projection matrix: $B_{k+1} = Q_k^T A Q_k$, and reduces it to $B_{k+1} = Y_k T_k Y_k^{-1}$, which is an eigendecomposition or Schur decomposition of B_{k+1}, finally overwrites Q_k by $Q_k Y_k$. It has been proved that the rate of the convergence of the ith column of Q_k depends on the ratio $|\lambda_{m+1}/\lambda_i|$, where we assume that the eigenvalues $\{\lambda_i\}$ of A are ordered so that $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$.

The following Arnoldi and Lanczos methods are in the class of Krylov subspace techniques. Members of this class generate (bi-)orthogonal basis for Krylov subspace $K_m(q, A) = \{q, Aq, A^2 q, \ldots, A^{m-1}q\}$ (and $K_m(p, A^T)$) with the given initial vector(s) q (and p), and then compute eigenvalues of reduced matrix obtained from A by restricting A to the Krylov subspace. A subset of the computed eigenvalues is selected as approximations to the eigenvalues of the original matrix. Specifically, the Arnoldi algorithm first produces an orthonormal basis of the Krylov subspace $K_m(q, A)$, which can also be regarded as a truncated reduction of an $n \times n$ matrix A to upper Hessenberg form. Starting with a given vector q_1 of $\|q_1\|_2 = 1$, after j steps, we have

$$AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T,$$

where $Q_m^T Q_m = I$, $Q_m^T q_{m+1} = 0$, $H_m \in \mathbb{R}^{m \times m}$ is an upper Hessenberg matrix. Then the eigenpairs $\{(\theta_i, z_i)\}$ of the reduced matrix H_m are computed. The approximate eigenpairs of A are selected from the so called Ritz pairs $\{(\theta_i, Q_m z_i)\}$.

The Lanczos process is a procedure to generate the biorthogonal bases of the Krylov subspaces $K_m(q, A)$ and $K_m(p, A^T)$ that can be regarded as a truncated procedure for the tridiagonalization of a general matrix. In matrix notation, starting with given vectors p_1 and q_1 and $(p_1, q_1) \neq 0$, after m steps, we have

$$\begin{aligned} AQ_m &= Q_m T_m + \beta_{m+1} q_{m+1} e_m^T, && (2) \\ A^T P_m &= P_m T_m^T + \gamma_{m+1} p_{m+1} e_m^T, && (3) \end{aligned}$$

where $P_m^T Q_m = D_m$, D_m is a diagonal matrix. There are certain free choices in the procedure to force D_m to be identity matrix, or $\|p_i\|_2 = \|q_i\|_2 = 1$. Then eigentriplets $\{(\theta_i, z_i, w_i)\}$ of the reduced matrix pencil (T_m, D_m) are computed, The approximate eigentriplets of A are selected from the Ritz triplets $\{(\theta_i, Q_m z_i, P_m w_i)\}$.

Transformation functions	Operation	References
polynomial $p(x)$	$p(A)v$	[39, 40]
shifting-and-inverting $(x-\mu)^{-1}$	$(A-\mu)^{-1}v$	[36]
Cayley transformation $c(x)=(x-\alpha)/(x+\alpha)$	$(A+\alpha I)^{-1}(A-\alpha I)v$	[7, 11, 22]
rational function $r(x)=p(x)/q(x)$	$r(A)v$	[37, 38]
exponential function e^x	$e^A v \approx p(A)v \approx r(A)v$	[19, 11, 43]

Table 1: Spectrum Transformation

A user should note that for simultaneous iteration or Arnoldi method, a subroutine to generate the vector $y = Ax$ for a given vector x has to be provided. However, for the Lanczos method, subroutines to generate both the vectors $y = Ax$ and $z = A^T x$ have to be provided.

5.2 Spectrum transformation

Unfortunately, each algorithm favors a particular part of the spectrum which may not be what the user wants. Simultaneous iteration favors the dominant eigenvalues, whereas the Arnoldi and Lanczos algorithms prefer the outer part of the spectrum. This is where the spectrum transformation comes into the picture. The spectrum transformation strategy (sometimes called preconditioning techniques in the notion of linear system) is to transform the desired eigenvalues in the complex plane towards the favorable part of the chosen algorithm in the hope of accelerating the convergence rate and separating the wanted eigenvalues from the unwanted ones, leading to better conditioning.

Many different spectrum transformation techniques have been tried over the years. Table 1 lists some of known spectrum transformation functions, the involved matrix-vector operations and references. It is clearly a problem dependent strategy. The user's knowledge of where eigenvalues are *likely* to lie can be used to choose the transformation function and make the solution easier.

5.3 Software scene

Table 2 summarizes the software scene for the large nonsymmetric eigenvalue problem, and the author apologizes for the ignorance of any other existing good implementation of the algorithms. As we have mentioned, most programs are still research codes, and are subject to change. The authors of these programs reserve all rights. The reader should not judge programs by the number of Fortran lines. Some software are not in public release format. The line count just gives a rough feeling about how much effort is needed to write the codes.

Let us sketch the main features of each program. The first three programs in Table 2 are three different implementation based on the simultaneous iteration:

1. `LOPSI` by Stewart and Jennings [51] is based on bi-iteration for computing the set of eigenvalues of largest absolute magnitude of a matrix together with the corresponding left and right eigenvectors.

2. `SRRIT` by Bai and Stewart [6] computes a partial Schur decomposition corresponding to the eigenvalues of largest modulus, instead of the computing the partial eigen-decomposition of `LOPSI` . If explicit eigenvectors are desired, they may be obtained

Authors	Year	Name	Method	Lines	References
Stewart-Jennings	81	`LOPSI`	Simul. iter.	1,054	[51]
Stewart-Bai	78,91	`SRRIT`	Simul. iter.	2,417	[50, 6]
Duff-Scott	91	`EB12`	Simul. iter.	3,091	[18]
Saad	85	`ARNLS`	Arnoldi	1,783	[41]
Saad	86	`ARNINV`	Arnoldi	2,619	[41]
Sorensen	91	`ARNUPD`	Arnoldi	2,743	[47]
Cullum-Willoughby	86	`NSLEVAL`	Lanczos	3,021	[12]
Freund-Nachtigal	90	`EIGLAL`	Lanczos	2,375	[20]

Table 2: Software Scene for Sparse Nonsymmetric Eigenvalue Problem

from the Schur form, say using LAPACK subroutine `STREVC`. The user guide report of `SRRIT` is available by anonymous ftp from `thales.cs.umd.edu` in the directory `pub/reports`. The program is available in `pub/srrit`.

3. `EB12` by Duff and Scott [18] is the latest entry to the simultaneous iteration, which can be found in the Harwell Subroutine Library. The Chebychev acceleration technique proposed by Saad is used in the `EB12` subroutine. It also has some features such as optional computation of eigenvalues of right-most or left-most real part, or largest modulus.

The programs `ARNLS` , `ARNINV` and `ARNUPD` in Table 2 are variant implementations of Arnoldi method. To restrict the required storage, all programs use a fixed number of steps in the Arnoldi process, and restart the process if necessary. Specifically,

1. `ARNLS` by Saad [41] is an implementation of the polynomially preconditioned Arnoldi method with deflation. The subroutine computes only one eigenvalue or a complex conjugate pair at a time. Deflation is then used to compute the next desired eigenpair until satisfied. For the next wanted eigenvalue, a polynomial transformation is made to transform the desired eigenvalue to be very large compared with its remain eigenvalues.

2. `ARNINV` by Saad [41] is an Arnoldi method with shifting-and-inverting preconditioning strategy for computing the desired number of eigenvalues and associated schur vectors closest to the shift σ of a banded matrix A. The matrix A is passed in banded format and then $A - \sigma I$ is factorized by LINPACK which is then used as the operator for Arnoldi method. Although this operator may be complex, the main computations are done in real arithmetic [36]. User can decide to use a new shift after a certain number of restarted Arnoldi when computing some eigenvalue.

3. `ARNUPD` by Sorensen [47] treats the residual vector, after a run of Arnoldi process, as a function of the initial vector. This initial vector is then updated through a chosen (polynomial) filter that is designed to force convergence of the residual to zero. The iterative scheme is shown to be a truncation of the standard implicitly shifted QR-iteration for the dense problem, and it avoids the need to explicitly restart the Arnoldi sequence.

There are two entries based on the nonsymmetric Lanczos method:

1. `NSLEVAL` by Cullum and Willoughby [12] is a Lanczos procedure with no reorthogonalization and no handling of possible breakdown. The procedure produces a complex symmetric tridiagonal matrix, whose eigenvalue problem is solved by a modified QL algorithm.

2. `EIGLAL` by Freund and Nachtigal [20] is a modified Lanczos procedure with look-ahead strategy to skip over the possible breakdown. The procedure produces a Hessenberg matrix, and then its eigenproblem is solved by the QR algorithm from EISPACK.

Both implementations give rise to "spurious" eigenvalues. In order to distinguish between "good" and "spurious" eigenvalues, the most common empirical way of doing this is to calculate eigenvalues of T_m for some m, and then repeat the calculation for a larger m. The "good" eigenvalue repeat in these calculation for different m, whereas the "spurious" eigenvalues do not repeat [12].

5.4 Simultaneous iteration versus Arnoldi versus Lanczos

So far, we have not seen a published comprehensive comparative study of the three methods. We have seen that many factors from implementation and application problems could affect such a comparison. In theory, the fundamental advantage of the Lanczos and Arnoldi methods is that they discard no information of the generated base vectors, whereas simultaneous iteration, at each step, overwrites a set of approximation eigenvectors (or Schur vectors) with a better set. Such advantage of the Lanczos and Arnoldi methods has been illustrated in many numerical experiments. However, a firm convergence theory has been established for simultaneous iteration [49], but there are still many qusetions for the Arnoldi and Lanczos algorithms [44]. The loss of (bi)-orthogonality among the computed basis (bases) of the Krylov subspaces in Arnoldi and Lanczos and its effects are not well-understood in contrast with the symmetric eigenproblem. Arnoldi method could also suffer from the unbounded growth in the core-storage. Although the Lanczos method only requires 6 or 8 vectors in the core-storage (for the Look-ahead Lanczos scheme, it is a little high), user must provide the vector $z = A^T x$, which may not be available in some applications. Without doubt, the three methods will still continue compete each other in the coming years. With the growing number of applications and improvement of software environment, the picture will be more clear.

6 What have we learned since 1982

Since the publication of Parlett's essay [34] (which was from a talk given at a Sparse matrix Symposium held in Fairfield Glade, Tennessee in 1982), over the past decades, we have still been seeking a better way for an old task. Nevertheless, with the contributions of many people, we have made a significant move of our expertise for the problem.

1. BLAS: The wide popularity of the (dense or sparse) BLAS for the basic vector and matrix operations, such as the inner product, is beneficial to the software development. The programs turn out to be of simpler structure and to gain better performance with optimized BLAS.

   SPARSKIT of Saad [42] is a tool to manipulate and perform simple operations with sparse matrices, such as exchange data structure, as well as basic linear algebra routines for sparse matrices. Its use in software development has been growing.

The most recent release of MATLAB includes sparse matrix storage and operation features [27].

2. The so-called algorithm independency: It means that an algorithm refers the matrix in question only through an external user supplied subroutines to form the matrix-vector multiplications Av and/or A^Tv (or $f(A)v$ and/or $f(A^T)v$, where $f(x)$ is a spectral transformation function). Although this has been known over decades, today it has been further exploited and is used in terms of *reverse communication* [26, 48]. One has recognized the importance of such treatment. It implies that one can develope a set of "universal" software that is applicable to any matrix. To apply the software to a particular matrix, one simply provides a subroutine to evaluate the required matrix-vector products. For some involved applications, the faster algorithms can be developed for such desired matrix-vector operation.

3. Spectrum transformation (Preconditioning): Such techniques have been reviewed in section 5.2.

4. Acceleration techniques: The idea of the acceleration techniques is to update the initial vector(s) in the restarted simultaneous iteration and Krylov subspaces methods, and to force the convergence of the residual vector(s) to zero faster. These techniques have been used in simultaneous iteration and Arnoldi method [44, 47].

5. Deflation technique: As a common phenomenon in all algorithms, some of the eigenvalues (and eigenvectors or Schur vectors) converge faster than the others. Therefore deflation procedures for those convergent eigenvalues (and eigenvectors or Schur vectors) can be quite effective in terms of execution time in practice. Although the technique has been used before 1982, for example, in [50], a complete study of the so called Wielandt deflation procedure is by Saad from 1985. It has become very common to include such deflation procedure in software.

6. Assessment of computed results: The nonsymmetric eigenvalue problem is harder than the symmetric counterpart, partially because of the possible extreme ill-conditioning of the problem, and higher sensitivity to the roundoff error. For nonsymmetric matrices, the norms of the residuals of approximate eigenpairs does not provide sufficient information to bound the error in the approximate eigenvalue. Parlett-Kahan-Jiang's theorem [29] developed in 1982 first gives a bound on the distance to the nearest matrix for which the given approximations are exact. This is the best we can expect. Such backward error analysis has served as the stopping criterion of the iteration methods. The numerical practices of this bound are reported by Cullum and Willoughby in 1986 [12].

7 Work in progress

The nonsymmetric eigenproblem, whether small or large, dense or sparse, is one of the hardest linear algebra problems to solve effectively and accurately. For the dense eigenvalue problem, the recent years research are mostly concentrated on the parallel algorithm development. The followings are some outline of such development in progress.

Increasing the efficiency of the QR algorithm on parallel computing environment is still a challenging problem. Dubrulle [17] reported the performance of the modified multishift

QR algorithm on IBM 3090 machines, Henry [25] studies the different strategies in the QR algorithm to improve the performance, such as the use of block methods, reducing the average stride and data movement with hybrid steps, and the use of block data structures. Van Dooren [53] proposed a prototype of block QR algorithm, but it is unclear the applicability of this idea. The convergence analysis of the general QR algorithm is carried out in [54].

The great success of the divide and conquer idea for the symmetric tridiagonal eigenproblem has led several researchers to extend it to the nonsymmetric Hessenberg eigenproblem. By setting one (or more) subdiagonal element(s) of Hessenberg matrix to zero to form independent Hessenberg eigenproblems which can be solved in parallel, and then merging the solutions of subproblems by rank-update and iterative refinement technique [16] or homotopy technique [30] to yield the solution of the original problem. Although some successful aspects have been seen, more investigation is needed to deal with the possible divergence of the iteration and identification problem, which means that we need to know if two iterations converge to the same root, is it multiple or have we missed a root? Theoretical aspects of such divide and conquer method have been discussed in recent papers by Adams and Arbenz [1], Jessup [28].

A new divide and conquer method to find the Schur decomposition by using tools such as matrix sign function or beta function is under investigation by several groups [31, 3, 5]. There is no identification problem for the methods and is built on the highly block matrix operations. But the use of the inverse of the original matrix and the possible significant increase in the number of the floating point operations are the potential barrier for the idea. It will be interesting to see what happens.

In the numerical analysis community, the sparse eigenproblem is still at the stage of developing more reliable, robust and efficient sequential algorithms, and analyzing the numerical behaviors of algorithms. Most recently, a generalization of the shift-and-invert Krylov subspace methods (Arnoldi or Lanczos) for the nonsymmetric matrix pencil eigenvalue problem has been investigated by Ruhe [38]. It uses several shifts at a time to build a rational Krylov subspace. The work is at its early stage.

A recent report by Chatelin and Godet-Thobie [10] is focused on the influence of the departure from normality of the matrix on the stability of the Arnoldi algorithm. The research being undertaken by Bai [4], Days [14] and Strakos [52] is aiming at the error analysis of the Krylov subspaces method in the presence of the floating point arithmetic and the effect of the loss of the (bi-)orthogonality in the practical computation.

We terminate this section with an interesting practical higher order λ-matrix eigenvalue problems of the form

$$(\lambda^5 A_0 + \lambda^4 A_1 + \lambda^3 A_2 + \lambda^2 A_3 + \lambda A_4 + A_5)x = 0.$$

The problem is from structural mechanics combined with aerodynamics [10]. Beside to cast such eigenvalue problem to the standard eigenvalues problem, no much progress has been seen how to solve such λ-matrix eigenvalue problem directly and efficiently.

8 Conclusions

The nonsymmetric eigenvalue problem is an inherently much harder problem than the symmetric eigenvalue problem. For small problems, the QR algorithm essentially occupies the picture, although the best way to solve it on parallel computational environment is still challenging us. For large problems, it is still an open research topic despite all the progress made. While numerical analysts are seeking better ways to implement existing algorithms or new algorithms, engineers will go ahead and try to solve their problems with or without having the help from numerical analysts. Certainly, when the numerical analysts provide better tools for solving the problem, the number of applications will grow.

Finally, in the event of honoring Professors Kahan and Parlett's 60th birthdays, let us quote a conversation between Professor Kahan and a group of computer science students:

> Question: Working on the problem you have been honored for, you must have had time when you thought about throwing it all in. So, what motivated you to continue to reach your aim?
>
> Kahan: The thought of "throwing it all in" never occurred to me; I was motivated by curiosity, each obstacle was just another problem to analyze, understand, and then overcome.

Acknowledgement

The author thanks many colleagues and friends for providing invaluable helpful information on the nonsymmetric eigenvalue problem and sharing their programs which the author could not possibly possess. In particular, the author is grateful for the unique opportunity of visiting IMA at University of Minnesota during its 1991–1992 Applied Linear Algebra year, and communication on this subject with numerical analysis experts Françoise Chatelin, Jane Cullum, Jim Demmel, Iain Duff, Roland Freund, Gene Golub, Anne Greenbaum, Nick Higham, Noel Nachtigal, Axel Ruhe, Youcef Saad, Danny Sorensen, Pete Stewart and Zdenek Strakos. The author would also like to express thanks to Nick Higham and the referees, whose comments lead to improvements in the presentation.

References

[1] L. Adams and P. Arbenz. Towards a divide and conquer algorithm for the real nonsymmetric eigenvalue problem. Dept. of App. Math., Univ. of Washington, 1991.

[2] 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, Release 1.0.* SIAM, Philadelphia, PA, 1992.

[3] L. Auslander and A. Tsao. On parallelizable eigensolvers. *Advances in Applied Mathematices.* to appear, 1992.

[4] Z. Bai. Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem. IMA Preprint 935, Univ. of Minnesota, April 1992. to appear in *Math. Comp.*

[5] Z. Bai and J. Demmel. Design of a parallel eigenroutine toolbox (Part I). Dept. of Math. RR 92-09, Univ. of Kentucky, Dec. 1992. Appears also as Univ. of California at Berkeley, Comp. Sci. Div. Report UCB/CSD-92-718.

[6] Z. Bai and G. W. Stewart. SRRIT: A Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix. Dept. of Comp. Sci. TR 2908, Univ. of Maryland, April 1992.

[7] N. E. Bixler. *Stability of coating flow.* PhD thesis, Univ. of Minnesota, Minneapolis, MN, 1982.

[8] A. G. Boudouvis, J. L. Puchalla, L. E. Scriven, and R. E. Rosensweig. Normal field instability and patterns in pools of ferrofluid. *J. of Magnetism and Magnetic Materials*, 65, 1987.

[9] A. G. Boudouvis and L. E. Scriven. A three-dimensional pattern selection scenario. Technical Report 91/248, Univ. of Minnesota Supercomput. Inst., 1991.

[10] F. Chatelin and S. Godet-Thobie. Stability analysis and nonnormal matrices in aeronautical industries. In R. Vichnevetsky and J. Miller, editors, *Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics*, 1991.

[11] K. L. Christodoulou and L. E. Scriven. Finding leading modes of a viscous free surface flow: an asymmetric generalized eigenproblem. *J. of Scient. Comput.*, 3:355–406, 1988.

[12] J. Cullum and R. A. Willoughby. A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices. In J. Cullum and R. A. Willoughby, editors, *Large Scale Eigenvalue Problems.* North-Holland, Amsterdam, 1986.

[13] E. R. Davidson. Super-matrix methods. *Comput. Phy. Comm.*, 53:49–60, 1989.

[14] D. Day. Semibiorthogonality in the Lanczos algorithm, manuscript, 1992.

[15] J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart. *LINPACK User's Guide.* SIAM, Philadelphia, PA, 1979.

[16] J. Dongarra and M. Sidani. A parallel algorithm for the non-symmetric eigenvalue problem. Comp. Sci. Dept. Tech. Rep. CS-91-137, Univ. of Tennessee, Knoxville, 1991.

[17] A. Dubrulle. The multishift QR algorithm: is it worth the trouble? Palo Alto Scientific Center Report G320-3558x, IBM Corp., 1991.

[18] I.S. Duff and J. A. Scott. Computing selected eigenvalues of sparse unsymmetric matrices using subspace iteration. Technical Report RAL-91-056, Rutherford Appleton Laboratory, 1991.

[19] L. E. Eriksson and A. Rizzi. Computer aidied analysis of the convergence to steady state of discrete approximations to the Euler equations. *J. of Comput. Phy.*, 57:90–128, 1985.

[20] R. Freund and N. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part II. Technical Report TR 90-46, RIACS, Nov. 1990.

[21] A. Galick, T. Kerkhoven, and U. Ravaioli. Iterative solution of the eigenvalue problem for a dielectric wageguide. CSRD Report 1119, Univ. of Illinois at Urbana-Champaign, July 1991.

[22] T. J. Garratt, G. Moore, and A. Spence. Two methods for the numerical detection of hopf bifurcations. In R. Seydel et al, editor, *Bifurcation and Chaos: analysis, algorithms, applications.* Birkhauser, 1991.

[23] I. Goldhirsch, S. A. Orszag, and B. K Maulik. An efficient method for computing leading eigenvalues and eigencectors of large asymmetric matrix. *J. of Scient. Comput.*, 2:33–58, 1987.

[24] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

[25] G. Henry. Increasing data reuse in the unsymmetric QR algorithm. Technical Report CTC-TR-100, Cornell Theory Center, Cornell University, 1992.

[26] N. J. Higham. Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation. *ACM Trans. Math. Soft.*, 14:381–396, 1988.

[27] C. Moler J. Gilbert and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM J. Matrix Anal. Appl.*, 13:333–356, 1992.

[28] E. Jessup. A case against a divide and conquer approach to the nonsymmetric eigenproblem. Technical Report ONRL/TM-11903, Oak Ridge National Laboratory, 1991.

[29] W. Kahan, B. Parlett, and K. Jiang. Residual bounds on approximate eigenvalues of a nonnormal matrices. *SIAM J. Numer. Anal.*, 19:470–484, 1982.

[30] T.-Y. Li and Z Zeng. Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 59(200):483–502, 1992.

[31] C-C. Lin and E. Zmijewski. A parallel algorithm for computing the eigenvalues of an unsymmetric matrix on an SIMD mesh of processors. Dept. of Comp. Sci. TRCS 91-15, Univ. of California, Santa Barbara, July 1991.

[32] A. Mahajan, E. H. Dowell, and D. Bliss. Eigenvalue calculation procedure for an Euler/Navier-Stokes solver with applications to flows over airfoils. *J. of Comput. Phy.*, 97:398–413, 1991.

[33] N. Martins. Efficient eigenvalue and frequency response methods applied to power system small-signal stability studies. *IEEE Trans. on Power Systems*, PWRS-1:217–226, 1986.

[34] B. Parlett. The software scene in the extraction of eigenvalues from sparse matrices. *SIAM J. Sci. Stat. Comput.*, 5:590–604, 1984.

[35] B. Parlett and W. Heng. The method of minimal representation in 2D Ising model calculations. PAM-549, Univ. of California, Berkeley, 1992.

[36] B. Parlett and Y. Saad. Complex shift and invert strategies for real matrices. *Lin. Alg. and Appl.*, 88/89:575–595, 1987.

[37] A. Ruhe. Rational Krylov sequence methods for eigenvalue computations. *Lin. Alg. Appl.*, 58:391–405, 1984.

[38] A. Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. in preparation, 1992.

[39] Y. Saad. Least squares polynomials in the complex plane with applications to solving sparse nonsymmetric matrix problems. Research Report YALEU/DCS/RR-276, Yale University, June 1983.

[40] Y. Saad. Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems. *SIAM J. Num. Anal.*, 24:155–169, 1987.

[41] Y. Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Comput. Phys. Comm.*, 53:71–90, 1989.

[42] Y. Saad. SPARSKIT: A basic tool kit for sparse matrix computation. CSRD Report 1029, Univ. of Illinois at Urbana-Champaign, August 1990.

[43] Y. Saad. Analysis of some Krylov subspace apporximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 1992.

[44] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Manchester and Halsted Press, New York, 1992.

[45] S. Seki, T. Yamanaka, and K. Yokoyama. Two-dimensional analysis of optical waveguides with a nonuniform finite difference method. *IEE Proceedings-J*, 138:123–127, 1991.

[46] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide.* Springer-Verlag, Berlin, 1976.

[47] D. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Mat. Anal. Appl.*, 13(1):357–385, 1992.

[48] D. Sorensen and P. Vu. private communication, 1992.

[49] G. W. Stewart. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrix. *Numer. Math.*, 25:123–126, 1976.

[50] G. W. Stewart. SRRIT: A Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix. Dept. of Comp. Sci. TR 154, Univ. of Maryland, 1978.

[51] W. J. Stewart and A. Jennings. Algorithm 570 LOPSI: A simultaeous iteration algorithm for real matrices. *ACM Trans. on Math. Soft.*, 7:230–232, 1981.

[52] Z. Strakos. private communication, 1992.

[53] P. Van Dooren. Block shifts and block QR, Sep. 1991. The presentation at SIAM Conf. on Appl. Lin. Alg., Minneapolis, MN.

[54] D. Watkins and L. Elsner. Convergence of algorithms of decomposition type for the eigenvalue problem. *Lin. Alg. Appl.*, 143:19–47, 1991.