Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Journal of Sound and Vibration 272 (2004) 731–748

JOURNAL OF SOUND AND VIBRATION

www.elsevier.com/locate/jsvi

Disk brake squeal prediction using the ABLE algorithm

G. Lou[a], T.W. Wu[a,*], Z. Bai[b]

[a] *Department of Mechanical Engineering, University of Kentucky, Lexington, KY 40506, USA*
[b] *Department of Computer Science, University of California, Davis, CA 95616, USA*

Received 5 June 2002; accepted 25 March 2003

Abstract

Disk brake squeal noise is mainly due to unstable friction-induced vibration. A typical disk brake system includes two pads, a rotor, a caliper and a piston. In order to predict if a disk brake system will generate squeal, the finite element method (FEM) is used to simulate the system. At the contact interfaces between the pads and the rotor, the normal displacement is continuous and Coulomb's friction law is applied. Thus, the resulting FEM matrices of the dynamic system become unsymmetric, which will yield complex eigenvalues. Any complex eigenvalue with a positive real part indicates an unstable mode, which may result in squeal. In real-world applications, the FEM model of a disk brake system usually contains tens of thousands of degrees of freedom (d.o.f.s). Therefore any direct eigenvalue solver based on the dense matrix data structure cannot efficiently perform the analysis, mainly due to its huge memory requirement and long computation time. It is well known that the FEM matrices are generally sparse and hence only the non-zeros of the matrices need to be stored for eigenvalue analysis. A recently developed iterative method named ABLE is used in this paper to search for any unstable modes within a certain user-specified frequency range. The complex eigenvalue solver ABLE is based on an adaptive block Lanczos method for sparse unsymmetric matrices. Numerical examples are presented to demonstrate the formulation and the eigenvalues are compared to the results from the component modal synthesis (CMS).
© 2003 Elsevier Ltd. All rights reserved.

1. Introduction

It is an important technology issue to reduce squeal during braking in order to improve the quiet and comfortable performance of vehicles; otherwise customers would lose confidence on the quality of their vehicles if the brake squeal occurs too often. Squeal usually manifests at higher frequencies (500 Hz or higher) [1]. When brake squeal occurs, the rubbing surfaces do not stick to

*Corresponding author. Tel.: +1-859-257-6336 Ext. 80644; fax: +1-859-257-3304.
E-mail address: timwu@engr.uky.edu (T.W. Wu).

0022-460X/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0022-460X(03)00416-4

each other and the relative sliding speed is always unidirectional. Brake squeal can be defined as a self-excited vibration caused by fluctuations in the friction forces at the pad-rotor interfaces. Fluctuations caused by friction coefficient–velocity characteristics and fluctuations due to normal forces between pads and rotor (even if the friction coefficient is constant) are the two theories to explain the cause of the friction force fluctuation [2].

The FEM has already been used to predict the eigenvalues of simplified disk brake systems [1,3–6]. For a 3-D FEM model, each node has three d.o.f.s. For any node at the pad–rotor interfaces, Coulomb's friction law is used to relate the d.o.f.s in the normal direction to the d.o.f.s in the two tangential directions, causing the resulting FEM matrices to become unsymmetric. By solving the complex eigenvalues of the newly formed matrices, one can predict if any unstable mode is present. Any complex eigenvalues with a positive real part may indicate an unstable mode. The real part of a complex eigenvalue is called "propensity", and the larger its magnitude is, the more likely the mode is an unstable mode.

Due to the high frequency nature, a real disk brake system usually has tens of thousands of d.o.f.s. The immediate difficulty with the FEM becomes the huge memory requirement and long solution time. It is well known that for any direct eigenvalue solvers such as the QZ method, the memory requirement is on the order of n^2 and the total CPU time is on the order of n^3, where n is the number of d.o.f.s of the system. Thus it is impractical to use any direct complex eigenvalue solver for brake squeal analysis.

The difficulty could be alleviated by using component mode synthesis (CMS). The CMS is an approximation method that significantly reduces the total d.o.f.s of a large system. The basic idea of the CMS is to divide the whole brake system into component structures [4,6]. First, the eigenvalues and the eigenvectors are found for each individual component (such as caliper, pads, disk rotor, etc.) by the FEM. The parts are then joined together by spring elements to construct a brake assembly. Special friction springs are applied between the nodes at the pad–rotor interfaces to simulate the frictional mechanism. Note that in order to solve the system, the physical co-ordinate model is actually transformed into a modal co-ordinate model. Finally a reduced-size eigenvalue problem is solved to identify if there are any unstable modes.

There are some limitations associated with the CMS. First, it is a kind of approximation method. One would not use all of the modes of each component to simulate the problem; otherwise it is meaningless to do the CMS. Second, since one can only use a small number of modes, then the problem becomes selecting how many and what modes of each component. The third limitation is that it is difficult to do an analysis for a quadratic eigenvalue problem (QEP) when the damping matrix is present. The reason is that in order to form the QEP, the normal contact forces at the pad–rotor interfaces need to be solved first so that the damping matrix can be formed. However, it is difficult to solve the contact forces in the modal space. Another limitation is the usage of massless springs to represent the frictional mechanism, which might lead to great error when modelling high-frequency squeal [1].

Because of these limitations of CMS, it may be desirable to solve the disk brake system in the physical displacement space. However as mentioned earlier, it is impractical to solve the complex eigenvalue problem using any direct solvers like the QZ method. Iterative methods could be a better choice when only the eigenvalues within a certain frequency range are to be sought. The Arnoldi and the Lanczos methods are the two most widely used iterative methods. However, a main drawback of the Arnoldi method is that at each step all the Arnoldi vectors are transferred from memory to the CPU, known as "long recurrence" [9]. On the other hand, in the Lanczos method, only current vectors are recalled at each step, or "short recurrence."

The Lanczos method originated in the 1950s. In the 1970s and 1980s most studies of the Lanczos method were on symmetric eigenvalue problems. In 1986, Cullum and Willoughby [7] first proposed an implementation of the non-symmetrical Lanczos method without reorthogonalization. Then this method was extended to the generalized eigenvalue problem $Ax = \lambda Bx$ in 1989 [8]. It is noted that sometimes the Lanczos method may fail with "breakdown." Since the 1980s, a lot of work has been done on the issue of how to fix the breakdown problem. Ye proposed a scheme to cure the breakdown by choosing a new starting vector. This vector generates another Krylov subspace, which is appropriately combined with the old one [10]. This idea has been incorporated in the recently proposed ABLE algorithm by Bai and Day [9]. ABLE is an adaptive block Lanczos method for non-Hermitian eigenvalue problems [11]. In this algorithm, an adaptive block size scheme is used to cure breakdowns and adapt to the order of multiple or clustered eigenvalues. ABLE also monitors the loss of biorthogonality and maintains semi-biorthogonality among the computed Lanczos vectors, which generalizes the standard technique used in the implementation in the symmetric Lanczos method. Another virtue of ABLE is that it can take great advantage of the sparsity of the FEM matrices. Since the FEM matrices are generally very sparse, the employed matrix-vector operation can be done very quickly.

In this paper, a general formulation of friction induced vibration to predict the complex eigenvalues for a disk brake system is presented. A brief introduction about the complex eigenvalue analysis is followed. Then the recently developed iterative method ABLE is described. Special implementation issues about ABLE in the application to disk brake systems are also discussed. Numerical examples on models with different number of d.o.f.s are presented. Good agreement is achieved in comparison with the CMS.

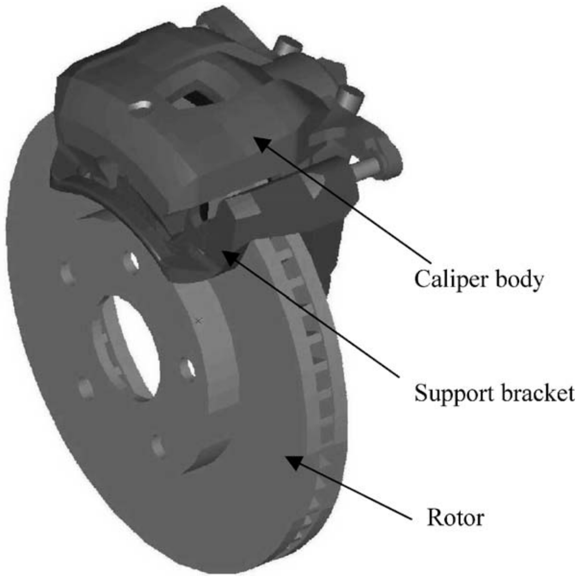

Fig. 1. Disk brake system assemblage.

2. Formulation of friction induced vibration for disk brake systems

A typical disk brake system assembly is shown in Fig. 1. In order to simplify the description of the general formulation, a simplified model shown in Fig. 2 is used. It should be noted that although the following derivation is based on this simplified model, the formulation is actually valid for more complex models.

At the beginning, the equilibrium analysis for steady sliding is first presented, where a certain instant is considered and the acceleration of the rotor is neglected. After that, the dynamic perturbation is applied to the equilibrium state to obtain the dynamic system equations.

2.1. Equilibrium analysis for steady sliding

With reference to Fig. 2, the nodes on the rotor at the contact surface are referred to as ''Master'' nodes (all variables with superscript M), and the corresponding nodes on the pads are

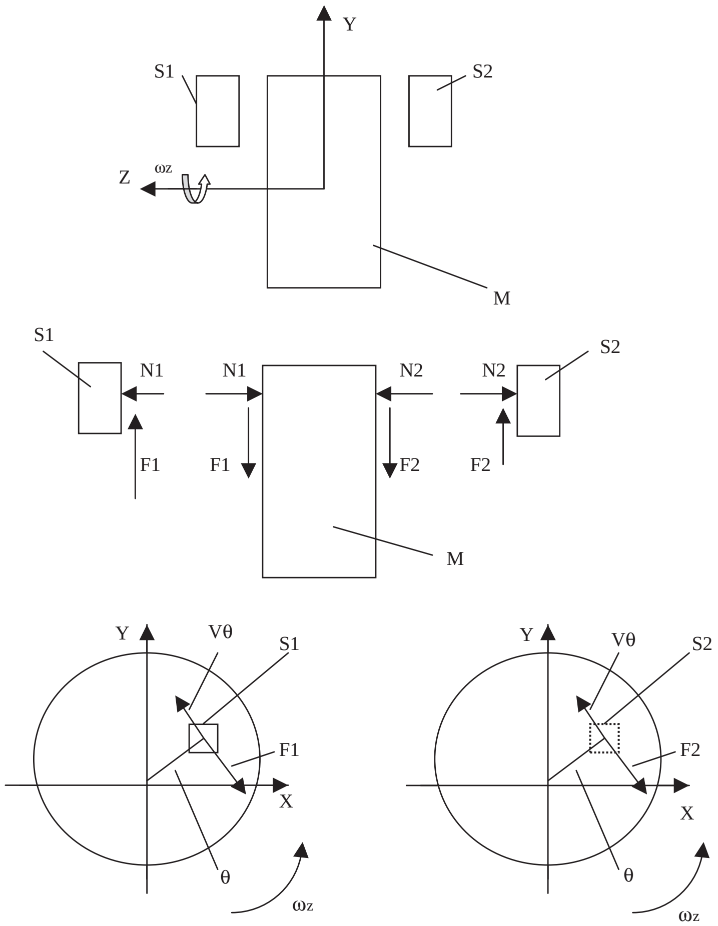

Fig. 2. A simplified disk brake system: M, master rotor; S1, S2, Slave 1 and 2 (pad); X, Y, tangent plane; Z, normal direction; F, friction force; N, normal force.

referred to as the "Slave" nodes (all variables with superscript S). The interface in the positive Z direction is called interface 1 and the interface in negative Z direction is called interface 2. Assume that the rotor rotates about the positive Z-axis as shown in Fig. 2.

The friction forces at the two interfaces are evaluated by Coulomb's friction law. In this study, the kinetic friction coefficient is assumed to have the form

$$\mu = \mu_0 + \alpha V_\theta, \tag{1}$$

where both μ_0 and α are constants, and V_θ is the relative sliding speed in the tangential direction. The value of α usually is negative, which means the $\mu - V_\theta$ slope is negative [1].

In Cartesian co-ordinates, the components of the friction force are

$$F_{1x} = F_1 \sin\theta = \mu N_1 \sin\theta = \mu_{1x} N_1, \tag{2}$$

$$F_{1y} = -F_1 \cos\theta = -\mu N_1 \cos\theta = -\mu_{1y} N_1, \tag{3}$$

$$F_{2x} = F_2 \sin\theta = \mu N_2 \sin\theta = \mu_{2x} N_2, \tag{4}$$

$$F_{2y} = -F_2 \cos\theta = -\mu N_2 \cos\theta = -\mu_{2y} N_2, \tag{5}$$

where N is the normal contact force, and $\mu_{1x}=\mu\sin\theta, \mu_{1y} = \mu\cos\theta, \mu_{2x}=\mu\sin\theta, \mu_{2y}=\mu\cos\theta$.

The FEM matrix equation for the whole system can be written as

$$\begin{bmatrix} K_{1,1}K_{1,2}\cdots K_{1,13} \\ K_{2,1}K_{2,2}\cdots K_{2,13} \\ K_{3,1}K_{3,2}\cdots K_{3,13} \\ K_{4,1}K_{4,2}\cdots K_{4,13} \\ K_{5,1}K_{5,2}\cdots K_{5,13} \\ K_{6,1}K_{6,2}\cdots K_{6,13} \\ K_{7,1}K_{7,2}\cdots K_{7,13} \\ K_{8,1}K_{8,2}\cdots K_{8,13} \\ K_{9,1}K_{9,2}\cdots K_{9,13} \\ K_{10,1}K_{10,2}\cdots K_{10,13} \\ K_{11,1}K_{11,2}\cdots K_{11,13} \\ K_{12,1}K_{12,2}\cdots K_{12,13} \\ K_{13,1}K_{13,2}\cdots K_{13,13} \end{bmatrix} \begin{bmatrix} u^S_{1X} \\ u^S_{1Y} \\ u^S_{1Z} \\ u^M_{1X} \\ u^M_{1Y} \\ u^M_{1Z} \\ u^S_{2X} \\ u^S_{2Y} \\ u^S_{2Z} \\ u^M_{2X} \\ u^M_{2Y} \\ u^M_{2Z} \\ u_{\text{other}} \end{bmatrix} = \begin{bmatrix} -F_{1X} \\ -F_{1Y} \\ N_1 \\ F_{1X} \\ F_{1Y} \\ -N_1 \\ -F_{2X} \\ -F_{2Y} \\ -N_2 \\ F_{2X} \\ F_{2Y} \\ N_2 \\ P \end{bmatrix}, \tag{6}$$

where K is the stiffness matrix, and u is the nodal displacement vector. The subscript "other" denotes all other nodes not on the two interfaces. The external forces acting on "other" nodes are denoted by P. Eliminating the friction forces in Eq. (6), we get

$$\begin{bmatrix} (K_{1,1}+\mu_{1x}K_{3,1})\cdots(K_{1,13}+\mu_{1x}K_{3,13}) \\ (K_{2,1}-\mu_{1y}K_{3,1})\cdots(K_{2,13}-\mu_{1y}K_{3,13}) \\ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ -1\ \ 0\cdots 0 \\ (K_{4,1}+\mu_{1x}K_{6,1})\cdots(K_{4,13}+\mu_{1x}K_{6,13}) \\ (K_{5,1}-\mu_{1y}K_{6,1})\cdots(K_{5,13}-\mu_{1y}K_{6,13}) \\ (K_{6,1}+K_{3,1})\cdots(K_{6,13}+K_{3,13}) \\ (K_{7,1}-\mu_{2x}K_{9,1})\cdots(K_{7,13}-\mu_{2x}K_{9,13}) \\ (K_{8,1}+\mu_{2y}K_{9,1})\cdots(K_{8,13}+\mu_{2y}K_{9,13}) \\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ -1\ \ 0 \\ (K_{10,1}-\mu_{2x}K_{12,1})\cdots(K_{10,13}-\mu_{2x}K_{12,13}) \\ (K_{11,1}+\mu_{2y}K_{12,1})\cdots(K_{11,13}+\mu_{2y}K_{12,13}) \\ (K_{12,1}+K_{9,1})\cdots(K_{12,13}+K_{9,13}) \\ K_{13,1}K_{13,2}\cdots K_{13,13} \end{bmatrix} \begin{bmatrix} u^S_{1X} \\ u^S_{1Y} \\ u^S_{1Z} \\ u^M_{1X} \\ u^M_{1Y} \\ u^M_{1Z} \\ u^S_{2X} \\ u^S_{2Y} \\ u^S_{2Z} \\ u^M_{2X} \\ u^M_{2Y} \\ u^M_{2Z} \\ u_{\text{other}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ P \end{bmatrix}. \tag{7}$$

This is the equilibrium equation at steady sliding when the acceleration is ignored. Solve Eq. (7) for the displacements, and then the normal contact forces N_1 and N_2 can be obtained by Eq. (6). It should be noted that although the third and ninth rows of the matrix equation are shown to enforce the continuity of displacement in the Z direction at the contact surfaces, the slave degrees of freedom in Z direction are actually condensed out in real numerical implementation.

2.2. Dynamic perturbation

Dynamic perturbation is performed when a small deviation from the equilibrium position is added to the system. By this way we can see whether the system is stable or not. By applying dynamic perturbation [1] and ensuring the continuity of the acceleration, velocity and displacement in the normal direction of the corresponding master and slave nodes, the following matrix equation can be formulated:

$$\begin{bmatrix} (M_{1,1}+\mu_{1xst}M_{3,1})\cdots(M_{1,13}+\mu_{1xst}M_{3,13}) \\ (M_{2,1}-\mu_{1yst}M_{3,1})\cdots(M_{2,13}-\mu_{1yst}M_{3,13}) \\ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ -1\ \ 0\cdots 0 \\ (M_{4,1}+\mu_{1xst}M_{6,1})\cdots(M_{4,13}+\mu_{1xst}M_{6,13}) \\ (M_{5,1}-\mu_{1yst}K_{6,1})\cdots(K_{5,13}-\mu_{1yst}K_{6,13}) \\ (M_{6,1}+M_{3,1})\cdots(M_{6,13}+M_{3,13}) \\ (M_{7,1}-\mu_{2xst}M_{9,1})\cdots(M_{7,13}-\mu_{2xst}M_{9,13}) \\ (M_{8,1}+\mu_{2yst}M_{9,1})\cdots(M_{8,13}+\mu_{2yst}M_{9,13}) \\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ -1\ \ 0 \\ (M_{10,1}-\mu_{2xst}M_{12,1})\cdots(M_{10,13}-\mu_{2xst}M_{12,13}) \\ (M_{11,1}+\mu_{2yst}M_{12,1})\cdots(M_{11,13}+\mu_{2yst}M_{12,13}) \\ (M_{12,1}+M_{9,1})\cdots(M_{12,13}+M_{9,13}) \\ M_{13,1}M_{13,2}\cdots M_{13,13} \end{bmatrix} \begin{bmatrix} \Delta\ddot{u}^S_{1X} \\ \Delta\ddot{u}^S_{1Y} \\ \Delta\ddot{u}^S_{1Z} \\ \Delta\ddot{u}^M_{1X} \\ \Delta\ddot{u}^M_{1Y} \\ \Delta\ddot{u}^M_{1Z} \\ \Delta\ddot{u}^S_{2X} \\ \Delta\ddot{u}^S_{2Y} \\ \Delta\ddot{u}^S_{2Z} \\ \Delta\ddot{u}^M_{2X} \\ \Delta\ddot{u}^M_{2Y} \\ \Delta\ddot{u}^M_{2Z} \\ \Delta\ddot{u}_{\text{other}} \end{bmatrix}$$

$$+\begin{bmatrix} -\alpha N_{St1}\;\;0\;\;0\;\;\alpha N_{St1}\;\;0\cdots0 \\ 0\;\;\alpha N_{St1}\;\;0\;\;0\;\;-\alpha N_{St1}\;\;0\cdots0 \\ 0\;\;0\;\;1\;\;0\;\;0\;\;-1\;\;0\cdots0 \\ \alpha N_{St1}\;\;0\;\;0\;\;-\alpha N_{St1}\;\;0\cdots0 \\ 0\;\;-\alpha N_{St1}\;\;0\;\;0\;\;\alpha N_{St1}\;\;0\cdots0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\cdots0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0-\alpha N_{St2}\;\;0\;\;0\;\;\alpha N_{St2}0\;\;0\;\;0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;\alpha N_{St2}\;\;0\;\;0\;\;-\alpha N_{St2}\;\;0\;\;0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;1\;\;0\;\;0\;\;-1\;\;0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;\alpha N_{St2}\;\;0\;\;0\;\;-\alpha N_{St2}\;\;0\;\;0\;\;0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;-\alpha N_{St2}0\;\;0\;\;\alpha N_{St2}\;\;0\;\;0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0 \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0 \end{bmatrix} \begin{bmatrix} \Delta\dot{u}^S_{1X} \\ \Delta\dot{u}^S_{1Y} \\ \Delta\dot{u}^S_{1Z} \\ \Delta\dot{u}^M_{1X} \\ \Delta\dot{u}^M_{1Y} \\ \Delta\dot{u}^M_{1Z} \\ \Delta\dot{u}^S_{2X} \\ \Delta\dot{u}^S_{2Y} \\ \Delta\dot{u}^S_{2Z} \\ \Delta\dot{u}^M_{2X} \\ \Delta\dot{u}^M_{2Y} \\ \Delta\dot{u}^M_{2Z} \\ \Delta\dot{u}_{\text{other}} \end{bmatrix}$$

$$+\begin{bmatrix} (K_{1,1}+\mu_{1xst}K_{3,1})\cdots(K_{1,13}+\mu_{1xst}K_{3,13}) \\ (K_{2,1}-\mu_{1yst}K_{3,1})\cdots(K_{2,13}-\mu_{1yst}K_{3,13}) \\ 0\;\;0\;\;1\;\;0\;\;0\;\;0-1\;\;0\cdots0 \\ (K_{4,1}+\mu_{1xst}K_{6,1})\cdots(K_{4,13}+\mu_{1xst}K_{6,13}) \\ (K_{5,1}-\mu_{1yst}K_{6,1})\cdots(K_{5,13}-\mu_{1yst}K_{6,13}) \\ (K_{6,1}+K_{3,1})\cdots(K_{6,13}+K_{3,13}) \\ (K_{7,1}-\mu_{2xst}K_{9,1})\cdots(K_{7,13}-\mu_{2xst}K_{9,13}) \\ (K_{8,1}+\mu_{2yst}K_{9,1})\cdots(K_{8,13}+\mu_{2yst}K_{9,13}) \\ 0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;0\;\;1\;\;0\;\;0\;\;-1\;\;0 \\ (K_{10,1}-\mu_{2xst}K_{12,1})\cdots(K_{10,13}-\mu_{2xst}K_{12,13}) \\ (K_{11,1}+\mu_{2yst}K_{12,1})\cdots(K_{11,13}+\mu_{2yst}K_{12,13}) \\ (K_{12,1}+K_{9,1})\cdots(K_{12,13}+K_{9,13}) \\ K_{13,1}K_{13,2}\cdots K_{13,13} \end{bmatrix} \begin{bmatrix} \Delta u^S_{1X} \\ \Delta u^S_{1Y} \\ \Delta u^S_{1Z} \\ \Delta u^M_{1X} \\ \Delta u^M_{1Y} \\ \Delta u^M_{1Z} \\ \Delta u^S_{2X} \\ \Delta u^S_{2Y} \\ \Delta u^S_{2Z} \\ \Delta u^M_{2X} \\ \Delta u^M_{2Y} \\ \Delta u^M_{2Z} \\ \Delta u_{\text{other}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \Delta P \end{bmatrix}. \tag{8}$$

We rewrite the above matrix equation in the following compact form:

$$[M]\{\Delta\ddot{u}\} + [C]\{\Delta\dot{u}\} + [K]\{\Delta u\} = \{\Delta f\}, \tag{9}$$

where $[M]$, $[C]$ and $[K]$ are the resulting mass, damping and stiffness matrices, respectively, with the friction mechanism applied. Note that all of the matrices are unsymmetric. Like in the static analysis, the slave d.o.f.s in Z direction at the contact surfaces are actually condensed out in real numerical implementation. Finally the complex eigenvalue analysis on Eq. (9) can be performed. It is noted that when $\alpha = 0$, the damping matrix vanishes and the static analysis is unnecessary.

The purpose of the static analysis is to find the normal contact force that only shows up in the damping matrix.

3. Complex eigenvalue analysis

Assume that the homogeneous solution for Eq. (9) has the form

$$u(t) = \mathrm{e}^{\lambda t} w.$$

Substitute the above equation into Eq. (9) to obtain the quadratic eigenvalue problem (QEP)

$$(\lambda^2 M + \lambda C + K)w = 0, \tag{10}$$

where λ and w are the eigenvalue and the eigenvector of the system, respectively. Both are complex in general.

The linearization method is a widely used to solve the QEP in Eq. (10). There are many different linearization forms. In this study, the following linearization form for Eq. (10) is used:

$$\begin{bmatrix} K & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} w \\ \lambda w \end{bmatrix} = \lambda \begin{bmatrix} -C & -M \\ I & 0 \end{bmatrix} \begin{bmatrix} w \\ \lambda w \end{bmatrix}, \tag{11}$$

where I is the identity matrix. Eq. (11) is in the form of the generalized eigenvalue problem

$$Ax = \lambda Bx. \tag{12}$$

Mathematically, Eqs. (10) and (11) have the same eigenvalues. Note that when $\alpha = 0$, the damping matrix $[C]$ is a zero matrix. Under this circumstance, the QEP in Eq. (10) reduces to

$$(\lambda^2 M + K)w = 0, \tag{13}$$

which is also in the form of Eq. (12) except that the eigenvalues can be solved without doubling the size of the problem. However since $[K]$ and $[M]$ are unsymmetric, complex eigenvalue analysis is still required when $\alpha = 0$.

4. ABLE: an adaptive block Lanczos method

The typical number of nodes in an FEM model for a disk brake system could well exceed 30,000. If $\alpha \neq 0$ and we linearize the QEP into a standard generalized eigenvalue problem, the order of the eigenvalue problem will be doubled as shown in Eq. (11). Thus it is impossible to solve the eigenvalue problem using any direct solvers such as the QZ method.

One characteristic of the FEM is that the matrices are generally very sparse. When the size of the FEM model increases, the matrices become even sparser. The iterative method ABLE can fully take advantage of the sparse matrix data structure.

4.1. ABLE algorithm template

The ABLE algorithm is based on the basic block Lanczos procedure (Appendices A.1 and A.2). The pseudo-code of ABLE is presented in Appendix A.3. The detailed explanation of the

Table 1
Functions

Logical variable	Functions
Fulldual	Use the two-sided Gram–Schmidt process to maintain full biorthogonality at each Lanczos step
Semidual	Monitor the loss of duality and correct the loss of biorthogonality at certain step
Treatbd	Cure the breakdown by increasing the block size
Group	Adapt the block size to the order of the cluster of the convergent Rayleigh–Ritz values

algorithm such as convergence criteria, TSMGS (two-sided modified Gram–Schmidt process) and CURE breakdown procedure, etc. can be found in Ref. [9]. Here we want to describe the four levels of this algorithm, which let users have great freedom to choose different levels for their special applications.

There are four logical variables, namely *fulldual*, *semidual*, *treatbd* and *group* in the ABLE algorithm. The functions for these variables are described in the following Table 1.

With different value combinations of these four variables, there are four levels of the algorithm.

Level 1: All logical flags are false

Implement the basic block Lanczos algorithm (Appendix A.1). Essentially it runs the 3-term short recurrences and does not save the computed Lanczos vectors. Only the eigenvalues are computed.

Level 2. fulldual=true and other logical flags are false

Maintain full biorthogonality. At each step the current Lanczos vectors are explicitly re-biorthogonalized against all previous Lanczos vectors by two-sided modified Gram–Schmidt process. Eigentriplets are computed in Levels$\geqslant$2.

Level 3. semidual=true and other logical flags are false

Maintain semi-biorthogonality. Less expensive in both floating point operations and memory references than full biorthogonality.

Level 4. (fulldual=true or semidual=true) and (treatbd=true and/or group=true)

Cure breakdown by increasing the block size and/or adapt the block size to be at least the order of any known cluster of eigenvalues.

Users may choose to implement the ABLE method to maintain full biorthogonality (Level 2) or semi-biorthogonality (Level 3) but not both. Level 4 is implemented on the top of the Level 2 or 3.

For the disk brake system, the eigenvalue distribution of a disk brake system is unknown. It is also unknown whether there are any clustered eigenvalues for a disk brake system. Furthermore, the eigenvectors corresponding to the squeal frequencies are required. Thus Level 4 of ABLE with *fulldual=true* is chosen to solve the eigenvalues in a certain frequency range. In this way, possible breakdown and failure to detect clustered eigenvalues can be avoided.

5. Some implementation issues for disk brake analysis

For the disk brake system application, there are several implementation issues that need to be taken into account before ABLE is applied to solve the eigenvalues.

First of all, the matrix size finally generated (after applying the friction mechanism) is huge (a typical matrix size is 100,000-by-100,000) but very sparse. Thus only the non-zero entries, i.e., the row numbers, column numbers and the values of the entries are stored. In this way, the storage requirement for the matrices of the typical problem is less than one hundred megabytes (200 Mbytes).

Secondly the magnitudes of the mass and the stiffness matrices are quite different. Typically the stiffness matrix magnitude is 10^8 times larger than that of the mass matrix. That could result in numerical loss of biorthogonality especially when the matrix size is large. A treatment we apply to the matrices is the simple scaling, i.e., using the infinity-norm of the individual matrix to scale the norms of the matrices to the same magnitude. In this way, ABLE will never numerically fail in our application.

Another issue we need to take into account is the frequency search range of the squeal. The disk brake system has complex eigenvalues and their distribution is on the complex plane. Numerically we can only find eigenvalues in a certain circle on the complex plane. The Rayleigh–Ritz triplets found by the Lanczos procedure actually approximate the outer eigentriplets of a matrix (Appendix A.2). The shift-and-invert spectral transformation is used to find the inner eigenvalues of a matrix.

Mathematically, if (λ, x) is an eigenpair for matrix A, i.e., $Ax = \lambda x$, and $\sigma \neq \lambda$, then the shift-and-invert eigenvalue problem is

$$(A - \sigma I)^{-1} x = \lambda' x, \tag{14}$$

where $\lambda' = 1/\lambda - \sigma$. This spectral transformation would effectively find the eigenvalues nearby the shift σ since the eigenvalues with largest magnitudes correspond to the nearby eigenvalues that are nearest to the shift. Similarly, for a general eigenvalue problem as Eq. (12), by using the shift-and-invert spectral transformation, it can be reduced to a standard eigenvalue problem

$$(A - \sigma B)^{-1} B x = \frac{1}{\lambda - \sigma} x. \tag{15}$$

The audible noise of a disk brake system is usually in the range from 20 Hz to 20 kHz. If we set the shift to zero, i.e., let $\sigma = 0$, then ABLE can find the desired number of eigenvalues in the circle centering at origin (0, 0) on the complex plane. By checking the frequencies found, ABLE can find eigenvalues within the user specified frequency range.

A potential drawback of the shift-and-invert technique is to find the inverse of a matrix, which will be used for matrix–vector multiplication during iteration. Instead of using the inverse of a matrix, the LU factorization of the matrix is used.

The only problem left is to find an LU factorization for a large sparse unsymmetrical matrix. It is desired that the resulting L and U are not too dense, i.e., not too many non-zeros after factorization. The reason is that too many non-zeros of L and U will require a lot of memory and the sparsity of LU will largely affect the speed of matrix–vector multiplication and thus the whole iterative speed. A recently developed unsymmetric-pattern multi-frontal method by Davis and Duff [12] is used in this paper to perform the LU factorization.

Since we use shift-and-invert for the eigenvalue problem, an LU factorization for matrix $[K]$ is required when $\alpha = 0$. When $\alpha \neq 0$, the QEP of Eq. (10) needs to be solved. By using the linearization form of Eq. (11), we only need to perform the LU factorization on the original matrix $[K]$ instead of the augmented matrix A. The proof is as follows.

Assume that $[K]=[L_1][U_1]$. From Eq. (11),

$$[A]=\begin{bmatrix} K & 0 \\ 0 & I \end{bmatrix}=LU=\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}\begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}=\begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12}+L_{22}U_{22} \end{bmatrix}.$$

Then, $[L_1]=[L_{11}]$, $[U_1]=[U_{11}]$. Since in the application of disk brake squeal prediction, matrix $[K]$ is non-singular, both $[L_1]$ and $[U_1]$ are not singular. Thus we have $[U_{12}]=[L_{21}]=[0]$ and $[I]=[L_{22}][U_{22}]$. The simplest form of $[L_{22}]$ and $[U_{22}]$ would be the identity matrix. Thus the LU factorization of $[A]$ can be easily written as

$$[A]=\begin{bmatrix} K & 0 \\ 0 & I \end{bmatrix}=\begin{bmatrix} L & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} U & 0 \\ 0 & I \end{bmatrix}.$$

Hence by using the linearization form of Eq. (11), the total CPU time and memory requirement can be dramatically reduced.

6. Numerical examples

In this section, three test cases for two different geometry models are given to demonstrate the ABLE algorithm. In the first two test cases, α is equal to zero. The eigenfrequencies of the unstable modes predicted by the current approach and the component mode synthesis (CMS[1]) are compared to validate both the FEM formulation and the ABLE algorithm. The third test case is an $\alpha \neq 0$ case.

Two different models are used for the two $\alpha = 0$ test cases. The first test case uses a coarse mesh model, which has a total of 1319 second order 10-node solid parabolic tetrahedron elements with a total of 2812 nodes. Ground springs and fixed d.o.f.s on certain nodes are used as the boundary conditions. The total number of d.o.f.s in this model is 7998. The FEM mesh is shown in Fig. 3.

The second test case uses a finer mesh model, which has a total number of 15,376 second order 10-node solid parabolic tetrahedron elements with a total of 29,978 nodes. Ground springs and fixed d.o.f.s on certain nodes are used as the boundary conditions. The total number of d.o.f.s in this model is 89,754. The FEM mesh for the second test case is shown in Fig. 4.

The third test case is for $\alpha \neq 0$. In this test case, the finer mesh model is used but small changes on the boundary conditions are applied.

In all three test cases, the friction coefficient μ is set to 0.4 and we let ABLE find the first 100 modes for each case. The computation device is a PC with an AMD Athlon XP2000 + CPU and 1.5 GB RAM.

For the first test case, the sparsity of stiffness matrix is 0.778% and that of the mass matrix is 0.013%. The ABLE algorithm uses 37 s to find the first 100 modes for this system. Predicted unstable modes by both the current approach and the CMS are listed in Table 2. Very good agreement is observed between the current approach and the CMS.

[1] It was performed independently by Akebono Corporation.

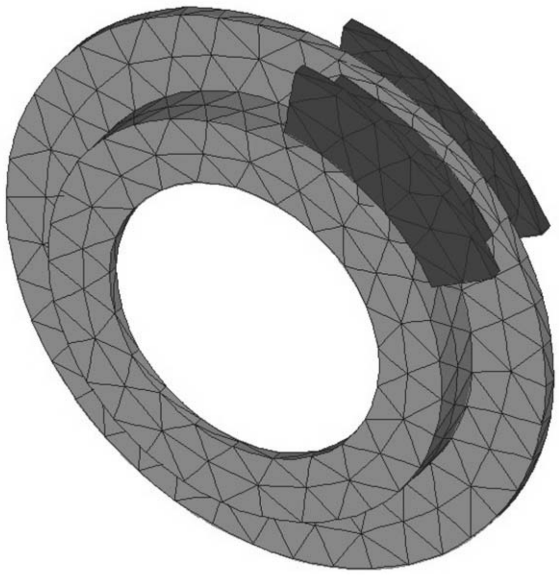
Fig. 3. FEM mesh of model 1.

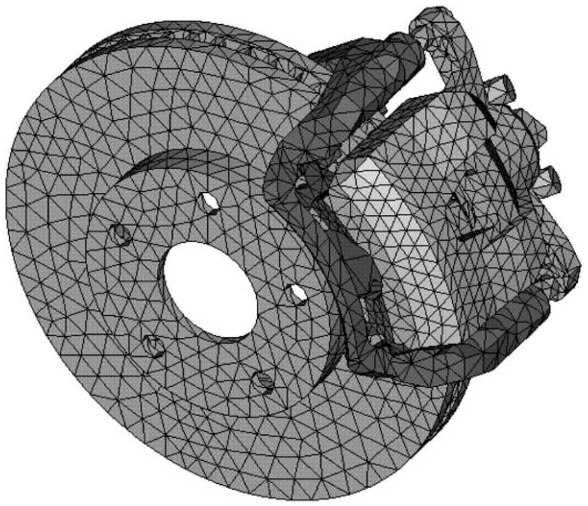
Fig. 4. FEM mesh of model 2.

For the second test case, the sparsity of the stiffness matrix and the mass matrix is 0.0770% and 0.0011%, respectively. The time for ABLE to find the first 100 modes is 639 s (about 11 min). Only one unstable mode is predicted by both the current approach and the CMS. The comparison is shown in Table 3.

In the third test case, ABLE is used to find the eigenvalues for a QEP. Note for a QEP, the order of the matrix is doubled. Thus in this case, the total number of d.o.f.s is 179,508 and the sparsity of matrix A and matrix B in Eq. (12) is 0.0195% and 0.00056%, respectively. From the second test case, it is seen that the first unstable mode for $\alpha = 0$ is at 8.043 kHz. Since current

Table 2
Comparisons between the current approach and the component mode synthesis for test case 1 (unstable modes in the first 100 modes, $\mu = 0.4$, $\alpha = 0$)

Current approach		CMS	
Frequency (kHz)	Propensity	Frequency (kHz)	Propensity
8.922	5.84	8.920	8.44
11.118	4.32e+03	11.140	4.33e+03
11.827	65.90	11.826	66.30
12.218	129.0	12.172	133.5
15.729	132.0	15.727	133.2
20.975	2.67e+03	20.983	2.68e+03
22.217	1.03e+03	22.223	1.01e+03
22.272	1.92e+03	22.280	1.91e+03
22.985	651.0	22.982	650.9

Table 3
Comparisons between the current approach and the component mode synthesis for test case 2 (unstable modes in the first 100 modes, $\mu = 0.4$, $\alpha = 0$)

Current approach		CMS	
Frequency (kHz)	Propensity	Frequency (kHz)	Propensity
8.043	5.00	8.043	7.02

implementation of ABLE is all in-core, due to the 1.5 GB memory limitation on our PC, we cannot find any eigenfrequencies around 8.043 kHz for the QEP when the order of the matrix is doubled. Therefore, we have slightly changed the boundary conditions in the second test model to generate some low-frequency unstable modes. Then we re-run the $\alpha = 0$ test case after the change of boundary conditions. Two unstable modes are captured by ABLE when $\alpha = 0$: one at 2.783 kHz with propensity equal to 123 and the other at 8.043 kHz with propensity equal to 6. Then we set $\alpha = -1 \times 10^{-5}$ s/mm and apply a uniform pressure of 0.5 MPa to each of the two pads. The angular frequency of the rotor is set at 0.75 Hz. The time used by ABLE to find the first 100 modes of the system is 1253 s (about 21 min, twice as the $\alpha = 0$ case). The unstable modes found by ABLE is listed in Table 4. More unstable modes are captured due to $\alpha \neq 0$.

7. Conclusions

An FEM formulation of friction-induced vibration to predict disk brake squeal is presented in this paper. A snapshot at steady sliding is first taken and the static equilibrium equation is solved. Then small perturbation is applied to the equilibrium state to obtain the dynamic system equation for stability study. Due to the friction between the pads and the rotor, the FEM matrices become unsymmetric, which results in complex eigenvalues. Any eigenvalue with a positive real part, which is defined as the propensity, may indicate an unstable mode. The bigger the propensity is, the most likely the mode is an unstable mode.

Table 4
Unstable modes predicted by current approach for test case 3 (unstable modes in the first 100 modes, $\mu = 0.4$, $\alpha \neq 0$)

No.	Frequency (kHz)	Propensity
1	0.805	12.6
2	1.806	165.6
3	2.176	12.6
4	2.783	120.1
5	3.071	60.97
6	3.249	34.4
7	3.389	14.24
8	4.376	11.9
9	4.898	23.8

A negative $\mu - v$ slope is assumed in the kinetic friction coefficient. When $\alpha = 0$ (which means the friction coefficient is a constant), solving a QEP can be avoided because the damping matrix is zero. When $\alpha \neq 0$, the size of the problem is doubled after the QEP is linearized. Nevertheless, a special linearization form of the QEP is used (Eq. (11)) so that the LU factorization is needed only for the stiffness matrix, instead of for the augmented matrix. This is an important step because the LU factorization could become the bottleneck of the whole procedure.

The recently developed ABLE algorithm is used to solve the complex eigenvalue problem. ABLE is an adaptive block Lanczos method for sparse unsymmetric matrices. The ABLE algorithm also avoids breakdowns and eliminates slow convergence by adaptively changing the block size and maintaining the full biorthogonality of the Lanczos vectors. Numerical examples show that the unstable modes predicted by ABLE are in good agreement with the results from the CMS when $\alpha = 0$. When $\alpha \neq 0$, no CMS results are available for comparison, but the linearization form of the QEP along with ABLE has shown very promising initial results.

The main memory is used for storing L, U factors and the Lanczos vectors. If we can adaptively involve out-of-core implementation of the algorithm when the problem size increases, the size of the problem to be solved on PC can be increased a lot.

Acknowledgements

This research was supported by Akebono Corporation. The authors would also like to thank L. Lee and K. Xu at Akebono Corporation for providing the CMS results for comparison.

Appendix A. Procedures and codes

A.1. Basic block Lanczos procedure

The basic block non-Hermitian Lanczos algorithm is presented in Fig. 5. It is a variation of the original Lanczos procedure.

```
1. Choose starting n×p vectors $P_1$ and $Q_1$ so that $P_1^H Q_1 = I$
2. $R = (P_1^H A)^H; S = AQ_1$
3. for j=1,2,…
    1. $A_j = P_j^H S$
    2. $R = R - P_j A_j^H$; $S = S - Q_j A_j$
    3. compute the QR decompositions: $R = P_{j+1} B_{j+1}^H$; $S = Q_{j+1} C_{j+1}$
    4. compute singular value decomposition: $P_{j+1}^H Q_{j+1} = U_j \Sigma_j V_j^H$
    5. if $\Sigma_j$ is (nearly) singular, stop
    6. $B_{j+1} = B_{j+1} U_j \Sigma_j^{1/2}$; $C_{j+1} = \Sigma_j^{1/2} V_j^H C_{j+1}$
    7. $P_{j+1} = P_{j+1} U_j \Sigma_j^{-1/2}$; $Q_{j+1} = Q_{j+1} V_j \Sigma_j^{-1/2}$
    8. $R = (P_{j+1}^H A - C_{j+1} P_j^H)^H$; $S = AQ_{j+1} - Q_j B_{j+1}$
```

Fig. 5. Simple block non-Hermitian Lanczos algorithm.

In the following derivation, superscript "H" denotes the conjugate transpose operator for a matrix and subscript "j" denotes the Lanczos step. Given an n-by-n matrix A and initial n-by-p block vectors P_1 and Q_1 with $P_1^H Q_1 = I$, the block Lanczos procedure uses the following short three term recurrences:

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

$$Q_{j+1} C_{j+1} = AQ_j - Q_j A_j - Q_{j-1} B_j \tag{A.2}$$

to generate two sequences of $n \times p$ multiple vectors $Q_1, Q_2, \dots, Q_j$ and $P_1, P_2, \dots, P_j (P_0 = Q_0 = 0)$, namely the left and right Lanczos vectors. These vectors are the biorthonormal bases of the Krylov subspaces

$$K_j(A, Q_1) = span\{Q_1, AQ_1, A^2 Q_1, \dots, A^{j-1} Q_1\}$$

and

$$K_j(A^H, P_1) = span\{P_1, A^H P_1, (A^H)^2 P_1, \dots, (A^H)^{j-1} P_1\}.$$

After j steps, the Lanczos procedure determines a block tridiagonal matrix

$$T_j = \begin{bmatrix} A_1 & B_2 & & \\ C_2 & A_2 & \ddots & \\ & \ddots & \ddots & B_j \\ & & C_j & A_j \end{bmatrix}, \tag{A.3}$$

and matrices of the left and right Lanczos vectors

$$\mathscr{P}_j = (P_1, P_2, \dots, P_j) \text{ and } \mathscr{Q}_j = (Q_1, Q_2, \dots, Q_j).$$

They satisfy the governing relations

$$\mathcal{P}_j^H A = T_j \mathcal{P}_j^H + E_j B_{j+1} P_{j+1}^H, \tag{A.4}$$

$$A \mathcal{Q}_j = \mathcal{Q}_j T_j + Q_{j+1} C_{j+1} E_j^H, \tag{A.5}$$

where E_j is a tall skinny matrix whose bottom square is an identity matrix and elsewhere are zeros. Furthermore, $\mathcal{P}_j$ and $\mathcal{Q}_j$ satisfy the biorthonormal condition

$$\mathcal{P}_j^H \mathcal{Q}_j = I. \tag{A.6}$$

A.2. Eigenvalue approximation

To use the Lanczos procedure to approximate the eigenvalues and the eigenvectors of A, we solve the eigenvalue problem of the $jp \times jp$ block tridiagonal matrix T_j after step 3.3 in Fig. 5. Each eigentriplet (i.e., eigenvalue and left/right eigenvector θ, w^H, z) of T_j,

$$w^H T_j = \theta w^H \quad \text{and} \quad T_j z = z\theta,$$

determines a Rayleigh–Ritz triplet (θ, y^H, x), where $y^H = w^H \mathcal{P}_j^H$ and $x = \mathcal{Q}_j z$. Rayleigh–Ritz triplets usually approximate the outer eigentriplets of A.

Let s and r denote the corresponding left and right residual vectors. By Eqs. (A.4) and (A.5), we have

$$s^H = y^H A - \theta y^H = (w^H E_j) B_{j+1} P_{j+1}^H, \tag{A.7}$$

$$r = Ax - x\theta = Q_{j+1} C_{j+1} (E_j^H z). \tag{A.8}$$

The residuals determine a backward error bound for the triplet. From the biorthonormal condition, Eq. (A.6), we combine Eqs. (A.7) and (A.8) to get

$$y^H (A - F) = \theta y^H, \tag{A.9}$$

$$(A - F)x = x\theta, \tag{A.10}$$

where the backward error matrix F is

$$F = \frac{r x^H}{\|x\|_2^2} + \frac{y s^H}{\|y\|_2^2}. \tag{A.11}$$

Note that $\|F\|_F^2 = \|r\|_2^2/\|x\|_2^2 + \|s^H\|_2^2/\|y\|_2^2$. Thus the left and right residual norms bound the distance to the nearby matrix to A with eigentriplet (θ, y^H, x).

By examining Eqs. (A.7) and (A.8), we notice that one of the remarkable features of the Lanczos algorithm is that the residual norms $\|s^H\|_2$ and $\|r\|_2$ are available without explicitly computing y^H and x. There is no need to form y^H and x until their accuracy is satisfactory.

The Lanczos algorithm may breakdown before convergence because the matrix $P_{j+1}^H Q_{j+1}$ (Fig. 5, step 3.4) is singular. Moreover the computed Rayleigh–Ritz triplets may converge slowly due to the loss of biorthogonality among the computed Lanczos vectors $\mathcal{P}_j$ and $\mathcal{Q}_j$, or the block size is smaller than the order of cluster of the desired eigenvalues. The ABLE algorithm avoids breakdowns and eliminates slow convergence by adaptively changing the block size and maintaining the full biorthogonality of the Lanczos vectors, i.e., $\|\mathcal{P}_j^H \mathcal{Q}_j - I\| \approx \varepsilon$, or semi-biorthogonality of the Lanczos vectors, i.e., $\|\mathcal{P}_j^H \mathcal{Q}_j - I\| \approx \sqrt{\varepsilon}$, where ε is the machine epsilon.

A.3. Pseudo code of ABLE

```
1. Choose starting vector $P1$ and $Q1$ $P_1^H Q_1 = I$
2. $R = (P_1^H A)^H$;  $S = AQ_1$
3. for $j = 1, 2, \ldots$
    1. $A_j = P_j^H S$
    2. $R = R - P_j A_j^H$;  $S = S - Q_j A_j$
    3. compute the eigendecomposition of $T_j$, and test for convergence
    4. $R = R - P_j A_j^H$; $S = S - Q_j(P_j^H S)$
    5. compute the OR decompositions: $R = P_{j+1} B_{j+1}^H$; $S = Q_{j+1} C_{j+1}$
    6. if R or S is rank deficient and (*fulldual=true* or *semidual=true*)
            call TSMGS to biorthogonalization
       end if
    7. compute the singular value decomposition: $P_{j+1}^H Q_{j+1} = U \Sigma V^H$
    8. if $w = min(diag(S)) < tolbd$,
            *breakdown=true*
            if *breakdown=true* and *treatbd=false*
                method fails
            end if
       end if
    9. if *group=true* and (*fulldual=true* or *semidual=true*)
            compute the maximal order of converged RR values
       end if
    10. if (*breakdown=true* and *treatbd=true*) or *group=true*
            call CURE to cure breakdown or adapt to the cluster
        end if
    11. $B_{j+1} = B_{j+1} U \Sigma^{1/2}$; $C_{j+1} = \Sigma^{1/2} V^H C_{j+1}$; $P_{j+1} U \Sigma^{-1/2}$; $Q_{j+1} = Q_{j+1} V \Sigma^{-1/2}$
    12. if *fulldual=true*
            call TSMGS to maintain full biorthogonality
        else if *semidual=true*
            call SEMI to monitor the loss biorthogonality and correction if
            necessary
        end if
    13. $R = (P_{j+1}^H A - C_{j+1} P_j^H)^H$; $S = AQ_{j+1} - Q_j B_{j+1}$
4. end
```

References

[1] Y. Yuan, An eigenvalue analysis approach to brake squeal problems, *Proceedings of the Dedicated Conference on Automotive Braking Systems 29th ISATA*, Florence, Italy, 1996.

[2] N. Miller, An analysis of disk brake squeal, SAE Paper No.780332, 1978.
[3] W.V. Nack, Brake squeal analysis by finite elements, *International Journal of Vehicle Design* 23 (3–4) (2000) 263–275.
[4] G. Lilies, Analysis of disk brake squeal using finite element methods, SAE Paper No. 891150, 1989.
[5] I. Kido, T. Kuachachi, M. Asai, A study on low-frequency brake squeal noise, SAE Paper No. 960993, 1996.
[6] T. Matsushima, H. Masumo, S. Ito, M. Nishiwak, FE analysis of low-frequency disk brake squeal (in case of floating type caliper), SAE Paper No. 982251, 1998.
[7] J. Cullum, R. Willoughby, Large scale eigenvalue problems, in: J. Cullum, R. Willoughby (Eds.), *A Practical Procedure for Computing Eigenvalues of Large Sparse Nonsymmetric Matrices*, North-Holland, Amsterdam, 1986.
[8] J. Cullum, W. Kerner, R. Willoughby, A generalized nonsymmetric Lanczos procedure, *Computer Physics Communications* 53 (1989) 19–48.
[9] Z. Bai, D. Day, Templates for the solution of algebraic eigenvalue problems, in: Z. Bai, J. Demmel, J. Dongarra, R. Ruhe, H. van der Vorst (Eds.), *A Practical Guide*, Section 7.9, Block Lanczos methods, SIAM, Philadelphia, PA, 2000, pp. 196–205.
[10] Q. Ye, A breakdown-free variation of the nonsymmetric Lanczos algorithm, *Mathematics of Computation* 62 (1994) 179–207.
[11] Z. Bai, D. Day, Q. Ye, ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems, *SIAM Journal on Matrix Analysis and Applications* 20 (1999) 1060–1082.
[12] T.A. Davis, I.S. Duff, An unsymmetric-pattern multifrontal method for sparse LU factorization, *SIAM Journal on Matrix Analysis and Applications* 18 (1997) 140–158.