Journal of Computational and Applied Mathematics 234 (2010) 3216–3225

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Saving flops in LU based shift-and-invert strategy

Laura Grigori [a], Desire Nuentsa Wakam [b], Hua Xiang [c,*,1]

[a] INRIA Saclay-Ile de France, Laboratoire de Recherche en Informatique, Bât 490 Université Paris-Sud 11, 91405 Orsay Cedex, France

[b] INRIA IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France

[c] School of Mathematics and Statistics, Wuhan University, Wuhan 430072, PR China

ARTICLE INFO

Article history:
Received 11 June 2008
Received in revised form 2 March 2010

MSC:
15A18
15A23
34L16
65F05
65F15

Keywords:
Shift-and-invert
Eigenvalue
Divide and conquer
LU factorization

ABSTRACT

The shift-and-invert method is very efficient in eigenvalue computations, in particular when interior eigenvalues are sought. This method involves solving linear systems of the form $(A - \sigma I)z = b$. The shift σ is variable, hence when a direct method is used to solve the linear system, the LU factorization of $(A - \sigma I)$ needs to be computed for every shift change. We present two strategies that reduce the number of floating point operations performed in the LU factorization when the shift changes. Both methods perform first a preprocessing step that aims at eliminating parts of the matrix that are not affected by the diagonal change. This leads to about 43% and 50% flops savings respectively for the dense matrices.

1. Introduction

The standard eigenvalue problem,

$$Ax = \lambda x, \tag{1}$$

where $x \neq 0$, $\lambda \in \mathbb{C}$, has broad applications in mechanics, physics, chemistry and economics. It is still a very challenging problem, even if there are many practical methods and software available [1]. A basic method in eigenvalue computation is the power method. Although simple, many methods are rooted in it. One enhancement of the power method is the inverse iteration, which applies the power method to $(A - \sigma I)^{-1}$, where σ is a shift. This method can converge to any desired eigenvalue, especially when a few interior eigenvalues are sought [2]. The iteration step can be expressed as

$$v_k = \frac{1}{\alpha_k}(A - \sigma I)^{-1} v_{k-1} \quad (k = 1, 2, \ldots), \tag{2}$$

where v_0 is the initial guess. The most expensive step in this computation is to find the solution of a linear system of the form

$$(A - \sigma I)z = b. \tag{3}$$

* Corresponding author.
E-mail addresses: laura.grigori@inria.fr (L. Grigori), dnuentsa@irisa.fr (D.N. Wakam), hxiang@whu.edu.cn, xiang@ann.jussieu.fr (H. Xiang).

[1] Most of the work was performed at INRIA Saclay-Ile de France, and LRI, Université Paris-Sud 11.

0377-0427/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2010.04.003

Note that $A - \sigma I$ is ill-conditioned when σ is close to the true eigenvalue. But most of the inaccuracies of the solution are in the direction of the eigenvector being approximated [3,4]. If we have a better approximation of an eigenvalue, we can change the shift occasionally. Because α_k in (2) converges to $1/(\lambda_j - \sigma)$, it is natural to take $\sigma_{\text{new}} = \sigma_{\text{old}} + \frac{1}{\alpha_k}$ [5]. If the Rayleigh quotient is used as the shift, then this method is called Rayleigh quotient iteration (RQI). For non-Hermitian matrices, the generalized Rayleigh quotient is used, $\sigma_j = y_j^* A x_j / y_j^* x_j$, where x and y are the approximate left eigenvector and right eigenvector respectively in the j-th step [6]. RQI also appears in the QR algorithm in a disguised form [7].

For large sparse eigenvalue problems, Krylov subspace methods are generally used, such as implicitly restarted Arnoldi (IRA) [8], or Bi-side Lanczos with look-ahead strategy [9]. Krylov subspace methods are good at computing the eigenvalues on the periphery of the spectrum [3,5,7,10]. Usually these exterior eigenvalues are well-approximated first, and the interior eigenvalues follow much later. If we need the interior eigenvalues, a spectral transformation like shift-and-invert $(A-\sigma I)^{-1}$ is needed to find the interior eigenvalues close to σ. For example, the Alfven spectrum is an interior part of the spectrum, and without shift-and-invert it is almost impossible to compute this part with Krylov subspace methods [11]. When we apply the Arnoldi or Lanczos method to $(A - \sigma I)^{-1}$, we must solve a sequence of linear equations accurately in order to capture the desired eigenvalues. The shift-and-invert strategy is also used implicitly in Krylov subspace methods. For example, the harmonic Rayleigh–Ritz procedure is related to the shift-and-invert $(A - \sigma I)^{-1}$, but it avoids the matrix inversion by its clever formulation to a projected generalized eigenvalue problem. Linear systems similar to (3) appear implicitly in the Jacobi–Davidson method [12–16]. This method expands the current subspace by computing an approximate solution t to a so-called correction equation, which is equivalent to $t = -u + \alpha(A - \sigma I)^{-1}u$, where α is chosen such that $t \perp u$ [15]. The shift-and-invert strategy also appears in the rational Krylov subspace method [17,18] and the truncated RQ iteration [19,20].

Hence, many methods are related to the shift-and-invert, and this strategy is very efficient. The potential drawback is that linear systems such as (3) need to be solved, which is the most expensive step of the strategy. The accuracy of the linear solver must be in accordance with the convergence tolerance of the eigensolver [21–23] otherwise loss of accuracy in solving (3) may result in the corruption of the Krylov subspace. In this paper, we focus on the shift-and-invert method with variable shifts as used in the dense standard eigenvalue problem. When the LU factorization is used to solve the system and the shift σ_j changes, the LU factorization needs to be performed again. We develop two strategies which consist in performing a pre-processing step such that the LU factorization is not computed from scratch when the shift changes. The pre-processing step annihilates some parts of the matrix A which are not influenced by the change of the diagonal elements. The first strategy is a divide and conquer strategy. We use a recursive 2×2 partition and symmetric permutation, and factorize the original matrix into a staircase shape. For each shift change, the factorization starts from this shape. In this process BLAS-3 operations can be used to achieve high performance, and about 43% flops can be saved for each shift change. For the second strategy, we use two row permutations and one column permutation during each column elimination to control the position of original diagonal elements, such that their influence during updating is confined in the right part of the matrix. This strategy leads to about 50% flops savings. These two strategies are discussed in detail in Sections 2 and 3 respectively. We give numerical examples in Section 4 that check the numerical stability, the flops saving and the efficiency of the two strategies. We conclude in Section 5.

2. Strategy I: a divide and conquer approach

The classical Gaussian elimination with partial pivoting of the matrix $A \in \mathbb{R}^{n\times n}$ can be expressed as

$$L_{n-1}^{-1} P_{n-1} L_{n-2}^{-1} P_{n-2} \cdots L_1^{-1} P_1 A = U, \tag{4}$$

where U is an upper triangular matrix, P_i $(i = 1, \ldots, n-1)$ are permutation matrices, L_i^{-1} $(i = 1, \ldots, n-1)$ are Gauss transformation matrices computed as $L_i = I + l_i e_i^T$, and l_i is the Gauss vector [24]. The Gaussian elimination (4) can be rewritten as

$$L_{n-1}^{-1} \widehat{L}_{n-2}^{-1} \cdots \widehat{L}_1^{-1} PA = U, \tag{5}$$

where $\widehat{L}_i^{-1} = P_{n-1} \cdots P_{i+1} L_i^{-1} P_{i+1}^T \cdots P_{n-1}^T$ $(i = 1, \ldots, n-2)$, $P = P_{n-1} P_{n-2} \cdots P_1$. For simplicity, we further define $L = \widehat{L}_1 \widehat{L}_2 \cdots \widehat{L}_{n-2} L_{n-1}$, so we achieve the LU factorization $PA = LU$. Note that in a real implementation, L and U can be stored in place of the matrix A.

Clearly, if the diagonal elements of A change, this affects the whole factorization. That is the reason why $(A-\sigma I)$ needs to be factorized again when the shift σ changes. Our goal is to restrict the influence of the diagonal elements, and reuse some eliminations at the next step. We can achieve this goal by using a recursive 2×2 partition and symmetric permutations.

First we illustrate our approach on a simple case. Suppose that A is partitioned into 2×2 blocks. We denote it as $A = [A_{11}, A_{12}; A_{21}, A_{22}]$ (see Fig. 1(a)). We can do some eliminations without modifying the diagonal elements. In the following we use three steps to explain the basic idea.

Step 1. *Local LU.* We perform the LU factorization on (2, 1) block. We formulate it as

$$PA_{21} = L^{(1)} U^{(1)}, \tag{6}$$

where $L^{(1)}$ is stored in the lower part of A_{21}, which is shown in Fig. 1(a).