New Accelerated Mixed-Precision Parallel Algorithms for Hermitian Eigenvalue Problem

Yaohung Tsai Piotr Luszczek Jack Dongarra

I. Introduction and Contributions

The motivation of this paper is to develop a mixed-precision algorithm for the symmetric/hermitian eigenvalue problem. In this context, we made the following contributions:

- We formulated and implemented new SICE-SM algorithm for symmetric and Hermitian matrices using Sherman-Morrison formula to solve the resulting tridiagonal systems with rank-one updates from SICE algorithm proposed by Dongarra et al. [1] for iteratively refining a pair of eigenvalues and corresponding eigenvectors.
- We also developed new blocked SICE-SM algorithm to refine multiple eigenvalues and corresponding eigenvectors simultaneously and improve the performance by aggregating the matrix-vector multiplication operations for improved utilization of the CPU/GPU memory hierarchy.
- We implemented a mixed-precision algorithm by scheduling the tasks in the 2-stage eigensolver on the machines with heterogeneous architecture and by improving the utilization of both the CPU and GPU.
- We achieved many-fold performance improvement of the mixed-precision algorithm with the use of iterative refinement when a subset of eigenpairs are requested.

II. The Oirignal SICE Algorithm

The SICE (Subroutine for Improving Computed Eigevalues) algorithm [1]–[3] improves accuracy of eigenproblem $Ax = \lambda x$ with additive modification $\tilde{y}$:

$$A(x+\tilde{y}) = (\lambda+\mu)(x+\tilde{y}) \tag{1}$$

where the initial eigenpair λ, x and is corrected to its nearby eigenpair $\lambda+\mu, x+\tilde{y}$. To remove excessive degrees of freedom we require $\tilde{y}_s = 0$ and assume that x is normalized in infinite norm: $|x|_\infty = 1 \equiv x_s$. By rearranging terms in Eq. (1) we get:

$$(A-\lambda I)\tilde{y} - \mu x = \lambda x - Ax - \mu\tilde{y} \tag{2}$$

This material is also based upon work supported by the National Science Foundation under OAC Grant No. 2004541 and the University of Tennessee grant MSE E01-1315-038 as Interdisciplinary Seed funding. This research used the computational resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725 provided by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The software library integration work was supported by the National Science Foundation under OAC Grant No. 2004541.

We observe that the last term is the second-order error in λ and x. By simplify Eq. 2, we introduce vector y, defined as:

$$y^\intercal \triangleq (\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_{s-1}, \mu, \tilde{y}_{s+1}, \ldots, \tilde{y}_{n-1}, \tilde{y}_n) \tag{3}$$

Thus, y encodes information from both $\tilde{y}$ and μ and Eq. (2) becomes:

$$By = r + y_s\tilde{y} = r + \mu\tilde{y} \tag{4}$$

where $r = \lambda x - Ax$ is the residual vector of λ and x and B is the matrix $A-\lambda I$ with column s replaced by $-x$.

We can also view it as the Newton's method. In particular, by setting $v = \binom{x}{\lambda}$ we can be formulate the eigenvalue problem as:

$$f(v) \equiv \begin{pmatrix} Ax - \lambda x \\ e_s^\intercal x - 1 \end{pmatrix} = 0 \tag{5}$$

where e_s is the s-th column of the identity matrix of size n. The Newton's method then solves the linear system of the Jacobian matrix:

$$J\begin{pmatrix}\tilde{y}\\ \mu\end{pmatrix} = \begin{pmatrix} A-\lambda I & -x \\ e_s^\intercal & 0 \end{pmatrix}\begin{pmatrix}\tilde{y}\\ \mu\end{pmatrix} = \begin{pmatrix} r \\ 0\end{pmatrix} = f(v) \tag{6}$$

Expanding it, we arrive at Eq. (2) without the second-order term:

$$(A-\lambda I)\tilde{y} - \mu x = r \tag{7}$$

By rewriting Eq. (4), we get:

$$[A_\lambda - (x+a_{\lambda s})e_s^\intercal]y = (A + ce_s^\intercal)y = r + y_s\tilde{y} \tag{8}$$

where $c = -x - a_{\lambda s}$. Using the Schur decomposition $A = QUQ^\intercal$, we have:

$$Q(U_\lambda + Q^\intercal c e_s^\intercal Q)Q^\intercal y = r + y_s\tilde{y} \tag{9}$$

$$(U_\lambda + df^\intercal)Q^\intercal y = Q^\intercal g \tag{10}$$

where $d = Q^\intercal c$, $f^\intercal = e_s^\intercal Q$ and $g = r + y_s\tilde{y}$.

Full text of SICE algorithm is given in the poster's PDF.

III. New SICE-SM Algorithm

For symmetric eigenvalue problems, the matrix A is first reduced to tridiagonal through unitary similarity transformations: $T = Q^\intercal AQ$ where $QQ^\intercal = I$ and T is a symmetric tridiagonal matrix. This corresponds to LAPACK routines `SSYTRD` and `DSYTRD` for single- and double-precision arithmetic, respectively. In the same fashion as SICE algorithm in Section II, we start with Eq. (8) and apply the tridiagonal reduction to it. Eqs. (9) and (10) in this case become

$$Q\left(T_\lambda + Q^\intercal c e_s^\intercal Q\right)Q^\intercal y = r + y_s\tilde{y} \tag{11}$$

$$\left(T_\lambda + d \times f^\intercal\right)Q^\intercal y = Q^\intercal g \tag{12}$$

TABLE I
PERFORMANCE OF $n \times n$ MATRIX TIMES $n \times m$ AGGREGATED VECTORS ON NVIDIA V100-SXM2-32GB GPU, DGEMM ROUTINE FROM CUBLAS V11.0.

Matrix size	Num. of vectors	Time (ms)	Performance (GFLOP/s)
20000	1	3.76	212.65
20000	8	3.79	1688.17
20000	32	6.48	3949.32
20000	128	13.57	7544.43

the same with $d = Q^\intercal c$, $f^\intercal = e_s^\intercal Q$ and $g = r + y_s\tilde{y}$.

Dongarra [2] discussed the approach of using the Sherman–Morrison formula [4]

$$(A - uv^\intercal)^{-1} = A^{-1} - \frac{A^{-1}uv^\intercal A^{-1}}{1 + v^\intercal A^{-1}u} \tag{13}$$

for solving the rank-one updated system. Eq. (12) does not apply since $T_\lambda = T - \lambda I$ is singular by construction. However, this may not be so in mixed-precision setting. Consider the scheme that first performs the tridiagonal reduction in single precision and then solves the tridiagonal eigenvalue problem in double precision. The initial λ_T will be the eigenvalue of T with double-precision accuracy, but it only approximates λ_A, the eigenvalue of A with single-precision accuracy. With suitably chosen offset δ of order of $\varepsilon_{\text{fp32}}$, $T - (\lambda + \delta)I$ will no longer be singular in double precision, and the Sherman–Morrison formula can be applied. The special case in which this would fail is when $\|\lambda_T - \lambda_A\| = O(\varepsilon_{\text{fp64}})$: the initial eigenvalue is also an accurate eigenvalue of A in double precision. In such a case, we do not need to refine the eigenvalue and can simply apply the inverse iteration to find the eigenvector.

Full text of SICE-SM algorithm is given in the poster's PDF.

IV. New Blocked SICE-SM Algorithm

The computational cost of SICE-SM is dominated by matrix-vector multiplications especially inside the refinement iteration. In the matrix-vector multiplication, the whole matrix is read once and only a single multiplication and addition are performed per each of the fetched elements. This results in a low arithmetic intensity of 2, which results in very low inefficient on modern hardware including CPU, GPUs, and computational accelerators. To improve on this implementation aspect, we can aggregate several eigenpairs simultaneously and refine them at the same time while they are cached in higher levels of the memory hierarchy. This blocking strategy is common in numerical linear algebra since it was introduced in LAPACK [5] and relies on grouping computations so that Level 3 Basic Linear Algebra Subprograms (BLAS) may be utilized to perform operations that are rich in matrix-matrix multilications. These operations perform more efficiently as they have higher arithmetic intensity resulting from higher data reuse in fast portions of the cache hierarchy. In our case, we assume that the matrix size is far greater than the number of eigenpairs to refine. Then the matrix-vector multiplication is dominated by the reading of the matrix elements. And with the blocked version, it the additional cost of refining extra eigenpairs is negligible. In Table I, we show examples of the performance rates and execution times for different numbers of vectors submitted to the DGEMM routine from cuBLAS on the NVIDIA V100 GPU. The times for 1 and 8 vectors are almost the same. And for 32 or 128 vectors the elapsed time increases 3.6×.

There are a few issues we need to solve while formulating a blocked variant of the algorithm. First, in SICE, the eigenvector is first normalized in infinity norm. The index s is also picked so that $\|x\|_\infty = s_x = 1$. If we allow different s for each of the eigenpairs, then we will have to access different columns in A to construct vector c, and also different rows of Q for vector $f^\intercal$. The row access required for the latter is performed in column major layout and results in non-coalescing memory accesses which are extremely slow and should be avoided as much as possible due to their low utilization of the GPU's memory bandwidth. To show that it is fine to choose s arbitrarily, we need to take a closer look at the matrix in Eq. (11) and expand it without canceling any terms we get

$$(QT_\lambda Q^\intercal + QQ^\intercal v e_s^\intercal QQ^\intercal)y = r + y_s\tilde{y} \tag{14}$$

Again, for our mixed-precision scheme, we would like to perform the tridiagonalization in single precision. Hence $QT_\lambda Q^\intercal$ is only an approximation of A with precision $\varepsilon_{\text{fp32}}$, i.e. $\|A_\lambda - QT_\lambda Q^\intercal\| \sim O(\varepsilon_{\text{fp32}})$. The same applies to $QQ^\intercal$ which is only an approximation of I with $\|QQ^\intercal - I\| \sim O(\varepsilon_{\text{fp32}})$. So no matter which index s we pick, we will always get an error of order $\varepsilon_{\text{fp32}}$ in the correction of eigenvalue y_s coming from the other elements in the solution vector y. There could be a potential problem if the eigenvalue itself is small and the error is preventing the eigenvalue to be refined to desire accuracy. This can be remedied by pre-scaling the matrix so that the eigenvalues are not too small.

The other issue is that by treating the eigenpairs independently they might lose their orthogonality. In the worst case, they might all converge to the same eigenpair. However, it is easy to reorthogonalize with

$$X' = X + \frac{1}{2}X(I - X^\intercal X) \tag{15}$$

In practice, we found that it is sufficient to reorthogonalize after the refinement is done. Doing so in each iteration would not speed up the convergence. The computation of $I - X^\intercal X$ also lets us detect if they converged to the same eigenvector. By combining these considerations, we arrive at blocked SICE-SM algorithm that is given in the poster's PDF.

References

[1] J. J. Dongarra, C. B. Moler, and J. H. Wilkinson, "Improving the accuracy of computed eigenvalues and eigenvectors," *SIAM Journal on Numerical Analysis*, vol. 20, no. 1, pp. 23–45, 1983.

[2] J. J. Dongarra, "Improving the accuracy of computed matrix eigenvalues," Argonne National Lab., Chicago, IL, USA, Tech. Rep., 1980.

[3] ——, "Algorithm 589: SICEDR: a FORTRAN subroutine for improving the accuracy of computed matrix eigenvalues," *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, no. 4, pp. 371–375, 1982.

[4] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950.

[5] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney *et al.*, *LAPACK Users' guide*. SIAM, 1999.