Multi-section with Multiple Eigenvalues Method for Computing Eigenvalues in Symmetric Tridiagonal Eigensolvers

Takahiro Katagiri,[†,††] Christof Vömel[††]
and James W. Demmel [††]

In this paper, a new parallel implementation method for computing eigenvalues in symmetric tridiagonal eigensolver is proposed. In this method, natural parallelism for multiple eigenvalue computations and multi-section points of the eigenvalue searching are used. The performance evaluation results with the HITACHI SR8000 using 8 processors per node indicated that: (1) maximum 7.7x speedup to a case of using conventional bisection method; (2) maximum 4.3x speedup to a case of using conventional multi-section method; were obtained.

1. Introduction

Bisection method is one of widely used methods to compute eigenvalues for the symmetric tridiagonal eigensolvers. The execution time for bisection method is getting heavy, if the target eigenvalues are tightly clustered in a LAPACK routine with MRRR algorithm[1)~5)]. To solve such problem, we need to speedup the part of bisection. Parallelizing the part with shared memory parallel machines is one of ways.

In this paper, a new parallel implementation method of the bisection part for computing eigenvalues is proposed. The target machine is shared memory parallel machine. The main idea for the method is using natural parallelism for computing multiple eigenvalues and multiple searching points for each eigenvalue. We call this method Multi-section with Multiple Eigenvalues (MME) method.

To obtain speedup, using multiple searching points is not new idea. The method is referred to *multi-section method* to bisection method. For example, Lo *et.al.*[6)] and Simon[7)] mentioned the merit to use the multiple searching points in vector machines. Their methods, however, focused on the vectorization to the IF-sentences located in the inside of loop, which is the counting part for the number of eigenvalues less than the value of σ.

Our method proposed in here is focusing on the natural parallelism for the outer loop of the kernel. In addition, the parallelism with multiple eigenvalues computation for the part is also considered. With taking into account of the multiple eigenvalue computations, the length of the outer loop can keep long, even if we take small length for the multi-section points. This is crucial factor to obtain high parallelism and high computation efficiency in shared memory parallel machines. The computation efficiency will be down for the conventional multi-section, if we take long loop length for the multi-section loop. This is because the additional overhead of multi-section cannot be ignored, if the target eigenvalue is found in early iteration time.

This paper is organized as follows. Section 2 explains MME method, and its kernel is derived. In Section 3, performance of MME is evaluated with a LAPACK routine with MRRR algorithm as one of benchmarking application on the HITACHI SR8000. The routine will be provided in LAPACK 4.0 as xSTEGR. Finally, we give a conclusion for this paper.

2. Kernel Derivation for The MME Method

In this chapter, we will derive the MME kernel from kernel of bisection method.

2.1 LAPACK `dlarrb` Routine

First of all, the kernel of bisection method explained in here is based on LAPACK xSTEGR implementation. Figure 1 shows the whole bisection kernel for `dlarrb` routine in LAPACK xSTEGR 4.0. However, the nature of MME method is not limited to the implementation of LAPACK xSTEGR.

The kernel in Figure 1 returns the number of eigenvalues under the value σ. Please note that the kernel in Figure 1 is bisection for target

[†] Graduate School of Information Systems, The University of Electro-Communications, JAPAN.
[††] Computer Science Division, University of California, Berkeley, U.S.A.

one eigenvalue. The total number of eigenvalues should be calculated in `dlarrb` routine is given by the Relatively Robust Representation (RRR)[3].

The value σ is determined by an interval for the target eigenvalue. The interval is given by the Representation Tree[3] in MRRR algorithm.

The `dlarrb` routine performs "limited" bisection to refine the eigenvalues of $L\ D\ L^T$. The IEEE-features[8], for example `NaN`, are used in the kernel to obtain speedup.

```
⟨0⟩ $S = 0$; $P = 0$; $NEG1 = 0$;
    $NEG2 = 0$; $NEGCNT = 0$;
⟨1⟩ I) Upper Part
⟨2⟩ do $J = 1, R-1$
⟨3⟩   $T = S - \sigma$
⟨4⟩   $DPLUS = D(J) + T$
⟨5⟩   $S = T*LLD(J)\ /\ DPLUS$
⟨6⟩   if ( $DPLUS$ .lt. $ZERO$ )
            $NEG1 = NEG1 + 1$
⟨7⟩ enddo
⟨8⟩ if ($S$ .eq. NaN) Use a slower version
      the above loop;
⟨9⟩ $NEGCNT = NEGCNT + NEG1$
⟨10⟩ II) Lower Part
⟨11⟩ do $J = N-1, R, -1$
⟨12⟩   $DMINUS = LLD(J) + P$
⟨13⟩   $P = P*D(J)\ /\ DMINUS - \sigma$
⟨14⟩   if ( $DMINUS$ .lt. $ZERO$ )
            $NEG2 = NEG2 + 1$
⟨15⟩ enddo
⟨16⟩ if ($P$ .eq. NaN) Use a slower version
      the above loop;
⟨17⟩ $NEGCNT =$
       $NEGCNT + NEG2$
⟨18⟩ III) Twist Index
⟨19⟩ $GAMMA = S + P$
⟨20⟩ if ($GAMMA$ .lt. $ZERO$)
         $NEGCNT = NEGCNT + 1$
⟨21⟩ return ($NEGCNT$)
```

Fig. 1 The Whole Bisection Kernel for The `dlarrb` Routine in LAPACK xSTEGR. The variable R is a twisted point for the twisted factorization to the tridiagonal matrix T.

The value of σ in Figure 1 shows the point for the bisection search for the target eigenvalue. The value of σ is calculated by $a + (b-a)/2$ in the bisection method, if the current interval is $[a,b]$.

In the kernel in Figure 1, there are three kinds of parts: I) Upper part of $LDL^T - \sigma I = L_+ D_+ L_+^T$, II) Lower part of $LDL^T - \sigma I = U_- D_- U_-^T$, and III) Twist index, respectively. This is because the MRRR algorithm is based on the Twisted Factorization[1] for the target tridiagonal matrix of T.

We can consider that the condensed bisection kernel is the lines ⟨2⟩–⟨7⟩ for the part I) in Figure 1, since the data dependency of Part II) is same as Part I), and the computation for Part III) is negligible. The kernel of Part I), hence, can be regarded as the kernel in bisection method in hereafter.

2.2 The Bisection Kernel

Figure 2 shows the kernel of bisection method for the target one eigenvalue.

```
⟨0⟩ $S = 0$; $NEG1 = 0$;
⟨1⟩ do $J = 1, R-1$
⟨2⟩   $T = S - \sigma$
⟨3⟩   $DPLUS = D(J) + T$
⟨4⟩   $S = T*LLD(J)\ /\ DPLUS$
⟨5⟩   if ( $DPLUS$ .lt. $ZERO$ )
            $NEG1 = NEG1 + 1$
⟨6⟩ enddo
```

Fig. 2 The Bisection Kernel.

The kernel of Figure 2 cannot be parallelized, since there is loop-carried flow-dependency for the variable of S [☆].

2.3 The Multi-section Kernel

Figure 3 shows the kernel of multi-section method with ML points for the target one eigenvalue.

The values of $\sigma(1:ML)$ in Figure 3 are calculated by $\sigma(i) = a + h \cdot i$, for $i = 1, 2, ..., ML$, where $h \equiv (b-a)/(ML+1)$, if the current interval is $[a,b]$.

The kernel of Figure 3 can be parallelized for the outer loop of I, since there is no dependency

[☆] There are two ways to be parallelized for the kernel (or the bisection method) of Figure 2. First method[9] is using parallelism for dividing the interval of the bisection. But the kernel of this method cannot be vectorized, since the method is using the kernel of Figure 2. The other method[10] is using tree based parallelism for the polynomias $p_k(x) = \det(T_k - xI)$ by parallel prefix method. However, this method has numerical instability problem.

```
⟨0⟩ $S(1:ML) = 0; NEG1(1:ML) = 0;$
⟨1⟩ do $I = 1, ML$
⟨2⟩   do $J = 1, R-1$
⟨3⟩     $T(I) = S(I) - \sigma(I)$
⟨4⟩     $DPLUS(I) = D(J) + T(I)$
⟨5⟩     $S(I) = T(I)*LLD(J) / DPLUS(I)$
⟨6⟩     if ( $DPLUS(I)$ .lt. $ZERO$ )
            $NEG1(I) = NEG1(I) + 1$
⟨7⟩   enddo
⟨8⟩ enddo
```

Fig. 3 The Multi-section Kernel.

for the all variables indicated I.

However, there is a problem for the multi-section kernel. If we want to take a large vector length for I to reduce parallel overhead, we should take a large number for the points of multi-section, which is the value of ML in Figure 3. But the efficiency of computation is getting worse according to the number of ML. This is caused by the additional overhead to apply the multi-section method. Hence, there is a trade-off between parallel execution efficiency and computation efficiency in this kernel.

The idea to solve this problem is: Computing multiple eigenvalues simultaneously with multi-section method. We call this method Multi-section with Multiple Eigenvalues (MME) method.

2.4 The MME Kernel

Figure 4 shows the kernel of MME.

```
⟨0⟩ $S(1:ML, 1:EL) = 0;$
    $NEG1(1:ML, 1:EL) = 0;$
⟨1⟩ do $K = 1, EL$
⟨2⟩   do $I = 1, ML$
⟨3⟩     do $J = 1, R-1$
⟨4⟩       $T(I,K) = S(I,K) - \sigma(I,K)$
⟨5⟩       $DPLUS(I,K) = D(J) + T(I,K)$
⟨6⟩       $S(I,K) = T(I,K)*LLD(J)$
                  $/ DPLUS(I,K)$
⟨7⟩       if ( $DPLUS(I,K)$ .lt. $ZERO$ )
            $NEG1(I,K) = NEG1(I,K) + 1$
⟨8⟩     enddo
⟨9⟩   enddo
⟨10⟩ enddo
```

Fig. 4 The Multi-section with Multiple Eigenvalues (MME) Kernel.

The values of σ ($1:ML$, $1:EL$) in Figure 4 are calculated by $\sigma(i,k) = a_k + h_k \cdot i$, for $i = 1, 2, ..., ML$, where $h_k \equiv (b_k - a_k)/(ML+1)$, if the current interval is $[a_k, b_k]$ for the k-th eigenvalue for $k = 1, 2, ..., EL$.

The loops of K and I in Figure 4 can be fused. Figure 5 shows the loop-fusion kernel.

```
⟨0⟩ $S(1:ML*EL) = 0;$
    $NEG1(1:ML*EL) = 0;$
⟨1⟩ do $I = 1, ML*EL$
⟨2⟩   do $J = 1, R-1$
⟨3⟩     $T(I) = S(I) - \sigma(I)$
⟨4⟩     $DPLUS(I) = D(J) + T(I)$
⟨5⟩     $S(I) = T(I)*LLD(J) / DPLUS(I)$
⟨6⟩     if ( $DPLUS(I)$ .lt. $ZERO$ )
            $NEG1(I) = NEG1(I) + 1$
⟨7⟩   enddo
⟨8⟩ enddo
```

Fig. 5 The loop-fusion MME Kernel.

The loop-fusion MME kernel in Figure 5 has the following merits to normal multi-section method.

First, with taking long loop-length for I, the parallel overhead can be reduced compared to normal multi-section method. This is because we can take the length EL times to the normal multi-section with ML. Hence, the ratio is EL times, which is the number of eigenvalues, to the normal multi-section method.

Second, although we take the small length for ML, the outer loop-length can keep long by setting the EL appropriately. This means that it can keep the computation efficiency high, since we should not take a long length for ML to obtain high parallelism.

There is a drawback for MME. Obviously, if there is no multiple eigenvalue in the routine of `dlarrb`, the merit of MME is same as the normal multi-section method.

2.5 Overall Process of MME

Figure 6 shows overall of MME method.

We do not know the number of eigenvalues in Figure 6 in advance even when we compute all eigenvalues for input matrix. This is because the usage of bisection routine strongly depends on the algorithm and the numerical characteristics in input matrices. In the case of MRRR algorithm, the number of eigenvalues is decided

```
⟨1⟩ if (#Eigenvalues .gt. 1) then
⟨2⟩   Call MME routine with
        $EL \equiv$ #Eigenvalues;
        $ML \equiv$ an appropriate value;
⟨3⟩ else
⟨4⟩   Call normal multi-section routine
        with $ML \equiv$ an appropriate value;
⟨5⟩ endif
```

Fig. 6 Overall of MME method.

by the Representation Tree based on the numerical characteristics for input matrices.

The optimal parameters for EL and ML, hence, cannot be found in advance. To obtain the best parameters, we need a run-time optimization for the parameters.

Figure 7 shows the detailed explanation for the MME method in LAPACK `dlarrb` routine.

There are three parts for the eigenvalue computation in Figure 7. They are I) Computation part of the intervals for $LEFT_j$ in ⟨3⟩–⟨9⟩, II) Computation part of the intervals for $RIGHT_j$ in ⟨10⟩–⟨16⟩, and III) Accuracy improvement part for the interval $[LEFT_j, RIGHT_j]$ in ⟨17⟩–⟨23⟩, for $j = J, J+1, ..., J+EL-1$ in Figure 7.

If the eigenvalues are very clustered, the part III in ⟨17⟩–⟨23⟩ will be heavy.

3. Performance Evaluation

3.1 Machine Environment

We used the HITACHI SR8000 (2Nodes/16PEs model.) The detailed information for this machine in this performance evaluation is shown:

- PE configuration: 8PEs / 1node.
- Job type: E8E, which is the mode for occupying 1 node of 8 PEs for the job.
- Compiler: HITACHI OFORT90 versioned V01-04-/B.
- Compiler Option: `-O4 -parallel=4`, which is the automatic parallelization option.
- Timer: `xclock`, which is a high accuracy timer provided by HITACHI.

3.2 Benchmarking Information

For benchmarking matrices, we used four kinds of matrix. The information is shown:

- Dimension: 2100
- Matrix#1: The (-1,2,-1) matrix.
- Matrix#2: A uniform random matrix generated from 0 to 1.

```
⟨1⟩  do $J = 1$, #Eigenvalues, $EL$
⟨2⟩    Make sure that $[LEFT_j, RIGHT_j]$
       for $j$-th eigenvalue ($j = J, ..., J+EL-1$);
  ⟨3⟩ I) Compute $NEGCNT_j$ from
          $L_+D_+L_+^T = LDL^T - LEFT_j$.
  ⟨4⟩ while (all intervals are enough small)
  ⟨5⟩   Set the points of $\sigma(1:EL)$ in
            the current interval of $LEFT_j$;
  ⟨6⟩   Call MME kernel with $EL$ and $ML$;
  ⟨7⟩   Fix the interval of $LEFT_j$ using
          returned numbers on $\sigma(1:EL)$
  ⟨8⟩   Check all intervals;
  ⟨9⟩ end while
  ⟨10⟩ II) Compute $NEGCNT_j$ from
          $L_+D_+L_+^T = LDL^T - RIGHT_j$.
  ⟨11⟩ while (all intervals are enough small)
  ⟨12⟩   Set the points of $\sigma(1:EL)$ in
            the current interval of $RIGHT_j$;
  ⟨13⟩  Call MME kernel with $EL$ and $ML$;
  ⟨14⟩   Fix the interval of $RIGHT_j$ using
         returned numbers on $\sigma(1:EL)$;
  ⟨15⟩   Check all intervals;
  ⟨16⟩ end while
  ⟨17⟩ III) There is unconverged interval.
  ⟨18⟩ while ((all intervals are enough
  small) .or. (#iteration .gt. $MAXITER$))
  ⟨19⟩   Set the points of $\sigma(1:EL)$ in
            the current interval of
            $[LEFT_j, RIGHT_j]$;
  ⟨20⟩  Call MME kernel with $EL$ and $ML$;
  ⟨21⟩   Fix the interval of $[LEFT_j, RIGHT_j]$
          using returned numbers on
          $\sigma(1:EL)$;
  ⟨22⟩   Check all intervals;
  ⟨23⟩ end while
⟨24⟩ enddo
⟨25⟩ if (#Eigenvalues is not divided by $EL$)
       Call multi-section routine with $ML$
       for the rest eigenvalue computations;
```

Fig. 7 Detailed Explanation for MME method. This is also detailed explanation for the `dlarrb` routine with MME method.

- Matrix#3: The Wilkinson Matrix W_{2100}^{+}
- Matrix#4: The "Rotated Subdiagonal" Glued Wilkinson Matrix W_{21}^{+}. This matrix is defined as: $Diag(T) = Diag($ Glued Wilkinson Matrix $W_{21}^{+})$. $Sub(T)$ is composed of 100 times of $(\delta, 1, .., 1)_{21}$. The glue value δ is $1e-1$.

Table 1 shows the distribution for the number of eigenvalues in the `dlarrb` routine for each benchmarking matrices.

Table 1 The Distribution for The Number of Eigenvalues in The `dlarrb` Routine with DQDS Mode.

(a) Matrix #1

#Eigenvalues	Frequency	% to Total Frequency
1	22	95.6
789	1	4.34

(b) Matrix #2

#Eigenvalues	Frequency	% to Total Frequency
1	850	68.4
2	145	11.6
3	78	6.28
4	45	3.62
5	31	2.49
6	21	1.69
7	15	1.20
8	11	0.88
9	7	0.56
10	9	0.72
11	5	0.40
12	4	0.32
13	1	0.08
14	4	0.32
15	3	0.24
16	2	0.16
17	1	0.08
18	3	0.24
19	1	0.08
20	1	0.08
21	1	0.08
22	1	0.08
24	1	0.08
35	1	0.08

(c) Matrix #3

#Eigenvalues	Frequency	% to Total Frequency
1	2093	66.7
2	1043	33.2
102	1	0.31

(d) Matrix #4

#Eigenvalues	Frequency	% to Total Frequency
1	235	82.7
2	27	9.50
3	1	0.35
99	7	2.46
100	12	4.22
200	2	0.70

For the target application, we used DSTEGR routine in LAPACK version 4.0. The information for the target application is summarized:

- Routine: The MME method was implemented in the bisection routine of DSTEGR, which is `dlarrb` routine.
- Computation: All eigenvalues and all eigenvectors.
- Process: There are two modes for the eigenvalue computation in DSTEGR. They are DQDS mode (Using LAPACK's `DLASQ1`[11]) and aggressive bisection mode.
  - The DQDS mode is using DQDS method in eigenvalue calculation part. The mode is using the bisection in eigenvector calculation part to fix the eigenvalue accuracy.
  - The aggressive bisection mode is using the bisection in both of the parts.

  The two modes were used in this evaluation.
- Object: Total execution time for the bisection routine `dlarrb` using MME method was measured.
- Static Fixing Method for The Parameter: The method using a constant parameter for EL and ML while the xSTEGR routing is running is defined as "Static Fixing Method." The execution time for the method was checked with:
  - $EL \in [1, 2, 3, 4, 8, 16, 32]$: 7 kinds,
  - $ML \in [1, 2, 4, 8, 16, 24]$: 6 kinds.

  Hence, the best parameter for $EL \cdot ML =$ 42 kinds of combinations was specified.

3.3 Effect of MME with Static Parameter Fixing Method to Multi-section Method

Table 2 shows the effect of MME with the static fixing method to multi-section method.

For Table 2, we obtained maximum 7.7x speedup to bisection method, and maximum 4.3x speedup to multi-section method by using MME method.

For Matrices #1 and #4, MME with 16 eigenvalues was selected. This is because there are the calls with 789 or 200 eigenvalues in the `dlarrb` in Table 1, which are very heavy executions.

On the other hand, MME method with 2 eigenvalues was selected in Matrices #2 and #3. This is because there are many calls with low number of eigenvalues compared to Matrices #1 and #4. For example, the execution is done with 35 eigenvalues or 102 eigenvalues for one time at most in Table 1.

4. Conclusion

In this paper, we propose a new implementation method for computing eigenvalues in symmetric tridiagonal eigensolvers. The key idea for this method is using natural parallelism

Table 2 Effect of MME Method with Static Fixing for the parameter to Multi-section Method.

(a) Total Execution Time for `dlarrb` Routine in The DQDS Mode.

Method / #Matrix	#1	#2	#3	#4
Bisection [s]	0.347	1.83	15.2	8.20
Multi-section [s]	0.323	1.45	5.90	3.30
The best *ML*	8	8	16	16
MME [s]	0.075	1.16	5.34	1.75
The best (*EL*,*ML*)	(16,2)	(2,8)	(2,8)	(16,1)
Sp. to Bisection	4.6x	1.5x	2.8x	4.6x
Sp. to Multi-section	4.3x	1.2x	1.1x	1.8x

(b) Total Execution Time for `dlarrb` Routine in The Aggressive Bisection Mode.

Method / #Matrix	#1	#2	#3	#4
Bisection [s]	3.94	7.11	23.9	17.2
Multi-section [s]	2.11	3.69	8.70	6.19
The best *ML*	16	16	16	16
MME [s]	0.51	2.93	7.89	2.75
The best (*EL*,*ML*)	(16,1)	(2,8)	(2,8)	(16,1)
Sp. to Bisection	7.7x	2.4x	3.0x	6.2x
Sp. to Multi-section	4.1x	1.2x	1.1x	2.2x

for multiple eigenvalues and multi-section with the interval for the target one eigenvalue. We named the method "Multi-section with Multiple Eigenvalues (MME)."

The static fixing method for MME, which is defined in this paper, is not optimal. This is because the number of eigenvalues changes in each `dlarrb` call. We proposed a run-time optimization method called "dynamic parameter fixing method" in 12). The performance evaluation results indicated that it was more effective than the static one.

The parallel effect of MME seems to be increased, if the number of processors is increasing. Hence, the most important future work is to evaluate the effectiveness with the other parallel machines. The performance evaluation with the other benchmark applications using the bisection method will be one of candidates for future work.

On the other hand, the difficulty to apply numerical libraries to heterogeneous computing environment is known[13]. Appling MME method for such environment, including GRID environment, will be challenging future work.

References

1) Parlett, B. N. and Dhillon, I. S.: Fernando's Solution to Wilkinson's Problem: An Application of Double Factorization, *Linear Algebra and Appl.*, Vol. 267, pp. 247–279 (1997).

2) Parlett, B. N. and Dhillon, I. S.: Relatively Robust Representations of Symmetric Tridiagonals, *Linear Algebra and Appl.*, Vol. 309, pp. 121–151 (2000).

3) Parlett, B. N. and Dhillon, I. S.: Multiple Representations to Compute Orthogonal Eigenvectors of Symmetric Tridiagonal Matrices, *Linear Algebra and Appl.*, Vol. 387, pp. 1–28 (2004).

4) Parlett, B. N. and Dhillon, I. S.: Orthogonal Eigenvectors and Relative Gaps, *SIAM J. Matrix Anal. Appl.*, Vol. 25, No. 3, pp. 858–899 (2004).

5) Dhillon, I.S., Parlett, B.N. and Vömel, C.: LAPACK Working Note 162: The Design and Implementation of the MRRR Algorithm, Technical Report UCBCSD-04-1346, University of California, Berkeley (2004).

6) Lo, S.-S., Philippe, B. and Sameh, A.: A Multiprocessor Algorithm for the Symmetric Tridiagonal Eigenvalue Problem, *SIAM J. Sci. Stat. Comput.*, Vol. 8, No. 2, pp. s155–s165 (1987).

7) Simon, H. D.: Bisection Is Not Optimal on vector Processors, *SIAM J. Sci. Stat. Comput.*, Vol. 10, No. 1, pp. 205–209 (1989).

8) Marques, O. A., Riedy, J. and Vömel, C.: Benefits of IEEE-754 features in Modern Symmetric Tridiagonal Eigensolvers, Technical Report UCBCSD-05-1414, University of California, Berkeley (2005).

9) Demmel, J. W., Dhillon, I. and Ren, H.: On the Correctness of Parallel Bisection in Floating Point, Technical Report UCB//CSD-94-805, University of California, Berkeley (1994).

10) Ren, H.: On the Error Analysis and Implementation of Some Eigenvalue Decomposition and Singular Value Decomposition Algorithms, Technical Report UT-CS-96-336, University of California, Berkeley (1996).

11) Parlett, B. N. and Marques, O.: An Implementation of The DQDS Algorithm (Positive Case), *Linear Algebra and Appl.*, Vol. 309, pp. 217–259 (2000).

12) Katagiri, T., Vömel, C. and Demmel, J. W.: Multi-section with Multiple Eigenvalues Method for Computing Eigenvalues in Symmetric Tridiagonal Eigensolvers (2005). Unpublished paper.

13) Blackford, L. S., Cleary, A., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Petitet, A., Ren, H., Stanley, K. and Whaley, R. C.: Practical Experience in the Dangers of Heterogeneous Computing, Technical Report UT-CS-96-330, University of California, Berkeley (1996).