Comparison of Tridiagonalization Methods using High-precision Arithmetic with MuPAT

Ryoya Ino, Kohei Asami, Emiko Ishiwata, and Hidehiko Hasegawa

Abstract In general, when computing the eigenvalues of symmetric matrices, a matrix is tridiagonalized using some orthogonal transformation. The Householder transformation, which is a tridiagonalization method, is accurate and stable for dense matrices, but is not applicable to sparse matrices because of the required memory space. The Lanczos and Arnoldi methods are also used for tridiagonalization and are applicable to sparse matrices, but these methods are sensitive to computational errors. In order to obtain a stable algorithm, it is necessary to apply numerous techniques to the original algorithm, or to simply use accurate arithmetic in the original algorithm. In floating-point arithmetic, computation errors are unavoidable, but can be reduced by using high-precision arithmetic, such as double-double (DD) arithmetic or quad-double (QD) arithmetic. In the present study, we compare double, double-double, and quad-double arithmetic for three tridiagonalization methods; the Householder method, the Lanczos method, and the Arnoldi method. To evaluate the robustness of these methods, we applied them to dense matrices that are appropriate for the Householder method. It was found that using high-precision arithmetic, the Arnoldi method can produce good tridiagonal matrices for some problems whereas the Lanczos method cannot.

1 Introduction

Recently, eigenvalue computation has become very important in several applications. For a real symmetric dense matrix, the target matrix is usually reduced to

Ryoya Ino, Kohei Asami, and Emiko Ishiwata
Tokyo University of Science, 1-3 Kagurazaka, Shinjuku-ku, Tokyo, Japan
e-mail: ishiwata@rs.kagu.tus.ac.jp

Hidehiko Hasegawa
University of Tsukuba, Kasuga 1-2, Tsukuba, Japan
e-mail: hasegawa@slis.tsukuba.ac.jp

symmetric tridiagonal form by orthogonal similarity transformations, and the eigenvalues of the obtained symmetric tridiagonal matrix are then computed by, for example, the QR method or bisection and inverse iteration algorithms. On the other hand, for sparse matrices other than band matrices, tridiagonalization by the Householder transformation is so difficult because of requiring a great deal of memory. The Lanczos method involves simple matrix-vector multiplication and vector operations, and does not require modification of the given matrix. The Lanczos and Arnoldi methods are simple algorithms, but the roundoff error causes the Lanczos vectors to lose orthogonality [1]. However, they may require less memory.

Mathematically simple algorithms are often unstable because of computation errors. In order to obtain a stable algorithm, we can apply several techniques to the original algorithm, or simply use accurate arithmetic. In floating-point arithmetic, computation errors are unavoidable, but can be reduced through the use of high-precision arithmetic, such as double-double (DD) arithmetic or quad-double (QD) arithmetic.

Kikkawa et al. and Saito et al.[2, 3] developed the Multiple Precision Arithmetic Toolbox (MuPAT), a high-precision arithmetic software package, on Scilab[4]. The MuPAT uses double-double arithmetic and quad-double arithmetic in order to work on conventional computers. The computation time for double-double-precision arithmetic is approximately 20 times greater than that for ordinary double-precision arithmetic, but this cost can be reduced through the use of parallel processing.

In the present paper, we compare double, double-double, and quad-double arithmetic for the Lanczos method, the Arnoldi method, and the Householder method [1] for obtaining symmetric tridiagonal matrices from symmetric matrices, and the QR method for finding all eigenvalues thereof. We use a sparse storage format of MuPAT in order to reduce the memory requirement, but did not use parallel processing.

2 Multiple-precision arithmetic on MuPAT

2.1 Double-double and quad-double arithmetic

Double-double and quad-double arithmetic were proposed as quasi-quadruple-precision and quasi-octuple-precision arithmetic by Hida et al. [5] and Dekker [6]. A double-double number is represented by two double-precision numbers, and a quad-double number is represented by four double-precision numbers. A double number $x_{(D)}$, a double-double number $x_{(DD)}$ and a quad-double number $x_{(QD)}$ are represented by an unevaluated sum of double-precision numbers x_0, x_1, x_2, x_3 as follows:

$$x_{(D)} = x_0, \quad x_{(DD)} = x_0 + x_1, \quad x_{(QD)} = x_0 + x_1 + x_2 + x_3,$$

where x_0, x_1, x_2 and x_3 satisfy the following inequalities:

$$|x_{i+1}| \leq \frac{1}{2}\mathrm{ulp}(x_i), \quad i = 0,1,2,$$

where ulp stands for 'units in the last place'. For a given decimal input data x, we can also denote that

$$x_{(D)} = (x_0)_{(D)}, \quad x_{(DD)} = (x_0, x_1)_{(DD)}, \quad x_{(QD)} = (x_0, x_1, x_2, x_3)_{(QD)}.$$

The lower portion is ignored or truncated from the longer format data to the shorter format data, and is assumed to be zeros from the shorter format data to the longer format data. A double-double (quad-double) number has 31 (63) significant decimal digits.

In this paper, we abbreviate double-double and quad-double on DD and QD. Both DD and QD arithmetic are performed using error-free floating point arithmetic algorithms that use only double-precision arithmetic and so require only double-precision arithmetic operations. Both DD and QD arithmetic are described in detail in [5] and [6]. Table 1 shows the number of double-precision arithmetic operations for DD and QD arithmetic.

Table 1 Number of double-precision arithmetic operations

type		add & sub	mul	div	total
DD	add& sub	11	0	0	11
	mul	15	9	0	24
	div	17	8	2	27
QD	add& sub	91	0	0	91
	mul	171	46	0	217
	div	579	66	0	649

2.2 Extended MuPAT with a sparse data structure

A quadruple- and octuple-precision arithmetic toolbox, i.e., the Multiple Precision Arithmetic Toolbox (MuPAT) and variants thereof [2, 3], allow the use of double-, quadruple-, and octuple-precision arithmetic with the same operators or functions, and mixed-precision arithmetic and partial use of different precision arithmetic becomes possible. The MuPAT is independent of hardware and operating system.

We developed an accelerated MuPAT for sparse matrices in [3] in order to reduce the amount of memory and computation time, and using the developed MuPAT, large matrices can easily be handled. We define two data types for a sparse matrix: `DDSP` for double-double numbers and `QDSP` for quad-double numbers.

These data types are based on the compressed column storage (CCS) format, which contains vectors in the form of row indices, column pointers, and values.

Note that DDSP uses two value vectors and QDSP uses four value vectors to represent double-double and quad-double numbers, respectively. As such, it is possible to use a combination of double, double-double, and quad-double arithmetic for both dense and sparse data structures. Based on the definitions of these data types, MuPAT has six data types: `constant`, DD, and QD for dense data, and `sparse`, DDSP, and QDSP for sparse data of double, double-double and quad-double numbers, respectively.

Quad-double arithmetic requires a tremendous number of double-precision operations. In particular, one QD division requires 649 double-precision operations, so the required computation time is hundreds of times greater than that for double-precision arithmetic on Scilab. In order to accelerate QD and DD arithmetic operations, external routines written in the C language are prepared. These MuPAT functions achieve high-speed processing but depend on the hardware and operating system used. Currently, this code is not parallelized but can be accelerated through the use of parallel processing.

3 Eigenvalue computation

In order to compute the eigenvalues of a real symmetric matrix A, the matrix A is usually tridiagonalized to an similarity tridiagonal matrix T by similarity transformations, and the eigenvalues of the matrix T are then computed. The Lanczos, Arnoldi, and Householder methods can be used for this purpose.

The Lanczos and Arnoldi methods involve matrix-vector multiplication and some vector operations. Since, unlike in the Householder method, updating the original matrix A is not necessary, the Lanczos and Arnoldi methods can be easily applied to sparse matrices.

The QR method and the bisection algorithm are used for computing the eigenvalues of a tridiagonal matrix T. For computing the eigenvectors of T, the QR method and inverse iteration are used. The quality of eigenvalues and eigenvectors depends only on the tridiagonal matrix T and not on the tridiagonalization method. If T is an inexact approximation of A, even if the eigenvalues and eigenvectors of T are correctly calculated, they do not correspond to those of A.

In the present paper, we used the implicit single shift QR algorithm based on [7] for computing all eigenvalues. The QR method generates eigenvalues as diagonal elements in descending order.

In particular, for a sparse matrix, the transformations used for tridiagonalizing A to T are not used to compute eigenvectors of A, which would require tremendous computation and memory. If eigenvalues λ_T of T are accurately computed, an inverse iteration method can be applied to compute the eigenvectors of $(T-\lambda_T I)$ or $(A-\lambda_T I)$. The inverse iteration method for sparse matrices uses a direct solver or an iterative solver, such as the conjugate gradient method.

3.1 Tridiagonalization

For a given symmetric matrix A, it is possible to find an orthogonal Q such that $Q^T AQ = T$ is tridiagonal. In the present paper, we consider three tridiagonalization methods for symmetric matrices: the Lanczos method, the Arnoldi method, and the Householder method. These methods are described in detail in [1].

3.1.1 Lanczos method

The Lanczos method can construct an equivalent tridiagonal matrix by generating orthogonal bases one after another. However, roundoff errors cause the Lanczos vectors to lose orthogonality [1].

Let A be an $n \times n$ symmetric matrix, and let Q be an $n \times n$ orthogonal matrix. Then, we generate $T = Q^T AQ$. We set the column of Q by

$$Q = [\boldsymbol{q}_1 | \boldsymbol{q}_2 | \cdots | \boldsymbol{q}_n]$$

and the components of T by

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{n-1} \\ 0 & \cdots & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Equating columns as $AQ = QT$, we conclude that

$$A\boldsymbol{q}_k = \beta_{k-1}\boldsymbol{q}_{k-1} + \alpha_k \boldsymbol{q}_k + \beta_k \boldsymbol{q}_{k+1} \qquad (\beta_0 \boldsymbol{q}_0 \equiv \boldsymbol{0}),$$

for $k = 1, 2, ..., n-1$. The orthonormality of the vector $\boldsymbol{q}_k$ implies

$$\alpha_k = \boldsymbol{q}_k^T A \boldsymbol{q}_k.$$

If we define the vector $\boldsymbol{r}_k$ as

$$\boldsymbol{r}_k = (A - \alpha_k I)\boldsymbol{q}_k - \beta_{k-1}\boldsymbol{q}_{k-1},$$

and if it is nonzero, then

$$\boldsymbol{q}_{k+1} = \frac{\boldsymbol{r}_k}{\beta_k},$$

where $\beta_k = \pm \|\boldsymbol{r}_k\|_2$.

For a given symmetric matrix $A \in R^{n \times n}$ and an initial vector $\boldsymbol{q}_0 \in R^n$, Algorithm 1 computes a matrix $Q = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_n]$ with orthonormal columns and a tridiagonal matrix $T \in R^{n \times n}$ so that $AQ = QT$. The diagonal and superdiagonal entries of T are $\alpha_1, \cdots, \alpha_n$ and $\beta_1, \cdots, \beta_{n-1}$, respectively.

Algorithm 1 The Lanczos method [1]

1: $k = 0, \boldsymbol{r}_0 = \boldsymbol{q}_0, \beta_0 = \|\boldsymbol{q}_0\|_2$
2: **while** $\beta_k \neq 0$ **do**
3: $\quad \boldsymbol{q}_{k+1} = \frac{\boldsymbol{r}_k}{\beta_k}$
4: $\quad k = k+1$
5: $\quad \alpha_k = \boldsymbol{q}_k^T A \boldsymbol{q}_k$
6: $\quad \boldsymbol{r}_k = (A - \alpha_k I)\boldsymbol{q}_k - \beta_{k-1}\boldsymbol{q}_{k-1}$
7: $\quad \beta_k = \|\boldsymbol{r}_k\|_2$
8: **end while**

3.1.2 Arnoldi method

The Arnoldi method is a way to extend the Lanczos method to non-symmetric matrices and generate the Hessenberg matrix $Q^T AQ = H$. However, for a symmetric matrix A, this process produces a tridiagonal matrix $T = H$.

In the same manner as the Lanczos iteration, we set $Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_n]$ and compare columns in $AQ = QH$. Then,

$$A\boldsymbol{q}_k = \sum_{i=1}^{k+1} h_{ik}\boldsymbol{q}_i, \qquad 1 \leq k \leq n-1.$$

Isolating the last term in the summation gives

$$\boldsymbol{r}_k \equiv A\boldsymbol{q}_k - \sum_{i=1}^{k} h_{ik}\boldsymbol{q}_i,$$

where $h_{ik} = \boldsymbol{q}_i^T A\boldsymbol{q}_k$ for $i = 1, 2, ..., k$. It follows that if $\boldsymbol{r}_k \neq \boldsymbol{0}$, then $\boldsymbol{q}_{k+1}$ is specified by

$$\boldsymbol{q}_{k+1} = \frac{\boldsymbol{r}_k}{h_{k+1,k}},$$

where $h_{k+1,k} = \|\boldsymbol{r}_k\|_2$. These equations define the Arnoldi method.

For a given matrix $A \in R^{n\times n}$ and an initial vector $\boldsymbol{q}_0 \in R^n$, Algorithm 2 computes a matrix $Q = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_n] \in R^{n\times n}$ with orthonormal columns and an upper Hessenberg matrix $H \in R^{n\times n}$ so that $AQ = QH$. Especially for a symmetric matrix, this algorithm generates an orthogonal matrix Q and a tridiagonal matrix T.

Algorithm 2 The Arnoldi method [1]

$k = 0, \boldsymbol{r}_0 = \boldsymbol{q}_0, h_{1,0} = \|\boldsymbol{q}_0\|_2$
while $h_{k+1,k} \neq 0$ **do**
  $\boldsymbol{q}_{k+1} = \frac{\boldsymbol{r}_k}{h_{k+1,k}}$
  $k = k+1$
  $\boldsymbol{r}_k = A\boldsymbol{q}_k$
  for $i = 1, 2, \cdots, k$ **do**
    $h_{ik} = \boldsymbol{q}_i^T \boldsymbol{r}_k$
    $\boldsymbol{r}_k = \boldsymbol{r}_k - h_{ik}\boldsymbol{q}_i$
  end for
  $h_{k+1,k} = \|\boldsymbol{r}_k\|_2$
end while

3.1.3 The Householder method

The Householder method for a symmetric matrix can generate an tridiagonal matrix $Q^T AQ = T$ using the Householder matrix [1]. Suppose that the Householder matrices $P_1, \cdots, P_{k-1}$ have been determined such that if

$$A_{k-1} = (P_1 \cdots P_{k-1})^T A (P_1 \cdots P_{k-1}),$$

then

$$A_{k-1} = \begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & B_{23} \\ 0 & B_{32} & B_{22} \end{bmatrix}$$

is tridiagonal through its first $k-1$ columns. If $\tilde{P}_k$ is an order-$(n-k)$ Householder matrix such that $\tilde{P}_k B_{32}$ is a multiple of I_{n-1} and if $P_k = diag(I_k, \tilde{P}_k)$, then the leading k-by-k principal submatrix of

$$A_k = P_k A_{k-1} P_k = \begin{bmatrix} B_{11} & B_{12} & 0 \\ B_{21} & B_{22} & B_{23}\tilde{P}_k \\ 0 & \tilde{P}_k B_{32} & \tilde{P}_k B_{33} \tilde{P}_k \end{bmatrix}$$

is tridiagonal. Clearly, if $U = P_1 \cdots P_{n-2}$, then $U^T AU = T$ is tridiagonal. In the calculation of A_k, it is important to exploit symmetry during the formation of the matrix $\tilde{P}_k B_{33} \tilde{P}_k$. More specifically, suppose that $\tilde{P}_k$ has the form

$$\tilde{P}_k = I - \beta \boldsymbol{v}\boldsymbol{v}^T, \qquad \beta = \frac{2}{\boldsymbol{v}^T \boldsymbol{v}}, \qquad 0 \neq \boldsymbol{v} \in R^{n-k}.$$

Note that if $\boldsymbol{p} = \beta B_{33} \boldsymbol{v}$ and $\boldsymbol{w} = \boldsymbol{p} - (\frac{\beta \boldsymbol{p}^T \boldsymbol{v}}{2})\boldsymbol{v}$, then

$$\tilde{P}_k B_{33} \tilde{P}_k = B_{33} - \boldsymbol{v}\boldsymbol{w}^T - \boldsymbol{w}\boldsymbol{v}^T.$$

We used the Householder algorithm written in [1].

Since only the upper triangular portion of this matrix needs to be calculated, we see that the transition from A_{k-1} to A_k can be accomplished in only $4(n-k)^2$ flops for a dense matrix.

4 Numerical experiments

In this section, we analyze the accuracy, numerical stability, and computing cost for three tridiagonalization methods and the computed eigenvalue by the implicit single shift QR algorithm [7] for the tridiagonal matrix T. For tridiagonalization, we compare three arithmetic precisions: double (D), DD, and QD.

The QR method can be applied to non-symmetric matrices (not only tridiagonal matrices), in which case complex eigenvalues would appear. Therefore, we use only tridiagonal factors in the Arnoldi method.

For the Lanczos and Arnoldi methods, the initial vector $\boldsymbol{q}_0$ is a uniformly distributed random vector between 0 and 1 using the 'rand' function of Scilab.

All experiments were carried out on an Intel Core i5-4200U, 1.60 GHz, 8GB memory and Scilab 5.5.0 on Windows 7 Professional. We assumed the 'true eigenvalue' to be the computation result produced by the 'Eigenvalues' function of Mathematica with 200 decimal digits.

4.1 Example 1: nos4 (small problem)

We demonstrate the results of the three tridiagonalization methods for a small matrix 'nos4' in MatrixMarket [8]. The dimension of this matrix was 100, the number of the nonzero elements was 594, the condition number on the matrix was 2.7×10^3, and the matrix originated from a structure problem. The eigenvalues of nos4 are distributed between 0.00053795... and 0.84913778... without any clustered eigenvalues.

Table 2 lists the accuracy of the eigenvalues, the loss of orthogonality, and the computation times for the three tridiagonalization methods and three precisions. Here, $max|\lambda_i - \bar{\lambda}_i|$ and $avg|\lambda_i - \bar{\lambda}_i|$ denote the maximum absolute error and the average of absolute errors, where λ_i and $\bar{\lambda}_i$ represent the ith computed eigenvalue and the true eigenvalue, respectively. We checked the loss of orthogonality by $\frac{\|Q^TQ-I\|_F}{\|I\|_F}$, where I and Q are a unit matrix and an orthogonal matrix, respectively, and $\|\cdot\|_F$ denotes the Frobenius norm. '*Avg* time' for dense and sparse implies the computation time only for tridiagonalization part.

For the QR algorithm, the accuracy of the eigenvalues in D, DD, and QD are approximately the same for all tridiagonalization methods. This means that the accuracy of the QR method with double-precision arithmetic is sufficient, and the accuracy of tridiagonalization is important in eigenvalue computation. Therefore, we hereinafter apply the QR method with only double-precision arithmetic and focus on the difference in accuracy and computation time among the tridiagonalization methods and their arithmetic precisions.

Table 2 Accuracy and tridiagonalization time [s] for nos4

Tridiagonalization	QR	$max\|\lambda_i - \bar{\lambda}_i\|$	$avg\|\lambda_i - \bar{\lambda}_i\|$	$avg\ \frac{\|Q^TQ-I\|_F}{\|I\|_F}$	*avg* time dense	*avg* time sparse
D	D	1.43×10^{-1}	4.56×10^{-2}	4.21×10^{-1}	0.016	0.006
	DD	1.43×10^{-1}	4.56×10^{-2}			
	QD	1.43×10^{-1}	4.56×10^{-2}			
DD	D	7.96×10^{-2}	1.91×10^{-2}	2.45×10^{-1}	0.113	0.095
	DD	7.96×10^{-2}	1.91×10^{-2}			
	QD	7.96×10^{-2}	1.91×10^{-2}			
QD	D	2.00×10^{-15}	2.31×10^{-16}	9.65×10^{-21}	0.554	0.157
	DD	1.97×10^{-16}	3.86×10^{-17}			
	QD	1.97×10^{-16}	3.86×10^{-17}			
D	D	1.40×10^{-2}	1.15×10^{-3}	1.41×10^{-1}	0.031	0.028
	DD	1.40×10^{-2}	1.15×10^{-3}			
	QD	1.40×10^{-2}	1.15×10^{-3}			
DD	D	1.55×10^{-15}	1.82×10^{-16}	7.25×10^{-13}	0.938	0.903
	DD	4.94×10^{-16}	5.18×10^{-16}			
	QD	4.94×10^{-16}	5.18×10^{-17}			
QD	D	2.25×10^{-15}	2.75×10^{-16}	6.17×10^{-46}	2.172	1.744
	DD	3.50×10^{-16}	3.88×10^{-17}			
	QD	3.50×10^{-16}	3.88×10^{-17}			
D	D	1.44×10^{-15}	2.53×10^{-16}	5.99×10^{-15}	0.038	-
	DD	4.54×10^{-16}	1.04×10^{-16}			
	QD	4.54×10^{-16}	1.04×10^{-16}			
DD	D	9.99×10^{-16}	1.90×10^{-16}	5.49×10^{-31}	5.054	-
	DD	2.87×10^{-16}	4.85×10^{-17}			
	QD	2.87×10^{-16}	4.85×10^{-17}			
QD	D	9.99×10^{-16}	1.90×10^{-16}	1.61×10^{-63}	41.697	-
	DD	2.87×10^{-16}	4.85×10^{-17}			
	QD	2.87×10^{-16}	4.85×10^{-17}			

Concerning the tridiagonalization methods, there is little difference between the maximum and average errors for Lanczos-QD, Arnoldi-DD, -QD, and Householder-D, -DD, -QD (where, for example, Lanczos-QD indicates the Lanczos method with QD precision).

The orthogonalities of Lanczos-QD, Arnoldi-DD, and Householder-D are approximately the same and can be improved by using DD and QD. The relationship between method and accuracy depends on the given matrix. In the case of nos4, however, Householder-D, Arnoldi-DD and Lanczos-QD are sufficient.

In Fig. 1, the horizontal axis indicates the index of the eigenvalues in descending order, and the vertical axis indicates the absolute error of eigenvalues $|\lambda_i - \bar{\lambda}_i|$. Fig. 1 shows that the absolute errors of Lanczos-D and -DD are large in general but become small for smaller eigenvalues, the absolute error of Arnoldi-D increases for smaller eigenvalues (from approximately the 50th eigenvalue), and Lanczos-QD, Arnoldi-DD, -QD, and Householder-D provide sufficient accuracy.

In Fig. 2, the horizontal axis again indicates the index of the eigenvalues in descending order, and the vertical axis indicates the value of the computed eigenvalues. The results for 'Mathematica', which represents the true eigenvalues, Lanczos-QD, Arnoldi-DD, and Householder-D are approximately the same. Both Lanczos-D and -DD have duplicative eigenvalues. Using higher-precision arithmetic, a plot is gradually brought closer to the true eigenvalue.

Table 3 Elements outside the tridiagonal part for the Arnoldi method for nos4

	D	DD	QD
$10^{-5} \leq x$	598	0	0
$10^{-10} \leq x < 10^{-5}$	1224	0	0
$10^{-15} \leq x < 10^{-10}$	2716	103	0
$10^{-20} \leq x < 10^{-15}$	313	453	0
$10^{-30} \leq x < 10^{-20}$	0	3171	0
$10^{-40} \leq x < 10^{-30}$	0	1124	0
$10^{-50} \leq x < 10^{-40}$	0	0	263
$x < 10^{-50}$	0	0	4588
maximum	1.83×10^{-1}	3.97×10^{-12}	7.42×10^{-45}

Table 3 lists the numbers of elements outside the tridiagonal part (upper triangular) for the Arnoldi method. The Arnoldi method is based on similarity transformation of non-symmetric matrices to Hessenberg matrices, and elements outside the tridiagonal part should be zero in the case of symmetric matrices. However, in our numerical experiments, nonzero elements appeared outside the tridiagonal part because of rounding errors. The relationship between nonzero elements and the accuracy of tridiagonalization is an area for future study.

In the case of using dense data, the ratio of the computation time for the Lanczos, Arnoldi, and Householder methods with double-precision arithmetic is approximately 1:2:2. For DD, the number of double-precision computations is 7 for the Lanczos method, 30 for the Arnoldi method, and 133 for the Householder method. For QD, the number of double-precision computations is 35 for the Lanczos method, 70 for the Arnoldi method, and 1,100 for the Householder method.

The computation times for Lanczos-DD and Lanczos-QD for sparse data are 84% and 28%, respectively, of those for dense data, and the computation times for Arnoldi-DD and Arnoldi-QD for sparse data are 96% and 80%, respectively, of those for dense data. Costs of high-precision arithmetic and saving of computation time in sparse data type are depending on used algorithms and their implemen-

tations. For small matrices, Householder-D is the best method, but the computation time for Lanczos-QD using sparse data is only four times greater than that for Householder-D.

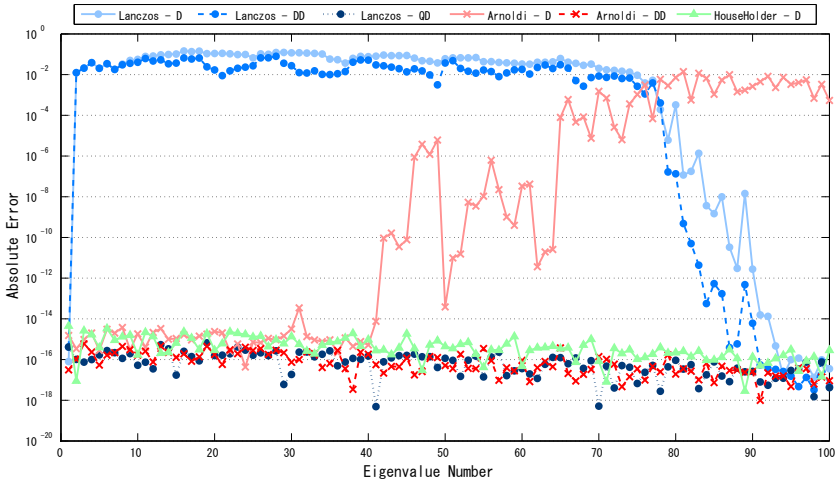

Fig. 1 Absolute error in eigenvalues for nos4 in descending order

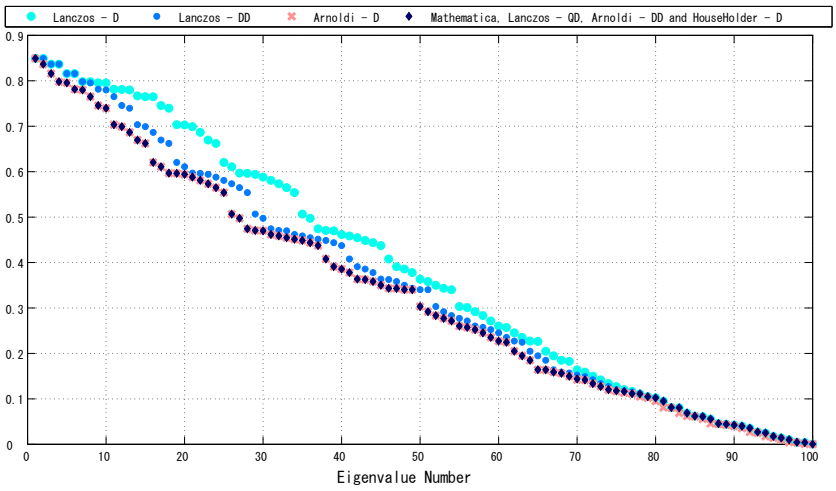

Fig. 2 Eigenvalues for nos4 in descending order

4.2 Example 2: Trefethen_200b (medium problem)

We used a larger test matrix 'Trefethen_200b' from the University of Florida Sparse Matrix Collection [9]. The dimension of this matrix was 199, the number of nonzero elements was $2,873$, the condition number was 5.2×10^2, and the matrix originated from a combinatorial problem. The eigenvalues of Trefethen_200b' are distributed between 2.3443911... and 1223.3718... without any clustered eigenvalues.

Table 4 shows the results for various combinations of methods and precisions. As mentioned in Section 4.1, we applied the QR method with only double-precision arithmetic.

Concerning the accuracy of the eigenvalues, Lanczos-QD and Arnoldi-DD are not improved, but Arnoldi-QD and Householder-D, -DD, -QD are sufficient. In terms of orthogonality, the accuracy of Arnoldi-QD and Householder-D is approximately the same.

In the case of using dense data, the ratio of the computation times for the Lanczos, Arnoldi, and Householder methods with double-precision arithmetic is approximately 1:14:30. For DD, the number of double-precision computation is 114 for the Lanczos method, 52 for the Arnoldi method, and 244 for the Householder method. For QD, the number of double-precision computations is 634 for the Lanczos method, 147 for the Arnoldi method, and 2,314 for the Householder method.

The computation times for Lanczos-DD and for Lanczos-QD for sparse data are 26% and 13%, respectively, of those for dense data, and the computation times for Arnoldi-DD and Arnoldi-QD are both 80% of those for dense data. For Trefethen_200b, the computation time was greatly reduced by the use of sparse data of MuPAT.

In Fig. 3, the horizontal and vertical axes are the same as in Fig. 1. Fig. 3 reveals the following: The absolute errors for Lanczos-QD are large, but become small for smaller eigenvalues from approximately half of dimension. In contrast, the absolute error for Arnoldi-DD increases for the smaller eigenvalues (from approximately the 80th eigenvalue). Arnoldi-QD and Householder-D are sufficient.

Table 5 shows the upper triangular factors outside the tridiagonal part, which should be zero using the Arnoldi method. For Arnoldi-D and -DD, there are numerous nonzero elements within the range of double precision, and these elements affect the accuracy of the eigenvalues. For Arnoldi-QD, the number of nonzero elements is sufficiently small and does not affect the accuracy of eigenvalues in double precision.

Table 4 Accuracy and tridiagonalization time [s] and memory space [KB] for Trefethen_200b

Tridiagonal -ization		$max\|\lambda_i - \overline{\lambda}_i\|$	$avg\|\lambda_i - \overline{\lambda}_i\|$	$\frac{\|Q^T Q - I\|_F}{\|I\|_F}$	dense time	dense space	sparse time	sparse space
Lanczos	D	1.10×10^{2}	3.44×10^{1}	6.26×10^{4}	0.01	77.95	0.02	40.26
	DD	7.94×10^{1}	2.81×10^{1}	4.50×10^{-4}	1.14	158.23	0.30	82.75
	QD	2.14×10^{1}	3.75×10^{0}	1.48×10^{-4}	6.34	314.79	0.83	163.88
Arnoldi	D	2.32×10^{1}	4.61×10^{0}	1.42×10^{-1}	0.14	78.34	0.10	39.67
	DD	1.37×10^{1}	2.34×10^{0}	1.00×10^{-1}	7.32	158.90	5.86	81.70
	QD	7.84×10^{-12}	1.53×10^{-12}	1.00×10^{-22}	20.66	316.21	16.59	161.71
Householder	D	9.78×10^{-12}	6.30×10^{-13}	5.46×10^{-15}	0.30	154.72	-	-
	DD	1.82×10^{-12}	1.03×10^{-14}	4.51×10^{-31}	73.41	311.94	-	-
	QD	1.82×10^{-12}	1.03×10^{-14}	9.68×10^{-64}	694.42	621.71	-	-

Table 5 Elements outside the tridiagonal part for the Arnoldi method for Trefethen_200b

	D	DD	QD
$10^{0} \leq x$	1474	269	0
$10^{-5} \leq x < 10^{0}$	6718	1543	0
$10^{-10} \leq x < 10^{-5}$	7465	2096	0
$10^{-15} \leq x < 10^{-10}$	3846	2517	0
$10^{-20} \leq x < 10^{-15}$	0	3168	56
$10^{-30} \leq x < 10^{-20}$	0	9906	1104
$10^{-40} \leq x < 10^{-30}$	0	4	2515
$10^{-50} \leq x < 10^{-40}$	0	0	4548
$10^{-60} \leq x < 10^{-50}$	0	0	9967
$x < 10^{-60}$	0	0	1313
maximum	2.91×10^{2}	3.70×10^{2}	2.19×10^{-18}

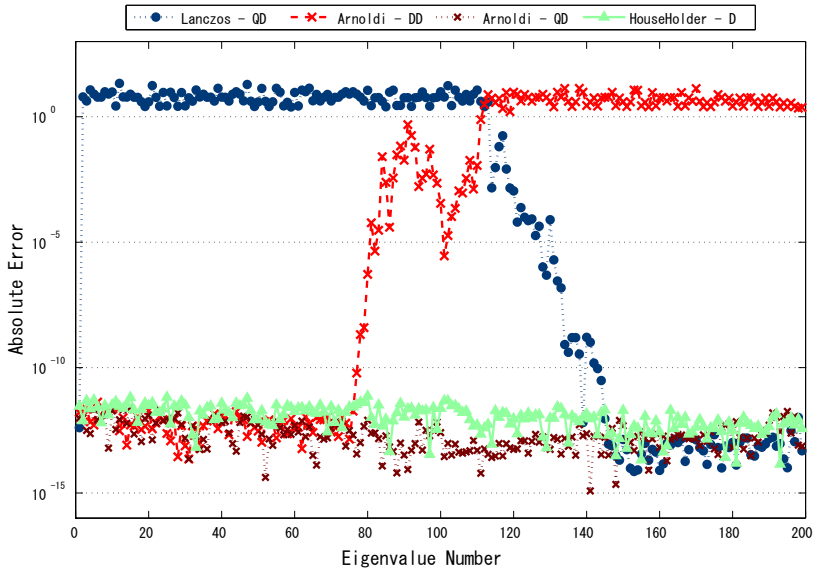

Fig. 3 Absolute error in eigenvalues for Trefethen_200b in descending order

4.3 Example 3: nos5 (slightly large problem)

We used the 'nos5' test matrix in Matrix Market [8]. The dimension of this matrix was 468, the number of nonzero elements was $5,172$, the condition number was 1.1×10^3, and the matrix originated from a structure problem. The eigenvalues of nos5 are distributed between 52.899482... and 582029.11... without any clustered eigenvalues.

Table 6 shows the results for various combinations of methods and precisions. Concerning the accuracy of the eigenvalues, the accuracy of Arnoldi-QD was not improved, but the accuracy of Householder-D was sufficient. With respect to the orthogonality, only Householder-D was sufficient. Although the condition number of nos5 was not so large, the Arnoldi method with QD cannot generate an accurate tridiagonal matrix. The modification of the implementation of the Lanczos method and the Arnoldi method and the choice of the initial value remain as areas for future research.

Arnoldi and Lanczos methods in high-precision arithmetic can not produce accurate eigenvalues in current implementation, for example, Lanczos-QD with sparse data structure can consume approximately the same as the computation time and the memory space with Householder-D. There are some possibility to improve the computation for high-precision arithmetics. By the compiled code and parallel process-

ing, the computation will be improved, however their codes depend on computing enviroment and loose ease of use.

In the Arnoldi method, the ratio of matrix-vector operations becomes smaller as the dimension of matrix becomes larger, but in the Lanczos method, the ratio is not changed regardless of the dimension. Thus, in the Arnoldi method, the speed up by the sparse data is small.

Table 6 Accuracy and tridiagonalization time [s] and memory space [KB] for nos5

Tridiagonal -ization		$max\|\lambda_i - \bar{\lambda}_i\|$	$avg\|\lambda_i - \bar{\lambda}_i\|$	$\frac{\|Q^TQ-I\|_F}{\|I\|_F}$	dense time	dense space	sparse time	sparse space
Lanczos	DD	1.31×10^5	5.27×10^4	7.59×10^{-1}	11.34	862.40	3.23	437.39
	QD	8.92×10^4	3.70×10^4	5.35×10^{-1}	56.03	1721.69	6.86	873.16
Arnoldi	DD	2.18×10^4	5.74×10^2	9.25×10^{-2}	199.91	860.62	188.57	434.76
	QD	2.15×10^4	2.87×10^2	6.54×10^{-2}	446.47	1718.11	409.35	867.83
Householder	D	6.64×10^{-9}	5.95×10^{-10}	1.04×10^{-14}	5.38	855.59	-	-

5 Concluding remarks

Although the authors believe that simple algorithms are good, floating-point number operations can break simple algorithms due to rounding errors. In the present paper, we attempted to stabilize the Lanczos and Arnoldi methods for tridiagonalization of symmetric matrices by using high-precision arithmetics; DD and QD. Since the Lanczos and Arnoldi methods are based on matrix-vector multiplication and do not change the given matrix, they have a possibility to be used for tridiagonalizing large sparse matrices.

We analyzed accuracy, numerical stability, and computing cost for tridiagonalization using dense and sparse matrix operations. We compared double (D), double-double (DD), and quad-double (QD) arithmetic for tridiagonalization by the Lanczos, Arnoldi, and Householder methods, and eigenvalue computation using the shifted QR method in only double-precision arithmetic.

The Lanczos method was stabilized by QD for only a small problem and required more precision. The Arnoldi method was also stabilized, although there were some problems in the case of relatively large test problems. A large matrix had some elements outside the tridiagonal part, resulting in an non-symmetric matrix. The Householder method was sufficient in double-precision arithmetic, but was not fit for large sparse matrices.

We conclude that a high-precision arithmetic is effective for tridiagonalization and no special technique is necessary for some problem. Lanczos and Arnoldi methods can work well with high-precision arithmetic. However, some improvement is necessary for other porblems. The best combination of algorithm and computing precision depends on the problem to be solved. The controlling precision in automatic is one of our future issues.

The sparse data type in MuPAT could reduce the required memory space and computation time for sparse matrices in high-precision arithmetic. For accelerating computation, parallel computing for these operations will be necessary. The analysis of the numerical stabilty and additional improvement of algorithms and implementation are our future issues.

Acknowledgements The authors would like to thank the reviewers for their careful reading and much helpful suggestions, and Mr. Takeru Shiiba in Tokyo University of Science for his kind support in numerical experiments. The present study was supported by the Grant-in-Aid for Scientific Research (C) No. 25330141 from the Japan Society for the Promotion of Science.

References

1. Golub, G. H. and Van Loan, C. F.: Matrix Computations, 4th edition. The Johns Hopkins University Press (2013)
2. Kikkawa, S., Saito, T., Ishiwata, E. and Hasegawa, H.: Development and acceleration of multiple precision arithmetic toolbox MuPAT for Scilab. JSIAM Letters **5**, 9–12 (2013)
3. Saito, T., Kikkawa, S., Ishiwata, E. and Hasegawa, H.: Effectiveness of Sparse Data Structure for Double-Double and Quad-Double Arithmetics. R. Wyrzykowski et al. (Eds.): PPAM 2013, Part I, LNCS **8384**, pp. 1–9 (2014)
4. Scilab. http://www.scilab.org/
5. Hida, Y., Li, X. S. and Bailey, D. H.: Quad-double arithmetic: algorithms, implementation, and application. Technical Report LBNL-46996 (2000).
6. Dekker, T. J.: A floating-point technique for extending the available precision. Numer. Math. **18**, 224–242 (1971)
7. Demmel, J. W.: Applied Numerical Linear Algebra. SIAM (1997)
8. MatrixMarket. http://math.nist.gov/MatrixMarket/
9. The University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse/matrices/