Exact Division applied to Diagonalizing Symmetric Matrices

Hoon Hong
Wolfgang Stöcher
RISC Linz
J. Kepler Universität, Linz, Austria

e-mail addresses:
Hoon.Hong@risc.uni-linz.ac.at
Wolfgang.Stoecher@risc.uni-linz.ac.at

June 23, 1994

Abstract

We show here that exact division, as described by Bareiss for Gaussian elimination, can be adotped straight forward to the congruent transformation of a symmetric integer matrix into diagonal form.

1 Introduction

Diagonalization of a symmetric matrix is important for computing the rank and signature, giving for instance information about the number of real or complex solutions of polynomial equation systems. As we need the sign of the diagonal elements, numerical algorithms often will not give the correct result: By accumulation of rounding errors a number close to zero might actually be zero or even a number with opposite sign. Therefor we need an algorithm with exact results.

As soon, as one deals with exact arithmetic, the time and storage consuming computation with arbitrarily long numbers comes in. For the implementation of an exact and efficient algorithm we will only consider an input matrix over $\mathbb{Z}$ and we want to make sure that all intermediate matrices and the result matrix are over $\mathbb{Z}$ as well. Usually an algorithm based on Gaussian elimination is used for congruent transformation of a symmetric matrix into diagonal form, where the involved numbers grow double exponentially. Here we show that analogous to [Bareiss 68] exact division can be applied, which makes the growth of the bitlength of the involved integers linear. This idea has also been investigated by ... .

In section 2 we give a review of exact division applied for triangularizing matrices by Gaussian elimination. In section 3 we propose our algorithm and prove it to be integer preserving. In section 4 we investigate the complexity of this algorithm.

Input: $A = (a_{ij})_{1 \leq i,j \leq n}$ of full rank, no pivoting necessary
Output: the triangularized input matrix in A
for $k = 1, \ldots n-1$ do
(1) for $j = k+1, \ldots n$ do
    (1.1) for $i = k, \ldots n$ do
        $a_{ji} \leftarrow (a_{kk} * a_{ji} - a_{jk} * a_{ki}) / a_{k-1,k-1}$

Figure 1: Algorithm Triangularization with Exact Division

Input: a symmetric matrix $A = (a_{ij})_{1 \leq i,j \leq n}$
Output: the congruent diagonalized matrix in A
$\boldsymbol{d} \leftarrow 1$
for $k = 1, \ldots n-1$ do
(1) if $a_{kk} = 0$ then [do pivoting:]
    (1.1) if $a_{jj} \neq 0$ for some $j : k < j \leq n$ then
        (1.1.1) for i = 1,...n: $a_{ki} \leftrightarrow a_{ji}$
        (1.1.2) for i = 1,...n: $a_{ik} \leftrightarrow a_{ij}$
    (1.2) else if $a_{jk} \neq 0$ for some $j : k < j \leq n$ then
        (1.2.1) for $i = 1, \ldots n$: $a_{ki} \leftarrow a_{ki} + a_{ji}$
        (1.2.2) for $i = 1, \ldots n$: $a_{ik} \leftarrow a_{ik} + a_{ij}$
(2) if $a_{kk} \neq 0$ then [do elimination:]
    (2.1) for $j = k+1, \ldots n$ do
        (2.1.1) for $i = k, \ldots n$ do
            $a_{ji} \leftarrow \boldsymbol{sign(}a_{kk}\boldsymbol{)} * (a_{kk} * a_{ji} - a_{jk} * a_{ki}) \boldsymbol{/ d}$
    (2.2) for $j = k+1, \ldots n$ do $a_{kj} \leftarrow 0$
    (2.3) $\boldsymbol{d} \leftarrow |a_{kk}|$

Figure 2: Algorithm Diagonalization with Exact Division

2 Review of Triangularization with Exact Division

Figure 1 contains the integer preserving algorithm for triangularization with exact division. This algorithm is Gaussian elimination enriched by the division by $a_{k-1,k-1}$ in step (1.1). In [Bareiss 68] this division is proven to be exact. We do not consider pivoting and allow only matrices of full rank as input, because this restricted case is sufficient for our needs in the next section.

3 Diagonalization with Exact Division

Figure 2 contains the algorithm of our interest for diagonalizing symmetric matrices. The multiplication with $sign(a_{kk})$ in step (2.1.1) and the zero assignments in step (2.2) make sure that the elimination is a congruent transformation. The division in step (2.1.1) is the only difference to the already known algorithm for diagonalizing a symmetric matrix based on Gaussian elimination. As the matrix need not be of full rank, some of the diagonal elements might be zero (and no elimination will be necessary then). So in $\boldsymbol{d}$ we keep the absolute value of the latest (nonzero) pivot element. We will show now that the division in step (2.1.1) in Algorithm Diagonalization with Exact Division is still exact (i. e. that the algorithm is integer preserving).

We first introduce some notations. Let $A^{(k)}$, $1 \leq k \leq n-1$, be the matrix $(a_{ij})_{1\leq i,j\leq n}$ at the time, when step (1) is entered for the k-th time during the run of the algorithm Diagonalization with Exact Division. $A^{(n)}$ shall be the matrix $(a_{ij})_{1\leq i,j\leq n}$ when the algorithm terminates. Let $\tilde{A}^{(k)}$, $1 \leq k \leq n-1$ be the matrices $(a_{ij})_{1\leq i,j\leq n}$ at the time, when step (2) is entered for the k-th time during the run of the algorithm.

Pivoting in the steps (1) – (1.2.2), when step (1) is entered for the k-th time, to get a nonzero value at the pivot position a_{kk} (if the whole pivot row is zero, than no elimination is necessary and at the pivot position a zero may remain) can be written as the congruent transformation:

$$\mathbf{P}_k: \qquad \tilde{A}^{(k)} = P_k A^{(k)} P_k^T, \tag{1}$$

where P_k is defined in the following way:

- If there is a nonzero value at the pivot position ($a_{kk}^{(k)} \neq 0$) or the whole pivot row and all further diagonal elements are zero ($a_{ki}^{(k)} = a_{ik}^{(k)} = a_{jj}^{(k)} = 0, 1 \leq i \leq n, k \leq j \leq n$), we take the identity matrix ($P_k = I$).
- If there is a zero at the pivot position ($a_{kk}^{(k)} = 0$), but there is another nonzero diagonal element $a_{jj}^{(k)}$ with $j > k$, then P_k is set to the identity matrix with the rows k and j exchanged. This gives $\tilde{a}_{kk}^{(k)} = a_{jj}^{(k)} \neq 0$.
- If the element at pivot position and all further diagonal elements are zero ($a_{ii}^{(k)} = 0, k \leq i \leq n$), and there is a nonzero element $a_{kj}^{(k)}$ with $j > k$ on the pivot row, we choose the identity matrix with additionally $p_{kj} = 1$. Thus we get $\tilde{a}_{kk}^{(k)} = a_{kj}^{(k)} + a_{jk}^{(k)} = 2a_{kj}^{(k)} \neq 0$.

Note that $\tilde{A}^{(k)}$ is over $\mathbb{Z}$ whenever $A^{(k)}$ is.

Now let us turn to the elimination in the steps (2) – (2.3). As the variable d contains the absolute value of the latest (nonzero) pivot element, which is $\tilde{a}_{jj}^{(k)} \neq 0$ with the largest possible j less than k, the elimination does only depend on the entering matrix $\tilde{A}^{(k)}$. Hence, for later needs, we can write the congruent transformation for the k-th elimination as a function:

$$\mathbf{E}_k: \qquad A^{(k+1)} = \mathbb{E}_k(\tilde{A}^{(k)}). \tag{2}$$

Setting $a_{00} := 1$ $\mathbb{E}_k$ is defined on any symmetric $n \times n$ matrix A for $1 \leq k \leq n-1$. We note that the transformations $\mathbf{P}_k$ and $\mathbf{E}_k$ leave the first $k-1$ rows and columns unchanged.

Before we deal with the general case let us consider input matrices for which no pivoting is necessary.

Lemma 3.1 *Let A_1 be a symmetric $n \times n$ input matrix over $\mathbb{Z}$ such that we can run the algorithm* Diagonlization with Exact Division *without pivoting (and still get the diagonlization as output). Then $A_k, 2 \leq k \leq n$, is also over $\mathbb{Z}$.* □

Proof: We prove the claim essentially by reducing this case to Bareiss' triangularization algorithm. Without pivoting there are only three differences to the algorithm Triangularization with Exact Division that matter:

1. First of all A_1 need not be of full rank. But this does not influence the divisibility. Let us assume that one of the rows is a linear combination of previous ones. When this row is reached for pivoting, it contains only zeros and is therefor skipped, as if it would not be there. Because of symmetry the same holds for columns. $\boldsymbol{d}$ remains unchanged and therefor keeps the (absolute value of the) last pivot element.
2. In step (2.2) the elements on the pivot row after the pivot element are set to zero. But this does not influence the further run of the algorithm, as those elements will not be used for consequent computations.
3. It remains to check the influence of the sign modification in step (2.1.1). Here we see that the sign of all elements being relevant for the consequent computations are altered in the same way. Hence this does not influence the divisibility by $\boldsymbol{d}$ in step (2.1.1) either.

Hence the computations in both algorithms are essentailly the same. Therefor all the matrices $A_k, 2 \leq k \leq n$, are over $\mathbb{Z}$ as well. □

The general case we now reduce to Lemma 3.1. The key observation for this is that elimination commutes with consequent pivoting. More precisely the congruent transformations $\mathbf{P}_i$ and $\mathbf{E}_k$ commute for $i > k$. This is expressed by the following Lemma.

Lemma 3.2 *Let A be a symmetric $n \times n$ matrix and let P_i be a $n \times n$ matrix of any of the forms as described above (not depending on A). Then the following equation holds:*

$$P_i\, E_k(A)\, P_i^T = E_k(P_i A P_i^T), \qquad \textit{whenever } i > k. \tag{3}$$

□

Proof: Let A and k be fixed. Then the elimination $\mathbb{E}_k$ can also be thought of as a function operating on the single rows $\vec{a}_j$ of A. From this point of view the function $\mathbb{E}_k$ is linear over the rows $\vec{a}_j$ with index $j > k$, i. e.

$$\mathbb{E}_k(\lambda_1 \vec{a}_{j_1} + \lambda_2 \vec{a}_{j_2}) = \lambda_1 \mathbb{E}_k(\vec{a}_{j_1}) + \lambda_2 \mathbb{E}_k(\vec{a}_{j_2}), \qquad k < j_1, j_2 \leq n, \tag{4}$$

which follows immediately from the definition of the elimination $\mathbb{E}_k$ by step (2.1.1) of the algorithm Diagonalization with Exact Division. Furthermore $\mathbb{E}_k$ is sort of linear on the rows with index $j > k$ *also with respect to linear operations on the* columns *with index $j > k$.* More precisely let L be a matrix such that, when multiplied from the right, it performs linear operations on the columns of A with index $j > k$ (i. e. multiplying them by a scalar and adding them to one another). Let l be a function on matrices defined by $l(A) := AL$ and let $l(\vec{a}_j)$ denote the j-th row of $l(A)$, then

$$\mathbb{E}_k(\, l(\vec{a}_j)) = l(\, \mathbb{E}_k(\vec{a}_j)), \qquad \text{whenever } j > k. \tag{5}$$

This again follows from the definition of the elimination $\mathbb{E}_k$. (5) contains (4) when L is also allowed to be multiplied from the left (performing linear operations on the *rows* of A with index $j > k$). Now pivoting with $P_i, i > k$, does nothing else but linear operations on the rows and columns with index $j \geq i > k$, and can therefor be written as a combination of such functions l. As $\mathbb{E}_k$ leaves the rows with index $j < k$ unchanged and in the k-th row just replaces the elements after the pivot element with zeros, which are not altered by linear operations, we have proven the equality of all rows in (3). □

Now we are prepared to prove that the algorithm Diagonalization with Exact Division is integer preserving also in the general case.

Theorem 3.1 *Let $A^{(1)}$ be a symmetric $n \times n$ matrix over $\mathbb{Z}$, on which we run the algorithm* Diagonalization with Exact Division. *Then $A^{(k)}, 2 \leq k \leq n$, is also over $\mathbb{Z}$.* □

Proof: We do the proof by reducing the claim to Lemma 3.1. Lemma 3.2 suggests that all pivoting can be done in advance. We define

$$\bar{A}_k := P_{n-1} \cdots P_k A^{(k)} P_k^T \cdots P_{n-1}^T, \qquad 1 \leq k \leq n. \tag{6}$$

Note that $\bar{A}_n = A^{(n)}$. Then we have the following recurrence for the sequence $\left(\bar{A}_k\right)_{1 \leq k \leq n}$:

$$\begin{aligned}
\bar{A}_{k+1} &= P_{n-1} \cdots P_{k+1}\, A^{(k+1)}\, P_{k+1}^T \cdots P_{n-1}^T \\
&= P_{n-1} \cdots P_{k+1}\, \mathbb{E}_k\,(P_k A^{(k)} P_k^T)\, P_{k+1}^T \cdots P_{n-1}^T \\
&= \mathbb{E}_k(P_{n-1} \cdots P_{k+1} P_k\, A^{(k)}\, P_k^T P_{k+1}^T \cdots P_{n-1}^T) \\
&= \mathbb{E}_k(\bar{A}_k).
\end{aligned}$$

Therefor we can run the algorithm Diagonalization with Exact Division on $\bar{A}_1$ as the input matrix without pivoting and still get the diagonlization as output. Hence Lemma 3.1 can be applied. Since $\bar{A}_1$ is over $\mathbb{Z}$ by (6) (the pivot transformations $\mathbf{P}_j$ are trivially integer preserving), all the matrices $\bar{A}_k, 2 \leq k \leq n$, are also over $\mathbb{Z}$. Now we can play this result back to the matrices $A^{(k)}$. Inverting (6) we get

$$A^{(k)} = P_k^{-1} \cdots P_{n-1}^{-1} \bar{A}_k P_{n-1}^{-T} \cdots P_k^{-T}. \tag{7}$$

The inverse pivoting transformation $A = P_i^{-1} \bar{A} P_i^{-T}$ operates on $\bar{A}$ in an analogous way as the pivoting transformation itself: Either two rows and columns are exchanged or rows and columns are subtracted from one another. Hence the inverse is also integer preserving. Now in (7) a sequence of $n - k$ such transformations operates on $\bar{A}_k$, which is a matrix over $\mathbb{Z}$. Thus also $A^{(k)}$ is over $\mathbb{Z}$. This completes the proof for showing that the algorithm Diagonalization with Exact Division is integer preserving. □

4 Complexity of Improved Algorithm

Let us first note that for an efficient implementation it is sufficient to compute the elements on and above the diagonal (since the input and all intermediate matrices are symmetric). Neglecting the time for pivoting and concentrating on the number of multiplations, we have to compute for each $k, 1 \leq k \leq n-1$ the $\binom{n-k+1}{2}$ elements $a_{ji}, k < j \leq i \leq n$ (steps (2.1), (2.1.1)), i. e. $(n-k)(n-k+1)$ multiplications. Having an initial maximum bitlength of l among the elements of the input matrix $A^{(1)}$, the maximum bitlength among elements in $\bar{A}_1$, as introduced in the proof of Theorem 3.1, is $l + n - 1$. (By (6) $\bar{A}_1$ is derived from $A^{(1)}$ with $n-1$ pivot transformations, each of them adding to rows and columns in the worst case, and therefor increasing the maximum bitlength at most by one.) The maximum bitlength among the elements of $\bar{A}^{(k)}$ is then the same as for Gaussian elimination: it is bounded by $k(l + n - 1)$ (see for instance **[Stöcher, Wegschaider 93]**). By (7) we see that the maximum bitlength among elements of $A^{(k)}$ is bounded by $k(l + n - 1) + n - k = kl + (k+1)n - 2k$, as each transformation $\mathbf{P}_j^{-1}$ again increases the maximum bitlength at most by one. Furthermore let us assume that the mutiplication of two numbers of bitlength b is of order $O(b^2)$, then the order of the runtime of the algorithm Diagonalization with Exact Division is

$$O\left(\sum_{k=1}^{n-1}(n-k)(n-k+1)(kl + (k+1)n - 2k)^2\right) = O(n^7 + n^6 l + n^5 l^2).$$

Neglecting the effect of pivoting on the maximum bitlength (which is the case in average) we have an order of the runtime of

$$O\left(\sum_{k=1}^{n-1}(n-k)(n-k+1)(kl)^2\right) = O(n^5 l^2).$$

References

[Bareiss 68] E. H. Bareiss. *Sylvester's Identity and Multistep Integer Preserving Gaussian Elimination.* Mathematics of Computation 22

[StoWe 93] W. Stöcher, K. Wegschaider. *AlgConSolv I Project-Report: Linear Systems over Integers.* Preprint