Algorithms for Recursive Block Matrices

Stephen M. Watt
University of Waterloo
smwatt@uwaterloo.ca

May 20, 2024

Abstract

We study certain linear algebra algorithms for recursive block matrices. This representation has useful practical and theoretical properties. We summarize some previous results for block matrix inversion and present some results on triangular decomposition of block matrices. The case of inverting matrices over a ring that is neither formally real nor formally complex was inspired by Gonzalez-Vega *et al.*

1 Introduction

Algorithms on block matrices have both useful theoretical and practical properties. From a theoretical point of view, algorithms for matrices with non-commuting elements allow recursive formulation of linear algebra problems, simplifying complexity analysis. Strassen's seminal result on matrix multiplication [8] is perfect example. From a practical point of view, block matrices provide a middle ground that avoids pathological communication bottlenecks in row-major or column-major code [4]. Recursive block matrices allow both dense and structured matrices to be represented with reasonable efficiency [1, 5]. For these reasons, having recursive block matrix representations in mathematical software is desirable.

Modern programming languages used in mathematical computing, including `C++`, Julia, Python and Fortran 2023, provide data abstraction mechanisms that can support recursive block matrices in a natural way. However, in designing a library for one such language, Aldor [9], that the standard algorithms for block matrices sometimes require breaking the block abstraction. In building a type of block $n \times n$ matrices over a ring R, providing ring operations on $R^{n\times n}$ is straightforward. However providing a partial function for matrix inverse is not. The usual formulation to invert a 2×2 block matrix requires at least one block and its Schur complement to be invertible. But this may not be the case — a nonsingular matrix may consist entirely of singular blocks. In situations such as this, the standard methods break the block abstraction and work on rows, *e.g.* [2].

In this article we explore algorithms for recursive block matrices that respect the block abstraction. We begin by summarizing earlier results for a matrix inversion method. The method is directly applicable to matrices over a ring with a formally real sub-field. The method may be generalized to matrices over other rings using a technique suggested by Gonzalez-Vega *et al.* [3]. Following this, we undertake some exploration into triangular decomposition of recursive block matrices.

2 Inversion of Recursive Block Matrices

Most ring operations on block matrices may be performed in a straightforward manner using only block operations. That is, for a block matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \in R^{2n\times 2n}$$

only ring operations on $A, B, C, D \in R^{n\times n}$ are needed. If all of the blocks of M are invertible, the inverse of M may be computed as

$$M^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & (C - DB^{-1}A)^{-1} \\ (B - AC^{-1}D)^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}.$$

In practice, only two inverses are required—that of A and its Schur complement, $S_A = D - CA^{-1}B$,

$$M^{-1} = \begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & S_A^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix} = \begin{bmatrix} A^{-1} + A^{-1}BS_A^{-1}CA^{-1} & -A^{-1}BS_A^{-1} \\ -S_A^{-1}CA^{-1} & S_A^{-1} \end{bmatrix}. \quad (1)$$

If A is not invertible, then a similar formula involving the inverse of another block and its Schur complement may be used, perhaps after a permutation of rows or columns. The problem with this approach is that M may be invertible even when all of A, B, C and D are singular. In this situation, permuting the blocks is of no help. One approach is to break the block abstraction and use operations on whole rows of M viewed as a flat $2^{2n\times 2n}$ matrix [2].

In earlier work [10], we have shown a recursive algorithm to invert matrices respecting a block abstraction. In particular, row operations are not required for pivoting or otherwise. The technique is to use the Moore-Penrose inverse so that the principal minors are guaranteed to be invertible and equation (1) may be used. We summarize those results here.

We use the notation $R^{[2^k\times 2^k]}$ to mean the ring of $2^k \times 2^k$ matrices with elements in R, structured in recursive 2×2 blocks. Any $n\times n$ matrix may be easily be embedded in such a ring.

Theorem 1. *If R is a formally real division ring and $M \in R^{n\times n}$ is invertible, then it is possible to compute M^{-1} as $(M^TM)^{-1}M^T$ using only block operations.*

By *block operations*, we mean ring operations in $R^{[2^{k-1}\times 2^{k-1}]}$. Examples of formally real rings are $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{Q}[\sqrt{2}]$ and $R[x, \partial]$ for formally real R.

Theorem 2. *Let C be a division ring with a formally real sub-ring R and involution "$*$", such that for all $c \in C$, $c^* \times c$ is a sum of squares in R. If $M \in C^{[2^k\times 2^k]}$ is invertible, then it is possible to compute M^{-1} as $(M^*M)^{-1}M^*$ using only block operations.*

Examples of such rings are the complexification of a formally real ring R as $R[i]/\langle i^2+1\rangle$ or quaternions over R with the involution $(a + bi + cj + dk)^* = a - bi - cj - dk$.

The next result follows an observation of Laureano Gonzalez-Vega, using a technique of [3, 7].

Theorem 3. *Let K be a field. If $M \in K^{2^k\times 2^k}$ is invertible, then it is possible to compute M^{-1} as $(M^\circ M)^{-1}M^\circ$ using only block operations.*

Here *block operations* mean ring operations in $K(t)^{[2^{k-1}\times 2^{k-1}]}$ and $M^\circ = Q_n^{-1} M^T Q_n$ is a group conjugate of M^T, with $Q_n = \mathrm{diag}(1, t, \ldots, t^{n-1})$.

In all cases, the time complexity is that of two $2^k \times 2^k$ matrix multiplications and one inversion using (1). The inversion may be achieved with two $2^{k-1} \times 2^{k-1}$ inversions and two $2^{k-1} \times 2^{k-1}$ multiplications to compute S_A and its inverse, and four $2^{k-1} \times 2^{k-1}$ multiplications, namely

$$t_1 = C \cdot A^{-1} \qquad t_2 = A^{-1} \cdot B \qquad t_3 = t_2 \cdot S_A^{-1} \qquad t_4 = t_3 \cdot t_1,$$

relying on the symmetries $(M^T M)_{ji} = (M^T M)_{ij}$, $(M^* M)_{ji} = (M^* M)_{ij}{}^*$ and $(M^\circ M)_{ji} = t^{i-j}(M^\circ M)_{ij}$ for $j \geq i$. Thus,

$$T_{\mathrm{inv}}(2^k) = 2T_\times(2^k) + 2T_{\mathrm{inv}}(2^{k-1}) + 4T_\times(2^{k-1})$$

where $T_\times(n)$ is the time complexity to multiply two $n \times n$ matrices. If $T_\times(n) = \alpha n^\omega$ and $T(1) = 1$, then

$$T_{\mathrm{inv}}(n) = 2\alpha n^\omega - (2\alpha - 1)n + \frac{8\alpha(2^\omega + 2)(n^\omega - n)}{4^\omega - 4} \in O(T_\times(n)).$$

This complexity was not spelled out in [10].

3 LU Decomposition of Recursive Block Matrices

It is often desirable to factor M as $L \cdot U$ where L and U are respectively lower- and upper-triangular.

Let M be a nonsingular 2×2 square block matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

We show how to compute such lower-triangular L and upper triangular U such that $M = PLUQ$ with permutation matrices P and Q.

3.1 Assuming a Nonsingular Block

If one of A, B, C or D is nonsingular, permute the rows and columns of M as necessary to obtain M' with nonsingular M'_{11}.

With $T = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$, let $M' = P_1\, M\, Q_1$ where

$$P_1 = \begin{cases} I & \text{when } A \text{ or } B \text{ nonsingular} \\ T & \text{otherwise} \end{cases} \qquad Q_1 = \begin{cases} I & \text{when } A \text{ or } C \text{ nonsingular} \\ T & \text{otherwise.} \end{cases}$$

If, at the recursive step, A' is arranged to A'' so A''_{11} is nonsingular, then we will have $M'' = P_2 P_1 M Q_1 Q_2$, *etc.* We can now drop the P_i, Q_i, and the primes in what follows.

3.2 Block LU Decomposition

It is well-known that M may be factored in an LDU decomposition as

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & S_A \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}. \tag{2}$$

It is then possible to multiply the block diagonal middle factor on either the left or right, depending on desired convention, to obtain a block LU decomposition.

We are interested, however, in an LU decomposition of M. Regardless of whether we multiply the middle factor to the left or right, the block LU decomposition will in general have non-triangular A as the $(1, 1)$ component of either L or U. So (2) is not what we want.

We could iterate the process and next find LDU decompositions of A and and S_A, and so on, combining lower triangular matrices on the left and upper triangular matrices on the right. In this situation some of the intermediate multiplications will be specialized, and it is more convenient to consider LU decomposition directly.

3.3 LU Decomposition

The problem is to find lower and upper triangular matrices L and U such that

$$M = LU \qquad L = \begin{bmatrix} L_1 & 0 \\ X & L_2 \end{bmatrix} \qquad U = \begin{bmatrix} U_1 & Y \\ 0 & U_2 \end{bmatrix}$$

with L_1 and L_2 lower triangular and U_1 and U_2 upper triangular. This LU decomposition is different from the LDU decomposition which has L and U as above, but with $L_i = U_i = I$ and D diagonal. It is therefore suitable for recursive application.

The four components of $M = LU$ are

$$\begin{aligned} A &= L_1U_1 & B &= L_1Y \\ C &= X\,U_1 & D &= L_2U_2 + XY. \end{aligned}$$

Assuming A is nonsingular, these give two recursive LU decompositions

$$A = L_1U_1 \qquad L_2U_2 = D - XY$$

in which case X and Y may be obtained as

$$X = CU_1^{-1} \qquad Y = L_1^{-1}B.$$

Note that if A is nonsingular, so are L_1 and U_1.

The matrices L_i, U_i, X and Y may be computed as:

$$\begin{aligned} L_1U_1 &= A \\ X &= CU_1^{-1} \\ Y &= L_1^{-1}B \\ L_2U_2 &= D - XY. \end{aligned}$$

The degree of freedom in choosing the diagonal elements of L and U is handled by taking the same convention in the recursive computations of L_1U_1 and L_2U_2.

3.4 Complexity Analysis

We have shown, subject to nonsingularity of a block, that the LU decomposition of an $n \times n$ matrix may be computed with $T_{LU}(n)$ multiplications using:

- 2 size $n/2$ LU decompositions, $2T_{LU}(n/2)$,
- 2 size $n/2$ triangular matrix inversions, $2T_{\triangle^{-1}}(n/2)$.
- 1 size $n/2$ general matrix multiplication, $T_\times(n/2)$.
- 2 size $n/2$ triangular times general matrix multiplications, $2T_{\triangle\times}(n/2)$.

The number of multiplications for LU decomposition by this method is therefore

$$T_{LU}(n) = 2T_{LU}(n/2) + 2T_{\triangle^{-1}}(n/2) + T_\times(n/2) + 2T_{\triangle\times}(n/2).$$

The number of multiplications and divisions to invert a triangular matrix is $T_{\triangle^{-1}}(n) = \frac{1}{2}n(n+1)$.

To multiply a triangular and general matrix of size $n \times n$, form

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} U & 0 \\ V & W \end{bmatrix} = \begin{bmatrix} AU + BV & BW \\ CU + DV & DW \end{bmatrix}$$

where U and W are lower triangular. So $T_{\Delta\times}(n) = 4T_{\Delta\times}(n/2) + 2T_\times(n/2)$. If $T_\times(n) = \alpha n^\omega$, then

$$T_{\triangle\times}(n) = \frac{2\alpha}{2^\omega - 4}(n^\omega - n^2) + n^2$$

and the LU decomposition method requires a number of multiplications

$$T_{LU}(n) = \alpha(n^\omega - n)\frac{2^\omega}{(2^\omega - 2)(2^\omega - 4)} + (n^2 - n)\left(\frac{3}{2} - \frac{2\alpha}{2^\omega - 4}\right) + \tfrac{1}{2}n\log_2(n) + n \in O(T_\times(n)). \tag{3}$$

3.5 When All Blocks Are Singular

It remains to handle when A, B, C and D are all singular. If M is nonsingular and its elements are from a sufficiently large domain, then one approach would be through randomization. Letting R_L and R_U be random lower and upper triangular matrices, one can with high probability compute $L'U' = R_LMR_U$ as above, so $L = R_L^{-1}L'$ and $U = U'R_U^{-1}$. Also, note that LU decomposition is meaningful when M is singular [6]. This remains a topic of on-going work.

References

[1] S.K. Abdali and D.S. Wise. Experiments with quadtree representation of matrices. In P. Gianni, editor, *Proc. ISSAC 88*, page 96–108. Springer Verlag LNCS 358, 1989.

[2] Alfred V. Aho, John E Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

[3] G.M. Diaz-Toca, L. Gonzalez-Vega, and H. Lombardi. Generalizing cramer's rule: Solving uniformly linear systems of equations. *SIAM J. Matrix Anal. Appl.*, 27(3):621–637, 2005.

[4] Jack J. Dongarra, Robert A. van de Geun, and David W. Walker. Scalability issues affecting the design of a dense linear algebra library. *J. Parallel and Distributed Computing*, 22:52–537, 1994.

[5] Irene Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[6] David J. Jeffrey. LU factoring of non-invertible matrices. *ACM Communications in Computer Algebra*, 44:1–8, 2010.

[7] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7:101–104, 1987.

[8] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[9] Stephen M. Watt. Aldor. In *Handbook of Computer Algebra*, page 265–270. Springer Verlag, 2003.

[10] Stephen M. Watt. Pivot-free block matrix inversion. In *Proc. 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006)*, pages 151–155, 2006.