10-1-2003

Fast Multiplication of Interval Matrices (Interval Version of Strassen's Algorithm)

Martine Ceberio
University of Texas at El Paso, mceberio@utep.edu

Vladik Kreinovich
University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-03-27
Published in *Reliable Computing*, 2004, Vol. 10, No. 3, pp. 241-243.

Fast Multiplication of Interval Matrices (Interval Version of Strassen's Algorithm)

Martine Ceberio and Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
{mceberio,vladik}@cs.utep.edu

Abstract

Strassen's algorithm multiplies two numerical matrices fast, but when applied to interval matrices, leads to excess width. We use Rump's interval arithmetic to propose an interval version of Strassen's algorithm whose only excess width is in second order terms.

Formulation of the problem. Many numerical algorithms – ranging from mathematical physics to ranking webpages – include matrix multiplication, and multiplication of large matrices often takes a significant portion of the algorithm's running time. It is therefore desirable to multiply matrices fast.

The product $C = AB = (c_{ij})_{i,j}$ of the matrices $A = (a_{ij})_{i,j}$ and $B = (b_{ij})_{i,j}$ is defined as

$$c_{ij} = \sum_k a_{ik} \cdot b_{kj}. \tag{1}$$

A straightforward algorithm for computing the product of two $n \times n$ matrices therefore requires $O(n)$ arithmetic operations to compute each of $n \times n$ elements c_{ij} – to the total of $O(n^3)$ operations.

The first faster algorithm was proposed by Strassen in his 1969 paper [7]; this algorithm enables us to multiply two matrices in time $O(n^{\log_2 7})$ (i.e., in $O(n^{2.81})$ steps). Since then, even faster algorithms have been proposed; the fastest one [2] requires only $O(n^{2.376})$ steps.

For a detailed description of Strassen's algorithm, its advantages and disadvantages, and more recent matrix multiplication algorithms see [3]. In particular, according to this exposition, one of the main disadvantages of Strassen's and similar algorithms is that they are much more numerically unstable than the straightforward $O(n^3)$ matrix multiplication.

For example, if we only know the values a_{ij} and b_{ij} with interval uncertainty – in other words, if, instead of the actual (unknown) matrices A and B, we only know the interval matrices $\mathbf{A} = (\mathbf{a}_{ij})$ and $\mathbf{B} = (\mathbf{b}_{ij})$ that contain A and B – then the straightforward algorithm leads to the exact range $\mathbf{c}_{ij}$ of c_{ij}, while Strassen's and other algorithms lead to excess width; see, e.g., [1].

The reason for this excess width is simple. In the formula (1), each variable occurs only once hence, as is all other single-use expressions (SUE), straightforward interval computations leads to the exact range (see, e.g., [4, 5]). For Strassen's algorithm, e.g., for $n = 2$, c_{12} is computed as $c_{12} = a_{11} \cdot (b_{12} - b_{22}) + (a_{11} + a_{12}) \cdot b_{22}$; the resulting dependency problems leads to excess width. Of course, we can apply algebraic transformations to eliminate this dependency – but then we are back to the expression (1), i.e., we lose speed.

In this paper, we show that we can decrease the excess width while preserving the algorithm's speed.

Rump's operations. In our algorithm, instead of the standard formulas for interval computations – formulas that describe, for arithmetic expressions, the exact range – we will use the simplified formulas first proposed by S. Rump (see, e.g., [6]). In these formulas, every interval $\mathbf{a} = [\underline{a}, \overline{a}]$ is represented by its midpoint

$\widetilde{a} = (\underline{a} + \overline{a})/2$ and its half-width (radius) $\Delta^a = (\overline{a} - \underline{a})/2$, so that $\mathbf{a} = [\widetilde{a} - \Delta^a, \widetilde{a} + \Delta^a]$, and the corresponding arithmetic operations take the following form:

$$[\widetilde{a} - \Delta^a, \widetilde{a} + \Delta^a] \odot [\widetilde{b} - \Delta^b, \widetilde{b} + \Delta^b] = [\widetilde{c} - \Delta^c, \widetilde{c} + \Delta^c], \tag{2}$$

where:

- for $a \odot b = a + b$, we have $\widetilde{c} = \widetilde{a} + \widetilde{b}$ and $\Delta^c = \Delta^a + \Delta^b$;
- for $a \odot b = a - b$, we have $\widetilde{c} = \widetilde{a} - \widetilde{b}$ and $\Delta^c = \Delta^a + \Delta^b$;
- for $a \odot b = a \cdot b$, we have $\widetilde{c} = \widetilde{a} \cdot \widetilde{b}$ and $\Delta^c = |\widetilde{a}| \cdot \Delta^b + |\widetilde{b}| \cdot \Delta^a + \Delta^a \cdot \Delta^b$.

For addition and subtraction, these formulas are the same as the standard ones (and thus, lead to the exact interval range). For multiplication, if we only consider the first order terms in terms of the half-widths Δ^a and Δ^b of the intervals $\mathbf{a}$ and $\mathbf{b}$, the new formula is exact; it does, however, lead to excess width if we take second order terms into account.

We are using these not very exact formulas because they are faster (as the very title of Rump's paper [6] shows), and they are exact when it comes to first order terms.

Resulting algorithm. For each interval matrix $\mathbf{A}$ with elements $\mathbf{a}_{ij} = [\widetilde{a}_{ij} - \Delta^a_{ij}, \widetilde{a}_{ij} + \Delta^a_{ij}]$, let us denote, by $\widetilde{A}$, the matrix formed by midpoints $\widetilde{a}_{ij}$, and by Δ^A, the matrix formed by the interval half-widths Δ^a_{ij}.

Applying Rump's formulas to the expression (1), we conclude that each element

$$\mathbf{c}_{ij} = [\widetilde{c}_{ij} - \Delta^c_{ij}, \widetilde{c}_{ij} + \Delta^c_{ij}]$$

of the resulting interval matrix $\mathbf{C}$ has the following form:

$$\widetilde{c}_{ij} = \sum_k \widetilde{a}_{ik} \cdot \widetilde{b}_{kj}; \tag{3}$$

$$\Delta^c_{ij} = \sum_k (|\widetilde{a}_{ik}| \cdot \Delta^b_{kj} + \Delta^a_{ik} \cdot |\widetilde{b}_{kj}| + \Delta^a_{ik} \cdot \Delta^b_{kj}), \tag{4}$$

i.e.,

$$\widetilde{C} = \widetilde{A}\widetilde{B} \tag{5}$$

and $\Delta^C = |\widetilde{A}|\Delta^B + \Delta^A|\widetilde{B}| + \Delta^A\Delta^B$, where $|\widetilde{A}|$ denotes a matrix with elements $|\widetilde{a}_{ij}|$.

Therefore, to compute $\widetilde{C}$ and Δ^C, we can use the formulas (5) and

$$\Delta^C = (|\widetilde{A}| + \Delta^A)(|\widetilde{B}| + \Delta^B) - |\widetilde{A}||\widetilde{B}|. \tag{6}$$

These formulas reduces the multiplication of two $n \times n$ interval matrices to 3 multiplications of $n \times n$ numerical matrices (and also three additions and subtractions of matrices, which only take $O(n^2)$ time). Thus, if we use the $O(n^{\log_2 7})$ Strassen's algorithm to multiply the corresponding numerical matrices, we thus get a $O(n^{\log_2 7})$ algorithm for computing interval matrices.

Similarly, for every $\alpha \geq 2$ for which there is a $O(n^\alpha)$ algorithm to multiply numerical matrices, if we apply this algorithm to compute the products $\widetilde{A}\widetilde{B}$, $(|\widetilde{A}| + \Delta^A)(|\widetilde{B}| + \Delta^B)$, and $|\widetilde{A}||\widetilde{B}|$, we thus get a $O(n^\alpha)$ algorithm for computing interval matrices.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-1-0365, by NSF grants EAR-0112968 and EAR-0225670, and by Army Research Laboratories grant DATM-05-02-C-0046.

The authors are thankful to Andreas Griewank for the discussions that led to this result.

References

[1] R. Castrapel, *Analysis of Error Propagation in Strassen's $M \times M$ Algorithm Using Interval Arithmetic*, Sun Microsystems Technical Report, June 2001
www.sun.com/products-n-solutions/edu/events/archive/hpc/presentations/june01/rick_castrapel.pdf

[2] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progression", *Journal of Symbolic Computation*, 1990, Vol. 9, No. 3, pp. 251–280.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., N.Y., 2001.

[4] E. Hansen, "Sharpness in interval computations", *Reliable Computing*, 1997, Vol. 3, pp. 7–29.

[5] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer, London, 2001.

[6] S. M. Rump, "Fast and parallel interval arithmetic", *BIT Numerical Mathematics*, 1999, Vol. 39, No. 3, pp. 534–554.

[7] V. Strassen, "Gaussian Elimination is Not Optimal", *Numerische Mathematick*, 1969, Vol. 14, No. 3, pp. 354–356.