Differentiation of Kaltofen's division-free determinant algorithm

Gilles Villard

DIFFERENTIATION OF KALTOFEN'S DIVISION-FREE DETERMINANT ALGORITHM

Abstract

Gilles Villard
CNRS, Université de Lyon
Laboratoire LIP, CNRS-ENSL-INRIA-UCBL
46, Allée d'Italie, 69364 Lyon Cedex 07, France
http://perso.ens-lyon.fr/gilles.villard

Kaltofen has proposed a new approach in [8] for computing matrix determinants. The algorithm is based on a baby steps/giant steps construction of Krylov subspaces, and computes the determinant as the constant term of a characteristic polynomial. For matrices over an abstract field and by the results of Baur and Strassen [1], the determinant algorithm, actually a straight-line program, leads to an algorithm with the same complexity for computing the adjoint of a matrix [8]. However, the latter is obtained by the reverse mode of automatic differentiation and somehow is not "explicit". We study this adjoint algorithm, show how it can be implemented (without resorting to an automatic transformation), and demonstrate its use on polynomial matrices.

Kaltofen has proposed in [8] a new approach for computing matrix determinants. This approach has brought breakthrough ideas for improving the complexity estimate for the problem of computing the determinant without divisions over an abstract ring [8, 11]. The same ideas also lead to the currently best known bit complexity estimates for some problems on integer matrices such as the problem of computing the characteristic polynomial [11].

We consider the straigth-line programs of [8] for computing the determinant over abstract fields or rings (with or without divisions). Using the reverse mode of automatic differentiation (see [12, 13, 14]), a straight-line program for computing the determinant of a matrix A can be (automatically) transformed into a program for computing the adjoint matrix A^* of A [1] (see the application in [8, §1.2] and [11, Theorem 5.1]). Since the latter program is derived by an automatic process, few is known about the way it computes the adjoint. The only available information seems to be the determinant program itself and the knowledge we have on the differentiation process. In this paper we study the adjoint programs that would be automatically generated by differentiation from Kaltofen's determinant programs. We show how they can be implemented with and without divisions, and study their behaviour on univariate polynomial matrices.

Our motivation for studying the differentiation and resulting adjoint algorithms is the importance of the determinant approach of [8, 11] for various complexity estimates. Recent advances around the determinant of polynomial or integer matrices [5, 11, 15, 16], and the adjoint of a univariate polynomial matrix in the generic case [7], also justify the study of the general adjoint problem.

1 Kaltofen's determinant algorithm

Let K be a commutative field. We consider $A \in \mathsf{K}^{n\times n}$, $u \in \mathsf{K}^{n\times 1}$, and $v \in \mathsf{K}^{n\times 1}$. Kaltofen's approach extends the Krylov-based methods of [18, 9, 10]. We introduce the Hankel matrix $H = (uA^{i+j-2}v)_{ij} \in \mathsf{K}^{n\times n}$, and let $h_k = uA^k v$ for $0 \le k \le 2n-1$. We assume that H is non-singular. In the applications the latter is ensured either by construction of A, u, and v [8, 11], or by randomization (see [11] and references therein).

With baby steps/giant steps parameters $r = \lceil 2n/s \rceil$ and $s = \lceil \sqrt{n} \rceil$ ($rs \geq 2n$) we consider the following algorithm (the algorithm without divisions will be described in Section 3).

Algorithm DET [8]

STEP 1. For $i = 0, 1, \ldots, r-1$ Do $v_i := A^i v$;
STEP 2. $B = A^r$;
STEP 3. For $j = 0, 1, \ldots, s-1$ Do $u_j := uB^j$;
STEP 4. For $i = 0, 1, \ldots, r-1$ Do
For $j = 0, 1, \ldots, s-1$ Do $h_{i+jr} := u_j v_i$;
STEP 5. Compute the minimum polynomial $f(\lambda)$ of the sequence $\{h_k\}_{0 \leq k \leq 2n-1}$;
Return $f(0)$.

2 The ajoint algorithm

The determinant of A is a polynomial in $\mathsf{K}[a_{11}, \ldots, a_{ij}, \ldots, a_{nn}]$ of the entries of A. If we denote the adjoint matrix by A^* such that $AA^* = A^*A = (\det A)I$, then the entries of A^* satisfy [1]:

$$a^*_{j,i} = \frac{\partial \Delta}{\partial a_{i,j}}, 1 \leq i, j \leq n. \tag{1}$$

The reverse mode of automatic differentiation (see [1, 12, 13, 14]) allows to transform a program which computes Δ into a program which computes all the partial derivatives in (1). We apply the transformation process to Algorithm DET.

The flow of computation for the adjoint is reversed compared to the flow of Algorithm DET. Hence we start with the differentiation of STEP 5. Consider the $n \times n$ Hankel matrices $H = (uA^{i+j-2}v)_{ij}$ and $H_A = (uA^{i+j-1}v)_{ij}$. Then the determinant $f(0)$ is computed as

$$\Delta = (\det H_A)/(\det H).$$

Viewing Δ as a function Δ_5 of the h_k's, we show that

$$\frac{\partial \Delta_5}{\partial h_k} = (\varphi_{k-1}(H_A^{-1}) - \varphi_k(H^{-1}))\Delta \tag{2}$$

where for a matrix $M = (m_{ij})$ we define $\varphi_k(M) = 0 + \sum_{i+j-2=k} m_{ij}$ for $1 \leq k \leq 2n-1$. Identity (2) gives the first step of the adjoint algorithm. Over an abstract field, and using intermediate data from Algorithm DET, its costs is essentially the cost of a Hankel matrix inversion.

For differentiating STEP 4, Δ is seen as a function Δ_4 of the v_i's and u_j's. The entries of v_i are involved in the computation of the s scalars $h_i, h_{i+r}, \ldots, h_{i+(s-1)r}$. The entries of u_j are used for computing the r scalars $h_{jr}, h_{1+jr}, \ldots, h_{(r-1)+jr}$. Let ∂v_i be the $1 \times n$ vector, respectively the $n \times 1$ vector ∂u_j, whose entries are the derivatives of Δ_4 with respect to the entries of v_i, respectively u_j. We show that

$$\begin{bmatrix} \partial v_0 \\ \partial v_1 \\ \vdots \\ \partial v_{r-1} \end{bmatrix} = H^v \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{s-1} \end{bmatrix} \tag{3}$$

and

$$\begin{bmatrix} \partial u_0, \partial u_1, \ldots \partial u_{s-1} \end{bmatrix} = \begin{bmatrix} v_0, v_1, \ldots v_{r-1} \end{bmatrix} H^u \tag{4}$$

where H^v and H^u are $r \times s$ matrices whose entries are selected $\partial\Delta_5/\partial h_k$'s. Identities (3) and (4) give the second step of the adjoint algorithm. Its costs is essentially the cost of two $n \times \sqrt{n}$ by $\sqrt{n} \times \sqrt{n}$ (unstructured) matrix products.

Note that (2), (3) and (4) somehow call to mind the matrix factorizations [3, (3.5)] (our objectives are similar to Eberly's ones) and [4, (3.1)].

Steps 3-1 of Det may then be differentiated. For differentiating Step 3 we recursively compute an $n \times n$ matrix ∂B from the δu_j's. The matrix ∂B gives the derivatives of Δ_3 (the determinant seen as a function of B and the v_i's) with respect to the entries of B.

For Step 2 we recursively compute from δB an $n \times n$ matrix δA that gives the derivatives of Δ_2 (the determinant seen as a function of v_i's).

Then the differentiation of Step 1 computes from δA and the δv_i's an update of δA that gives the derivatives of $\Delta_1 = \Delta$. From (1) we know that $A^* = (\delta A)^T$.

The recursive process for differentiating Step 3 to Step 1 may be written in terms of the differentiation of the basic operation (or its transposed operation)

$$q := p \times M \tag{5}$$

where p and q are row vectors of dimension n and M is an $n \times n$ matrix. We assume at this point (recursive process) that column vectors δp and δq of derivatives with respect to the entries of p and q are available. We also assume that an $n \times n$ matrix δM that gives the derivatives with respect to the m_{ij}'s has been computed. We show that differentiating (5) amounts to updating δp and δM as follows:

$$\begin{cases} \delta p := \delta p + M \times \delta q, \\ \delta M := \delta M + p^T \times (\delta q)^T. \end{cases} \tag{6}$$

We see that the complexity is essentially preserved between (5) and (6) and corresponds to a matrix by vector product. In particular, if Step 2 of Algorithm Det is implemented in $O(\log r)$ matrix products, then Step 2 differentiation will cost $O(n^3 \log r)$ operations (by decomposing the $O(n^3)$ matrix product).

Let us call Adjoint the algorithm just described for computing A^*.

3 Application to computing the adjoint without divisions

Now let A be an $n \times n$ matrix over an abstract ring R. Kaltofen's method for computing the determinant of A without divisions applies Algorithm Det on a well chosen univariate polynomial matrix $Z(z) = C + z(A - C)$ where $C \in \mathbb{Z}^{n \times n}$. The choice of C as well as a dedicated choice for the projections u and v allow the use of Strassen's general method of avoiding divisions [17, 8]. The determinant is a polynomial Δ of degree n, the arithmetic operations in Det are replaced by operations on power series modulo z^{n+1}. Once the determinant of $Z(z)$ is computed, $(\det Z)(1) = \det(C + 1 \times (A - C))$ gives the determinant of A.

In Step 1 and Step 2 in Algorithm Det applied to $Z(z)$ the matrix entries are actually polynomials of degree at most $\sqrt{n}$. This is a key point for reducing the overall complexity estimate of the problem. Since the adjoint algorithm has a reversed flow, this key point does not seem to be relevant for Adjoint. For computing $\det A$ without divisions, Kaltofen's algorithm goes through the computation of $\det Z(z)$. Adjoint applied to $Z(z)$ computes A^* but does not seem to compute $Z^*(z)$ with the same complexity. In particular, differentiation of Step 3 using (6) leads to products $A^l(\delta B)^T$ that are more expensive over power series (one computes $A(z)^l(\delta B(z))^T$) than the initial computation in Det A^r ($A(z)^r$ on series).

For computing A^* without divisions only $Z^*(1)$ needs to be computed. We extend algorithm Adjoint with input $Z(z)$ by evaluating polynomials (truncated power series) partially. With a final evaluation at $z = 1$ in mind, a polynomial $p(z) = p_0 + p_1 z + \ldots + p_{n-1} z^{n-1} + p_n z^n$ may typically be replaced by

$(p_0 + p_1 + \ldots + p_m) + p_{m+1}x^{m+1} + \ldots + p_{n-1}z^{n-1} + p_n z^n$ as soon as any subsequent use of $p(z)$ will not require its coefficients of degree less than m.

4 Fast matrix product and application to polynomial matrices

We show how to integrate asymptotically fast matrix products in Algorithm Ajoint. On univariate polynomial matrices $A(z)$ with power series operations modulo z^n, Algorithm Adjoint leads to intermediary square matrix products where one of the operand has a degree much smaller than the other. In this case we show how to use fast rectangular matrix products [2, 6] for a (tiny) improvement of the complexity estimate of general polynomial matrix inversion.

Concluding remarks

Our understanding of the differentiation of Kaltofen's determinant algorithm has to be improved. We have proposed an implementation whose mathematical explanation remains to be given. Our work also has to be generalized to the block algorithm of [11].

Acknoledgements. We thank Erich Kaltofen who has brought reference [14] to our attention.

References

[1] W. Baur and V. Strassen. The complexity of partial derivatives. *Theor. Comp. Sc.*, 22:317–330, 1983.

[2] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. of Symbolic Computations*, 9(3):251–280, 1990.

[3] W. Eberly. Processor-efficient parallel matrix inversion over abstract fields: two extensions. In *Proc. Second International Symposium on Parallel Symbolic Computation, Maui, Hawaii, USA*, pages 38–45. ACM Press, Jul 1997.

[4] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann, and G. Villard. Faster inversion and other black box matrix computation using efficient block projections. In *Proc. International Symposium on Symbolic and Algebraic Computation, Waterloo, Canada*, pages 143–150. ACM Press, August 2007.

[5] M. Giesbrecht, W. Eberly, and G. Villard. Fast computations of integer determinants. In *The 6th International IMACS Conference on Applications of Computer Algebra, St. Petersburg, Russia*, June 2000.

[6] X. Huang and V.Y. Pan. Fast rectangular matrix multiplications and improving parallel matrix computations. In *Proc. Second International Symposium on Parallel Symbolic Computation, Maui, Hawaii, USA*, pages 11–23, Jul 1997.

[7] C.P. Jeannerod and G. Villard. Asymptotically fast polynomial matrix algorithms for multivariable systems. *Int. J. Control*, 79(11):1359–1367, 2006.

[8] E. Kaltofen. On computing determinants without divisions. In *International Symposium on Symbolic and Algebraic Computation, Berkeley, California USA*, pages 342–349. ACM Press, July 1992.

[9] E. Kaltofen and V.Y. Pan. Processor efficient parallel solution of linear systems over an abstract field. In *Proc. 3rd Annual ACM Symposium on Parallel Algorithms and Architecture*, pages 180–191. ACM-Press, 1991.

[10] E. Kaltofen and B.D. Saunders. On Wiedemann's method of solving sparse linear systems. In *Proc. AAECC-9*, LNCS 539, Springer Verlag, pages 29–38, 1991.

[11] E. Kaltofen and G. Villard. On the complexity of computing determinants. *Computational Complexity*, 13:91–130, 2004.

[12] S. Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors (in Finnish). Master's thesis, University of Helsinki, Dpt of Computer Science, 1970.

[13] S. Linnainmaa. Taylor expansion of the accumulated rounding errors. *BIT*, 16:146–160, 1976.

[14] G. M. Ostrowski, Ju. M. Wolin, and W. W. Borisow. Über die Berechnung von Ableitungen (in German). *Wissenschaftliche Zeitschrift der Technischen Hochschule für Chemie, Leuna-Merseburg*, 13(4):382–384, 1971.

[15] A. Storjohann. High-order lifting and integrality certification. *Journal of Symbolic Computation*, 36(3-4):613–648, 2003. Special issue International Symposium on Symbolic and Algebraic Computation (ISSAC'2002). Guest editors: M. Giusti & L. M. Pardo.

[16] A. Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4):609–650, 2005.

[17] V. Strassen. Vermeidung von Divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973.

[18] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transf. Inform. Theory*, IT-32:54–62, 1986.