Differentiation of Kaltofen’s division-free determinant algorithm
Gilles Villard

To cite this version:

HAL Id: ensl-00270753
https://ens-lyon.hal.science/ensl-00270753
Submitted on 7 Apr 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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.

1 Kaltofen’s determinant algorithm

Let $K$ be a commutative field. We consider $A \in K^{n \times n}$, $u \in K^{n \times 1}$, and $v \in K^{n \times 1}$. Kaltofen’s approach extends the Krylov-based methods of [13, 9, 10]. We introduce the Hankel matrix $H = (uA^{i+j-2}v)_{i,j} \in K^{n \times n}$, and let $h_k = uA^kv$ for $0 \leq k \leq 2n - 1$. 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 $r = \lfloor 2n/s \rfloor$ 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**

**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 adjoint algorithm**

The determinant of $A$ is a polynomial in $K[a_{11}, \ldots, a_{ij}, \ldots, a_{nn}]$ 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

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

The reverse mode of automatic differentiation (see [12, 13, 14]) allows to transform a program which computes $\Delta$ 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 $H = (uA^{i+j-2})_{ij}$ and $H_A = (uA^{i+j-1})_{ij}$. Then the determinant $f(0)$ is computed as

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

Viewing $\Delta$ as a function $\Delta_5$ of the $h_k$’s, we show that

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

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, $\Delta$ is seen as a function $\Delta_4$ 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 $\partial v_i$ be the $1 \times n$ vector, respectively the $n \times 1$ vector $\partial u_j$, whose entries are the derivatives of $\Delta_4$ with respect to the entries of $v_i$, respectively $u_j$. We show that

$$\begin{bmatrix} \partial v_0 \\ \partial v_1 \\ \vdots \\ \partial v_{r-1} \end{bmatrix} = H^u \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{s-1} \end{bmatrix} \quad (3)$$

and

$$\begin{bmatrix} \partial u_0, \partial u_1, \ldots, \partial u_{s-1} \end{bmatrix} = \begin{bmatrix} v_0, v_1, \ldots, v_{r-1} \end{bmatrix} H^u \quad (4)$$
where $H^s$ and $H^n$ are $r \times s$ matrices whose entries are selected $\partial \Delta_3 / \partial h_k$'s. Identities $\[9\]$ and $\[10\]$ 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 $\[4\]$, $\[5\]$ and $\[6\]$ 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 $\delta B$ from the $\delta u_i$’s. The matrix $\delta B$ gives the derivatives of $\Delta_3$ (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 $\delta B$ an $n \times n$ matrix $\delta A$ that gives the derivatives of $\Delta_2$ (the determinant seen as a function of $v_i$’s).

Then the differentiation of Step 1 computes from $\delta A$ and the $\delta v_i$’s an update of $\delta A$ 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)

$$q := p \times M$$

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 $\delta p$ and $\delta q$ of derivatives with respect to the entries of $p$ and $q$ are available. We also assume that an $n \times n$ matrix $\delta M$ that gives the derivatives with respect to the $m_{ij}$’s has been computed. We show that differentiating $\[4\]$ amounts to updating $\delta p$ and $\delta M$ as follows:

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

We see that the complexity is essentially preserved between $\[4\]$ 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 $\mathbb{R}$. 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 $\[17\]$ $\[18\]$. The determinant is a polynomial $\Delta$ 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 $A^T (\delta B)^T$ that are more expensive over power series (one computes $A(z)^T (\delta B(z))^T$) than the initial computation in Det $A^T (A(z)^r$ 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
\( (p_0 + p_1 + \ldots + p_m) + p_{m+1}x^{m+1} + \ldots + p_{n-1}z^{n-1} + p_nz^n \) 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 AJoint 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 \([11]\).

Acknowledgements. We thank Erich Kaltofen who has brought reference \([14]\) to our attention.

References


