Using an Efficient Sparse Minor Expansion Algorithm to Compute Polynomial Subresultants and the Greatest Common Denominator

MARTIN L. GRISS, MEMBER, IEEE

Abstract—**In this paper, the use of an efficient sparse minor expansion method to directly compute the subresultants needed for the greatest common denominator (GCD) of two polynomials is described. The sparse minor expansion method (applied either to Sylvester's or Bezout's matrix) naturally computes the coefficients of the subresultants in the order corresponding to a polynomial remainder sequence (PRS), avoiding wasteful recomputation as much as possible. It is suggested that this is an efficient method to compute the resultant and GCD of sparse polynomials.**

Index Terms—**Inners, minor expansion, polynomial GCD, subresultants, sparse matrices, sparse polynomials.**

I. Introduction

THE EMPIRICAL study of polynomial resultants by Ku and Adler [1], the discussions of the subresultant greatest common denominator (GCD) algorithm by Brown [2], [3], and the use of "inners" (subresultants) [4] for a multivariable polynomial greatest common factor (GCF) by Bose [5] suggest that the direct application of minor expansion to either the Sylvester or Bezout matrices is an efficient and interesting method of computing the GCD of two sparse polynomials.

It will be shown in this paper that the process of computing the subminors required in an efficient sparse minor algorithm [6], [7] quite easily and naturally yields the requisite subresultants and GCD. The crux lies in an appropriate reordering of either Sylvester's or Bezout's matrix, very similar to that studied in [7].

A number of authors have studied the problems of computing determinants of matrices with polynomial elements. When the polynomials are sparse (many zero coefficients in a full expansion), minor expansion is significantly more efficient than elimination methods [8], [9]. A particularly effective iterative algorithm [8] processes the rows of an $N \times N$ matrix in order, computing the minors of all of the $m \times m$ submatrices consisting of the first m rows and m of the N columns. This set of $m \times m$ minors is then combined (with due attention to sign) with the elements of the $m+1$st row to produce a new set of $(m+1) \times (m+1)$ minors.

When the matrix itself is sparse (many zero elements), a careful choice of the order in which the rows are processed can have a significant effect on the total storage and time required to compute the determinant. An efficient sparse minor expansion algorithm, using a sparse matrix representation and careful row ordering has been discussed by the author [6], [7].

Manuscript received June 3, 1977; revised February 2, 1978. This work was supported in part by the National Science Foundation under Grant MCS76-15035.

The author is with the Department of Computer Science, University of Utah, Salt Lake City, UT 84112.

II. Subresultants and Sylvester's Matrix

The method will be described using the notation of Brown and Traub [3], which we briefly review here. We are interested in computing the resultant or GCD of two multivariate polynomials, F and G. Expand F and G (assumed to be primitive) with respect to a "main variable," x:

$$F(x) = \sum_{i=0}^{\psi} f_i x^{\psi - i}, \; G(x) = \sum_{j=0}^{\gamma} g_i x^{\gamma - j}, \qquad \text{with } \psi \geq \gamma. \quad (1)$$

The coefficients $\{f_i, g_i\}$ are polynomials in other variables, and $\psi = \partial(F)$, $\gamma = \partial(G)$ are the degrees of the polynomials.

Brown and Traub define the subresultant, $S_j(F, G)$ as

$$S_j(F, G) = \det \begin{bmatrix} f_0 & f_1 & \cdots & f_{\gamma-j-1} & \cdots & f_{\psi+\gamma-2j-2} & x^{\gamma-j-1}F \\ & f_0 & \cdots & f_{\gamma-j-2} & & & x^{\gamma-j-2}F \\ & & & f_0 & & f_{\psi-j-1} & F \\ g_0 & g_1 & \cdots & g_{\psi-j-1} & & g_{\psi+\gamma-2j-2} & x^{\psi-j-1}G \\ & g_0 & \cdots & g_{\psi-j-2} & & & x^{\psi-j-2}G \\ & & & g_0 & & g_{\gamma-j-1} & G \end{bmatrix} \quad (2)$$

where $f_k = 0$ if $k > \psi$, $g_k = 0$ if $k > \gamma$, and $0 \leq j < \min(\psi, \gamma) = \gamma$. (This matrix is the transpose of that given in [3].)

Expanding the polynomials in the last column, it is shown that

$$S_j(F, G) = \sum_{l=0}^{j} s_{jl} x^{j-l}, \qquad \text{for } 0 \leq j < \min(\psi, \gamma) = \gamma. \quad (3)$$

The coefficients, s_{jl}, are determinants of the $(\psi + \gamma - 2j) * (\psi + \gamma - 2j)$ matrices, S_{jl}:

$$S_{jl} = \begin{bmatrix} \longleftarrow & (\psi+\gamma-2j) \text{ columns} & \longrightarrow \\ f_0 & f_1 & f_2 & \cdots & f_{\psi+\gamma-2j-2} & f_{\psi+\gamma-2j+l-1} \\ & f_0 & f_1 & \cdots & f_{\psi+\gamma-2j-3} & f_{\psi+\gamma-2j+l-2} \\ & & \vdots & & \vdots & \vdots \\ & & f_0 & \cdots & f_{\psi-j-1} & f_{\psi-j+l} \\ g_0 & g_1 & g_2 & \cdots & g_{\psi+\gamma-2j-2} & g_{\psi+\gamma-2j+l-1} \\ & g_0 & g_1 & \cdots & g_{\psi+\gamma-2j-3} & g_{\psi+\gamma-2j+l-2} \\ & & \vdots & & \vdots & \vdots \\ & & g_0 & \cdots & g_{\psi-j-1} & g_{\gamma-j+l} \end{bmatrix} \begin{matrix} \uparrow \\ (\gamma-j) \\ \text{rows} \\ \downarrow \\ \uparrow \\ (\psi-j) \\ \text{rows} \\ \downarrow \end{matrix} \quad (4)$$

where g_k and f_k are 0 if k is out of range, and $0 \le j < \min(\psi, \gamma) = \gamma$, $0 \le l \le j$.

Now S_{oo} is, of course, Sylvester's matrix, and $\det S_{oo} = s_{oo}$ is the resultant $R(F, G)$. The matrices S_{jl} are submatrices of Sylvester's matrix, S_{oo}, hence the term *subresultant* for $S_j(F, G)$.

The fundamental theorem presented by Brown and Traub [3] is the relation between a polynomial remainder sequence (PRS) $\{F_i\}$, generated from a pair of polynomials F_1 and F_2, and the subresultants $S_j(F_1, F_2)$, for $0 \le j < \partial(F_2)$.

For each j, $0 \le j < \partial(F_2)$, either $S_j(F_1, F_2) \sim F_i$ for some F_i, or $S_j(F_1, F_2) = 0$. Hence, since GCD $(F_1, F_2) \sim F_k$ for the "last" polynomial F_k in the PRS, GCD $(F_1, F_2) \sim S_j(F_1, F_2)$ for some j. (Here "$\sim$" is similarity and j is determined such that $S_k(F, G) \equiv 0$ for $k < j$.) Further details, and the exact similarity coefficients can be found in [3].

If we now reorder the rows of S_{oo} as suggested in [7], we obtain the matrix

$$\tilde{S}_{oo} = \begin{bmatrix}
g_0 & g_1 & g_2 & \cdots & & g_\gamma & & & & & \\
 & g_0 & g_1 & \cdots & & g_{\gamma-1} & g_\gamma & & & & \\
 & & \ddots & g_0 & & \cdots & & g_\gamma & & & \\
f_0 & f_1 & & \cdots & & & \cdots & & f_\psi & & \\
 & & & & g_0 & & \cdots & & g_\gamma & & \\
 & f_0 & & \cdots & & & & \cdots & & f_\psi & \\
 & & \ddots & & & g_0 & & \cdots & & g_\gamma & \\
 & & & f_0 & & & & & & \ddots & \\
 & & & & & & & & & & f_\psi \\
 & & & & & & g_0 & & \cdots & & g_\gamma
\end{bmatrix}
\begin{matrix} \uparrow \\ (\psi-\gamma) \\ \text{base rows} \\ \downarrow \\ \uparrow \\ \\ \gamma \text{ pairs} \\ \\ \\ \downarrow \end{matrix} \tag{5}$$

This is a "good" order for efficient computation using the sparse minor expansion algorithm. We apply this algorithm first to the "base" set of rows of G coefficients (if $\psi \neq \gamma$), to produce a set of $(\psi - \gamma) * (\psi - \gamma)$ minors. At each successive step, $k = 1, 2, \cdots, \gamma$, we merge in a "shifted" pair of F/G rows,

$$\begin{pmatrix} 0\,0\,0 & f_0 & f_1 & \cdots & f_\psi \\ 0\,0\,0 & 0 & g_0 & \cdots & g_\gamma \end{pmatrix} \tag{6}$$

to obtain a new set of $(\psi - \gamma + 2k) * (\psi - \gamma + 2k)$ minors. The subresultant coefficients, s_{jl}, will be among the minors in the set for $k = \gamma - j$, $j = 0, \cdots, \gamma - 1$. If at any step $k + 1$, the new set of (nonzero) minors is empty, the coefficients of the GCD (F, G) are among the kth set, and thus,

$$\text{GCD}(F, G) \sim \sum_{l=0}^{\gamma-k} s_{\gamma-k,l}\, x^l \tag{7}$$

[in fact, GCD $(F, G) = pp(S_j(F, G))$].

The minor set at step k will also include additional minors, required to compute the new $k + 1$st set. The most interesting point is that this method will compute all of the subresultant coefficients required, sharing as much as possible the computation of common subminors, and generates these coefficients in the order required to compute most easily the GCD. As suggested by Brown [2] and Ku and Adler [1], this minor expansion method should be very effective for sparse polynomials and, of course, Sylvester's matrix is rather sparse and structured, motivating the use of the sparse minor expansion algorithm. The matrix would be even more sparse than indicated above, since certain of the $\{f_i\}$ and $\{g_i\}$ coefficients can be zero as well.

III. Use of the Sparse Minor Expansion Method

As in [7] the matrix will be converted to a sparse form, a list of sparse rows. Each column (c) will be identified by its "key" 2^c. The essence of the sparse minor expansion algorithm is as follows:

1) Each sparse row is represented as a list

$$\text{SROW} = ((2^{j_1} \cdot f_{j_1})(2^{j_2} \cdot f_{j_2}) \cdots (2^{j_k} \cdot f_{j_k}))$$

where the $\{f_{j_i}\}$ are the nonzero elements in this row and 2^{j_i} is the associated key;

2) An $m \times m$ minor, consisting of the first m rows and columns $(j_1, j_2, \cdots, j_m)$, is identified by a key $= 2^{j_1} + 2^{j_2} + \cdots + 2^{j_m}$;

3) A set of $m \times m$ minors will also be represented as a list:

$$\text{DLIST} = ((\text{KEY}_1 \cdot D_1)(\text{KEY}_2 \cdot D_2) \cdots (\text{KEY}_n \cdot D_n))$$

(Thus, a sparse row can be thought of as a set of 1×1 minors.)

4) Minor expansion is accomplished by "merging" an "old" set of $m \times m$ minors (DOLD) and the $m + 1$st row (SROW) to produce a new set (DNEW) of $(m + 1) \times (m + 1)$ minors, using a function DMRG1:

$$\text{DNEW} \leftarrow \text{DMRG1}(\text{DOLD}, \text{SROW});$$

DMRG1 simply collects contributions to an $(m + 1) \times (m + 1)$ minor D, with Key $= 2^{j_1} + 2^{j_2} + \cdots + 2^{j_{m+1}}$, from each $m \times m$ minor D_k and row element f_c, such that $\text{Key}_k + 2^c = \text{Key}$; c is clearly one of the $\{j_i\}$. The sign of the contribution is obtained from the number of transpositions of $[j_1, \cdots, j_{m+1}, c]$ to obtain the order $j_1 < j_2 \cdots j_{k-1} < c < j_{k+1} < \cdots < j_{m+1}$. The details of an efficient implementation of DMRG1 are given in [7].

F and G will be written as ordered sparse rows:

$$F = ((2^0 \cdot f_0)(2^1 \cdot f_1) \cdots (2^\psi \cdot f_\psi))$$
$$G = ((2^0 \cdot g_0) \cdots (2^\gamma \cdot g_\gamma)) \tag{8}$$

representing polynomials with coefficients in other variables (and the main variable x suppressed), $lc(F) = f_0$, $\partial(F) = \psi$.

As can be seen by examining (3) and (4), S_{jl} is the minor of a submatrix of $\tilde{S}_{oo}$, composed of the $(\psi + \gamma - 2j - 1)$ columns $0 \cdots \psi + \gamma - 2j - 2$, and a column $q = (l + \psi + \gamma - 2j - 1)$, $l = 0 \cdots j$ selected from the remaining columns. We

identify s_{jl} by the key

$$\begin{aligned}\text{KEY}(s_{jl}) &= \text{KEY}(0, 1, \cdots, \psi+\gamma-2j-2,\\ &\qquad l+\psi+\gamma-2j-1)\\ &= 2^0+2^1+\cdots+2^{x-1}+2^{x+l}\\ &= (2^x-1)+2^x\cdot 2^l,\\ &\qquad l=0\cdots j \text{ with } x=\psi+\gamma-2j-1. \quad (9)\end{aligned}$$

Recall that $j=\gamma-k$, $k=1\cdots\gamma$ so that

$$\text{KEY}(s_{jl}) = (B_k-1)+B_k * 2^l$$

where

$$\begin{aligned}B_k &= 2^{\psi+\gamma+2k-2\gamma-1}\\ &= 2^{\psi-\gamma+2k-1} = 4 * B_{k-1}. \quad (10)\end{aligned}$$

Now $B_0 \equiv 2^{\psi-\gamma-1}$ and is computed while computing the "base" set:

```
B := 1/2;
SET := NIL;
FOR I = 1: (ψ − γ) DO
    ⟨⟨SET := DMRG1 (SET, G);
      G := SHIFT (G, 2);
      B := 2 * B⟩⟩;
```

where SHIFT $(G, 2)$ simply "shifts" all keys of G by multiplying by 2. Now the main loop begins:

```
    J := γ − 1;
STEP: NSET := DMRG1 (DMRG1 (SET, F), G));
    IF NSET = NIL THEN GOTO DONE;
    SET := NSET; B := 4 * B;
    IF J = 0 THEN GOTO DONE;
    F := SHIFT (F, 2); G := SHIFT (G, 2);
    J := J − 1;
    GOTO STEP;
```

Finally we compute the coefficients s_{jl}:

```
DONE: B1 := B − 1;
    GCD := NIL;
    FOR L := 0: J DO
      ⟨⟨GCD := LOOKUP (B1 + B, SET) · GCD;
        B := 2 * B⟩⟩;
    GCD := REVERSE GCD;
```

which computes the GCD in a nonsparse form. The actual program is given in the Appendix, in MODE REDUCE [7], [10]. (MODE REDUCE is a computer algebra system with a Pascal-like type definition facility.)

In order to find the GCD of nonprimitive polynomials, the above algorithm will be applied recursively to the coefficients, to obtain GCD (cont (F_1), cont (F_2)), as suggested in [5], so that finally,

$$\begin{aligned}&\text{GCD}(F_1, F_2)\\ &\quad = \text{GCD}(pp(F_1), pp(F_2))*\text{GCD}(\text{cont}(F_1), \text{cont}(F_2)) \quad (11)\end{aligned}$$

Once the "base" set has been computed, subsequent sets of minors always involve a joint shift of F and G:

```
NSET ← DMRG1 (DMRG1 (SET, F), G));
F ← SHIFT (F, 2); G ← SHIFT (G, 2);        (12)
```

If we first make up a set of 2×2 minors, $FG \leftarrow$ DMRG1 (F, G), and generalize DMRG1 to accept 2 sets of minors (as suggested in [7]), we find a simpler form

```
NSET ← DMRG2 (SET, FG);
FG ← SHIFT (FG, 2);                        (13)
```

where the 2×2 minors in FG are therefore computed only once, saving repeated computation.

It is interesting to note that the elements of Bezout's matrix (a $\psi\times\psi$ matrix derived from Sylvester's matrix) are simple sums of the 2×2 minors in FG. This suggests that the above method be directly applied to Bezout's matrix, producing an even more efficient method, discussed in the next section.

IV. Bezout's Matrix

The following $\psi\times\psi$ matrix B (known as Bezout's matrix) has been used by Ku and Adler [1] to compute the resultant, $S(F, G) = \det(B)$:

$$B = \begin{bmatrix} g_0 & g_1 & & g_2 & \cdots & g_\gamma & & \\ & g_0 & & g_1 & \cdots & & g_\gamma & \\ & & \ddots & & & & & \\ & & & g_0 & g_1 & \cdots & g_\gamma & \\ A_{11} & A_{12} & & & A_{13} & \cdots & A_{1\psi} & \\ \vdots & & & & & & & \\ A_{\gamma 1} & A_{\gamma 2} & & & & \cdots & A_{\gamma\psi} & \end{bmatrix} \begin{matrix} \uparrow \\ \\ \\ \\ \\ \\ \downarrow \end{matrix} \begin{matrix} \psi-\gamma \text{ base rows} \\ \\ \\ \\ \gamma \text{ rows} \\ \\ \\ \end{matrix} \quad (14)$$

The A_{ij} are sums of 2×2 minors, D_{ij},

$$D_{ij} = f_i g_j - f_j g_i \quad (15)$$

(where f_i or $g_i = 0$ if i is out of range); and

$$A_{ij} = \sum_{k=0}^{w} D_{k,i+k} \quad (16)$$

with

$$w = \min(i-j, \psi-i-1, j-1, \psi-j-1).$$

The A_{ij} are obtained from the coefficients of x in a homogeneous system of equations [1], just as Sylvester's matrix is obtained from the system of equations

$$\begin{bmatrix} x^{\psi-1}G(x)=0 \\ x^{\psi-2}G(x)=0 \\ \vdots \\ x^0 G(x)=0 \\ x^{\gamma-1}F(x)=0 \\ \vdots \\ x^0 F(x)=0 \end{bmatrix}. \quad (17)$$

We first define partial sums ("head" and "tail" polynomials) of $F(x)$ and $G(x)$:

$$F_k(x) = \sum_{i=0}^{k} f_i x^{k-i}, \qquad \bar{F}_k(x) = \sum_{i=k+1}^{\psi} f_i x^{\psi-i}$$

$$G_k(x) = \sum_{i=0}^{k} g_i x^{k-i}, \qquad \bar{G}_k(x) = \sum_{i=k+1}^{\gamma} g_i x^{\gamma-i} \tag{18}$$

so that

$$\begin{aligned} F(x) &= F_\psi(x) = \bar{F}_{-1}(x) \\ &= F_k(x)\cdot x^{\psi-k} + \bar{F}_k(x), \qquad k = 0 \cdots \psi \\ G(x) &= G_\gamma(x) = \bar{G}_{-1}(x) \\ &= G_k(x)\cdot x^{\gamma-k} + \bar{G}_k(x), \qquad k = 0 \cdots \gamma. \end{aligned} \tag{19}$$

We now combine γ pairs of equations $x^i F(x) = 0$, $x^{\psi-\gamma+i}G(x) = 0$ as $i = 0 \cdots \gamma - 1$:

$$\begin{bmatrix} x^{\psi-\gamma}G(x) = 0 \\ \vdots \\ x^0 G(x) = 0 \\ F_0(x)\bar{G}_0(x) - G_0(x)\bar{F}_0(x) = 0 \\ \vdots \\ F_{\gamma-1}(x)\bar{G}_{\gamma-1}(x) - G_{\gamma-1}(x)\bar{F}_{\gamma-1}(x) = 0 \end{bmatrix} \tag{21}$$

given in [1].

This derivation shows that if the columns of Sylvester's matrix are labeled with powers of x, $[x^{\psi+\gamma-1}, x^{\psi+\gamma-2}, \cdots, x^0]$ [corresponding to the system of equations in (17)], the transformation (19) corresponds to simple row operations on (say) $x^i F(x)$, adding multiples of other rows. This can be used to show the equivalence of det (S_{oo}) and det (B).

We can now show rather simply that the determinants of the matrices S_{jl} (selected from the first $\psi - \gamma + 2(\gamma - j)$ rows of $\tilde{S}_{oo}$) are identical to a similar process applied to the first $\psi - \gamma + (\gamma - j)$ rows of Bezout's matrix B. These are smaller minors, and the process will therefore be more efficient than direct application to Sylvester's matrix as in Section III.

We permute the first $\psi - \gamma + 2(\gamma - j)$ rows of $\tilde{S}_{oo}$ to correctly pair the $x^i F(x)$ and $x^{\psi-\gamma+i}G(x)$ rows:

$$M_j = \begin{array}{c} \\ F_0 \\ G_0 \\ F_1 \\ G_1 \\ \vdots \\ F_{\gamma-1} \\ G_{\gamma-1} \end{array} \left[\begin{array}{cccc|cc|ccccc} & & & & g_0 & g_1 & \cdots & & g_\gamma & & \\ & & & & & g_0 & g_1 & \cdots & & g_\gamma & \\ & & & & & & \ddots & & & & \\ & & & & & & g_0 & \cdots & & g_\gamma & \\ & & & f_0 & f_1 & & \cdots & & f_\gamma & \cdots & f_\psi \\ & & & g_0 & g_1 & & \cdots & & g_\gamma & & \\ & & \iddots & & & & & f_\gamma & & & f_\psi \\ & & & & & & & g_\gamma & & & \\ & \iddots & & & & & \iddots & & & & \\ f_0 & f_1 & & & \cdots & & & f_\gamma & & & f_\psi \\ g_0 & g_1 & & & & & & g_\gamma & & & \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{l} \uparrow \\ \\ \\ \psi-\gamma \\ \text{rows} \\ \\ \\ \\ \uparrow \\ 2(\gamma-j) \\ \text{rows} \\ \downarrow \end{array} \tag{22}$$

excess columns

We now eliminate all $\{F_i\}$ coefficients in the "excess columns" of the F rows, by performing the row operations indicated in equation (19), using the F_i and G_i row "labels":

$$\text{row } x^i F \to x^i F \cdot G_i(x) - F_i(x)\cdot x^{i+\psi-\gamma}G(x). \tag{23}$$

This multiplies each F row by g_0, and adds (or subtracts) a multiple of other rows. Thus, det (M_j) is increased by $g_0^{\gamma-j}$. M_j now contains the principal segment of B, and additional rows of G:

$$\begin{aligned} [x^i F(x)]G_i(x) - [x^{\psi-\gamma+i}G(x)]F_i(x) &= 0 \\ &= x^i G_i(x)[F_i(x)x^{\psi-i} + \bar{F}_i(x)] \\ &\quad - x^{\psi-\gamma+i}F_i[G_i(x)x^{\gamma-i} + \bar{G}_i(x)] \\ &= x^i[G_i(x)F_i(x) - x^{\psi-\gamma}\bar{G}_i(x)F_i(x)] \end{aligned} \tag{20}$$

which eliminates all powers of x greater than $x^{\psi-\gamma-1}$. This leads to the ψ equations:

$$M_j = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \left[\begin{array}{cccc|ccccc} & & & & g_0 & g_1 & \cdots & g_\gamma & & \\ & & & & & g_0 & g_1 & \cdots & g_\gamma & \\ & & & & & & \ddots & & & \\ & & & & & & g_0 & \cdots & g_\gamma & \\ & & & 0 & D_{01} & D_{02} & \cdots & & D_{0\psi} & \\ & & & g_0 & g_1 & \cdots & & g_\gamma & & \\ & & 0 & 0 & D_{02} & (D_{03}+D_{12}) & & \cdots & & D_{i\psi} \\ & & g_0 & g_1 & g_2 & & & & & \\ & \iddots & & & & & & & & \\ 0 & 0 & \cdots & 0 & D_{0\gamma} & & & \cdots & & D_{\gamma\psi} \\ g_0 & g_1 & & \cdots & & & & g_\gamma & & \end{array}\right] \tag{24}$$

$$\begin{array}{c} \longleftarrow l \longrightarrow \\ 0 \cdots\cdots\cdots j \end{array}$$

Each of the determinants s_{jl} now corresponds to a determinant of M_j, with l selecting one of the last j columns. The excess triangular portion of G rows can be replaced by 1 (after dividing by the $g_0^{\gamma-j}$ factors) so that the determinants s_{jl} are obtained from the matrix

$$B_j = \begin{bmatrix} g_0 & g_1 & \cdots & g_\gamma & 0 & \cdots & 0 \\ & g_0 & \cdots & & g_\gamma & & \\ & & \ddots & & & \ddots & \\ & & & g_0 & & & g_\gamma \\ D_{01} & & & \cdots & & & D_{0\psi} \\ D_{02} & (D_{03}+D_{12}) & & & \cdots & & D_{1\psi} \\ \vdots & & & & & & \vdots \\ D_{0\gamma} & \cdots & & & & & \end{bmatrix} \tag{25}$$

applying the stepwise method of Section III to Bezout's matrix B. A program can be easily devised to compute the A_{ij} elements as the minors are generated in a stepwise fashion.

IV. Conclusions

The application of the efficient sparse minor expansion algorithm to the direct computation of the GCD via generation of all requisite subresultants is quite simple and efficient.

I am grateful for many enjoyable discussions on this subject with D. Yun, M. Rothstein, and S. Brown. The method can be applied either to Sylvester's matrix or to Bezout's reduced matrix, leading to an even more efficient method. When the input polynomials are sparse, Sylvester's matrix and even Bezout's matrix will be rather sparse, clearly motivating the use of a *sparse* minor expansion method.

Appendix
Use of the Efficient Sparse Minor Algorithm to Compute GCD

```
Load SPARSE MINOR  package (ref. 7) to define sparse minor
data structures and procedures:

MODE  ELEM =POLYNOMIAL,  EROW=LISTOF ELEM;     Dense row
MODE  DELEM=STRUCT(KEY:INTEGER,DETM:ELEM),
      DROW =LISTOF DELEM;                      Sparse row

Procedures:   DMRG1(drow,drow)->drow           Minor expansion
              SHIFT(drow,integer)->drow        Shift  columns
```

```
ELEM PROCEDURE LOOKUP(K,D);
% Lookup K'th entry in minor list D;
   DCL K:INTEGER, D:DROW;
   IF NULL D THEN 0
    ELSE IF KEY(HEAD D)=K THEN DETM(HEAD D)
    ELSE LOOKUP(K,TAIL D);

EROW PROCEDURE EGCD(E1,E2);
% Generate minors in "good-order" to obtain Subresultants or  GCD;
   BEGIN
        DCL PHI,GAMMA,M,B1,B,J,L:INTEGER,   E,GCD:EROW,
            F,G,SET,NSET:DROW;
  % Ensure PHI>=GAMMA;
        PHI:=LENGTH E1 -1; GAMMA:=LENGTH E2-1;
        IF GAMMA > PHI THEN <<M:=PHI; PHI:=GAMMA; GAMMA:=M;
                              E:=E1; E1:=E2; E2:=E>>;
  % Convert to sparse rows;
        G:=DROW E1; F:=DROW E2;
        M:=PHI-GAMMA;
  % Block of PHI-GAMMA "short" rows;
        IF M>0 THEN  <<SET:=F; M:=M-1; B:=1;
                       FOR I:=1:M DO  <<F:=SHIFT(F,2);
                                       B:=2*B;
                                       SET:=DMRG1(SET,F)>>;
                       F:=SHIFT(F,2) >>
         ELSE <<SET:=DMRG1(F,G); GAMMA:=GAMMA-1; B:=2;
                G:=SHIFT(G,2);   F:=SHIFT(F,2)>>;
        J:=GAMMA;
   STEP:        % Merge successive rows, until NSET is empty;
        NSET:=DMRG1(DMRG1(SET,F),G);
        IF NULL NSET THEN GOTO DONE;
        SET:=NSET;
        IF J=0 THEN GOTO DONE;
  % Shift columns by 1 place;
        G:=SHIFT(G,2); F:=SHIFT(F,2);
        B:=4*B; J:=J-1;
        GOTO STEP;
   DONE:        % Produce list of subresultants (or GCD);
        B1:=B-1;
        GCD:=NIL;
        FOR L:=0:J DO
            <<GCD:=LOOKUP(B1+B,SET) . GCD;
              B:=2*B>>;
        RETURN REVERSE GCD
   END;
```

REFERENCES

[1] S. Y. Ku and R. J. Adler, "Computing polynomial resultants: Bezout's determinant vs. Collins' reduced PRS algorithm," *Commun. Ass. Comput. Mach.*, vol. 12, p. 23, 1969.

[2] W. S. Brown, "On the subresultant PRS algorithm," presented at SYMSAC 76, Yorktown Heights, 1976. Bell Lab. Comput. Sci. Tech. Rep. 56, 1976. (Submitted to Ass. Comput. Mach. TOMS.)

[3] W. S. Brown and J. F. Traub, "On Euclids algorithm and the theory of subresultants," *J. Ass. Comput. Mach.*, vol. 18, p. 505, 1971.

[4] E. I. Jury, "The theory and applications of the inners," *Proc. IEEE*, vol. 63, pp. 1044–1068,

[5] N. K. Bose, "An algorithm for GCF extraction from two multivariable polynomials," *Proc. IEEE* (Letters), pp. 185–186, Jan. 1976.

[6] M. L. Griss, "The algebraic solution of sparse linear systems via minor expansion," *Ass. Comput. Mach. TOMS*, vol. 2, p. 39, 1976.

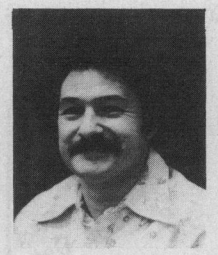
[7] ——, "An efficient sparse minor expansion algorithm," *Proc. Ass. Comput. Mach. 76*, p. 429, 1976.

[8] W. M. Gentleman and S. C. Johnson, "Analysis of algorithms, a case study: Determinants of polynomials," *Ass. Comput. Mach. TOMS*, vol. 2, p. 232, 1976.

[9] E. Horowitz and S. Sahni, "On computing the exact determinant of matrices with polynomial entries," *J. Ass. Comput. Mach.*, vol. 22, p. 38, 1975.

[10] A. C. Hearn, "A mode analyzing algebraic manipulation program," *Proc. Ass. Comput. Mach. 74*, p. 722, 1974.

[11] M. L. Griss, "The definition and use of data-structures in REDUCE," *Proc. SYMSAC 76, Ass. Comput. Mach.*, p. 53, 1976.



Martin L. Griss (M'76) was born in Capetown, South Africa, on September 4, 1944. He received the B.Sc. degree in mathematical physics at the Technion, Haifa, Israel, in 1967, and the Ph.D. degree in theoretical physics at the University of Illinois, Urbana, IL, in 1971.

He spent two years at the California Institute of Technology, Pasadena, CA, as a Postdoctoral Fellow in Theoretical Physics, and two years with the Utah Computational Physics Group as a Research Assistant Professor in the Department of Physics. He is currently an Assistant Professor in the Department of Computer Science, University of Utah, Salt Lake City, UT. He has been involved with mathematical software and computational methods since 1963. Current research involves computer algebra and portable symbol manipulation systems. Areas of interest include sparse matrices, general computational methods, and portable programming language design and implementation.

Dr. Griss is a member of the Association for Computing Machinery (SIGPLAN, SIGNUM, SIGSAM, SIGSOFT, SIGMINI, and SIGMICRO).