Two Resultant Based Methods Computing the Greatest Common Divisor of Two Polynomials

D. Triantafyllou and M. Mitrouli

Department of Mathematics, University of Athens,
Panepistimiopolis 15784, Athens, Greece
dtriant@math.uoa.gr, mmitroul@cc.uoa.gr

Abstract. In this paper we develop two resultant based methods for the computation of the Greatest Common Divisor (GCD) of two polynomials. Let S be the resultant Sylvester matrix of the two polynomials. We modified matrix S to S^*, such that the rows with non-zero elements under the main diagonal, at every column, to be gathered together. We constructed modified versions of the LU and QR procedures which require only the $\frac{1}{3}$ of floating point operations than the operations performed in the general LU and QR algorithms. Finally, we give a bound for the error matrix which arises if we perform Gaussian elimination with partial pivoting to S^*. Both methods are tested for several sets of polynomials and tables summarizing all the achieved results are given.

Keywords: Greater Common Divisor, Sylvester matrix, Gaussian elimination, QR factorization.

1 Introduction

The computation of the greatest common divisor (GCD) of two or more polynomials is one of the most frequent problems in several fields such as numerical analysis, linear and numerical linear algebra, control theory, matrix theory, statistics etc. Many numerical algorithms have been created to solve this problem [1],[5]. We can classify these algorithms in two categories [4] :

- Numerical methods that are based on Euclid's algorithm.
- Numerical methods that are based on procedures involving matrices.

A resultant based computation of the greatest common divisor of two polynomials belongs to the second of the above categories. We can use either non-orthogonal or orthogonal algorithms in order to find the GCD. Non-orthogonal algorithms are faster but we can not prove their stability in contrast to orthogonal algorithms that are slower but stable. In practice there are only few cases in which non-orthogonal algorithms give wrong results. Except for the previous dilemma there is another one : We know that during numerical operations on a floating point arithmetic rounding off errors (catastrophic cancellation) are

Z. Li et al. (Eds.): NAA 2004, LNCS 3401, pp. 519–526, 2005.

caused. On this account we introduce a numerical accuracy as we will see below. Finally there are also methods which compute the approximate GCD.

Let R[s] be the ring of real polynomials [3], $\Re^{m\times n}$ the set of all mxn real matrices, r(A) the rank of the matrix $A\epsilon\Re^{m\times n}$ and $\vartheta\{f(s)\}$ the degree of a polynomial. Consider two polynomials a(s),b(s)$\epsilon R[s]$, $\vartheta\{a(s)\}$=m, where a(s) is a monic polynomial and $\vartheta\{b(s)\}$=n, with $n \leq$m,where [1]

$$a(s) = s^m + a_{m-1}s^{m-1} + ... + a_1 s + a_0$$
$$b(s) = b_n s^n + b_{n-1}s^{n-1} + ... + b_1 s + b_0$$

We define the resultant S(a,b) of the two polynomials by

$$S(a,b) = \begin{bmatrix} 1 & a_{m-1} & a_{m-2} & \dots & \dots & \dots & \dots & a_0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1 & a_{m-1} & a_{m-2} & \dots & \dots & \dots & \dots & a_0 & 0 & \dots & 0 & 0 \\ . & . & . & \dots & \dots & \dots & \dots & \dots & . & \dots & \dots & . & . \\ 0 & 0 & 0 & \dots & 1 & a_{m-1} & . & . & . & . & \dots & a_1 & a_0 \\ - & - & - & - & - & - & - & - & - & - & - & - & - \\ b_n & b_{n-1} & b_{n-2} & \dots & \dots & b_0 & 0 & \dots & \dots & \dots & \dots & 0 & 0 \\ 0 & b_n & b_{n-1} & \dots & \dots & \dots & b_0 & 0 & \dots & \dots & \dots & 0 & 0 \\ . & . & . & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & . & . \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & b_n & b_{n-1} & \dots & \dots & b_1 & b_0 \end{bmatrix}$$

Obviously $S(a,b)\epsilon\Re^{(n+m)\times(n+m)}$. Applying elementary row operations with scalars from $\Re$ to matrix S(a,b) we transform it to an upper triangular form (this can be managed to using e.g. gaussian or orthogonal transformations). The existence, the degree and the coefficients of the GCD are arising from the following theorem :

Theorem 1. *[3] Let a(s),b(s)$\in$R[s], $\vartheta\{a(s)\}$=m, $\vartheta\{b(s)\}$=n,m $\geq$n and let* $z(s) = s^r + z_{r-1}s^{r-1} + ... + z_1 s + z_0$ *be the GCD. The following properties hold true:*

(i) (a(s),b(s)) are coprime, if and only if rank(S(a,b))=n+m.
(ii) r=$\vartheta\{z(s)\}$=m+n-rank(S(a,b)).
(iii) If S_H(a,b) is the row echelon form of S(a,b), then the last non-vanishing row gives the coefficients of GCD of (a(s),b(s)). □

In the previous matrix S we observe that its first column has only two non-zero elements. We interchange the second and the (n+1)-th row in order to collect the two non-zero elements of the first column in the two first rows of S. Afterwards only three elements of the second column of S are non-zero. Interchanging the corresponding rows, we collect them to the three first rows of S. The number of non-zero elements under the diagonal will be increasing per one at every column until the deg{b(x)}-1 column. From the next column and until the m+n-deg{b(x)} column the number of non-zero elements is constant and equal to deg{b(x)}+1. Finally at the remaining deg{b(x)}-1 columns the number of non-zero elements will be decreasing per one. With the corresponding row interchanges we collect at every column all those rows that have non-zero entries under the diagonal.

The modified Sylvester matrix will have the following form:

$$S^*(a,b) = \begin{bmatrix}
1 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_{m-n} & a_{m-n-1} & a_{m-n-2} & \cdots & a_0 & 0 & 0 & \cdots & 0 \\
b_n & b_{n-1} & b_{n-2} & b_{n-3} & \cdots & b_0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 1 & a_{m-1} & a_{m-2} & \cdots & a_{m-n+1} & a_{m-n} & a_{m-n-1} & \cdots & a_1 & a_0 & 0 & \cdots & 0 \\
0 & b_n & b_{n-1} & b_{n-2} & \cdots & b_1 & b_0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 1 & a_{m-1} & \cdots & a_{m-n+2} & a_{m-n+1} & a_{m-n} & \cdots & a_2 & a_1 & a_0 & \cdots & 0 \\
0 & 0 & b_n & b_{n-1} & \cdots & b_2 & b_1 & b_0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
. & . & . & . & \cdots & . & . & . & \cdots & . & . & . & \cdots & . \\
0 & 0 & 0 & 0 & \cdots & 0 & 1 & a_{m-1} & \cdots & a_{m-n+1} & a_{m-n} & a_{m-n-1} & \cdots & a_0 \\
0 & 0 & 0 & 0 & \cdots & 0 & b_n & b_{n-1} & \cdots & b_1 & b_0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & b_n & \cdots & b_2 & b_1 & b_0 & \cdots & 0 \\
. & . & . & . & \cdots & . & . & . & \cdots & . & . & . & \cdots & . \\
. & . & . & . & \cdots & . & . & . & \cdots & . & . & . & \cdots & . \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & b_n & \cdots & b_0
\end{bmatrix}$$

2 The Resultant LU Method

Corollary 1. *Let S(a,b) be the resultant matrix of the pair of polynomials (a(s), b(s)), ρ=rank(S(a,b)) and let S_1(a,b) denote the upper triangular form of S(a,b) obtained under the Gauss row transformations [3], i.e.*

$$S_1(a,b) = \begin{bmatrix}
x & x & \ldots & \ldots & x & \ldots & \ldots & x \\
0 & x & \ldots & \ldots & x & \ldots & \ldots & x \\
. & . & \ldots & \ldots & . & \ldots & \ldots & . \\
. & . & \ldots & \ldots & . & \ldots & \ldots & . \\
. & . & \ldots & \ldots & . & \ldots & \ldots & . \\
0 & 0 & \ldots & \ldots & x & \ldots & \ldots & x \\
0 & 0 & \ldots & \ldots & 0 & \ldots & \ldots & 0 \\
. & . & \ldots & \ldots & . & \ldots & \ldots & . \\
. & . & \ldots & \ldots & . & \ldots & \ldots & . \\
0 & 0 & \ldots & \ldots & 0 & \ldots & \ldots & 0
\end{bmatrix}$$

where the x leading element of each nonzero row is also nonzero. The nonzero elements of the last nonzero row of S_1(a,b) define the coefficients of GCD in reverse order. □

These results hold for the modified matrix S^* too.

As we have seen, the Sylvester matrix S has a specific form. We can take advantage of this form, during the procedure of triangulation. More particular, because the Sylvester matrix has many zeros in many columns, we can zero only the non-zero elements of S. After we produce the Sylvester matrix, we know exactly how many non-zero elements has each column of S. So, we transform the Sylvester matrix S to the modified matrix S^*. Then the nullification of the elements at k step in column k, is not necessary to take place to the whole sub-column (i=k+1:m+n), but only to the first s elements (i=k+1:k+s) since the other elements are already zero (where s is the number of non-zero elements under the diagonal)(i=r:s denotes i=r,. . . , s). The required row interchanges do not affect the final result, the greater common divisor. After we zero the elements under the diagonal at stage k, we must update the elements in submatrix $S^*_{k+1:m+n,k+1:m+n}$. But the big advantage is that at row k the non-zero elements are until the column $[\frac{k+1}{2}]$+deg{a(x)} for the first 2·deg{b(x)} rows and until the deg{a(x)}+deg{b(x)} column for the last ones. So the part of the matrix that we must update is only the submatrix k+1:s,k+1:P_k at k-stage, where the element P_k of matrix P is the number of the column, that after it we have only zeros at row k.

The number of non-zero elements under the diagonal will be increasing per one at every step until the deg{b(x)}-1 step. From the next step and until deg{a(x)} step the number of non-zero elements is constant and equal to deg{b(x)}. Finally at the remaining deg{b(x)}-1 steps the number of non-zero elements will be decreasing per one at every step. We mention that because of the special form of S^* only few elements (which are adjoining) are needed to be zero at every step. Also not the entire (m+n-k+1)x(m+n-k+1) submatrix at stage k must be updated. These two observations will decrease a lot the complexity.

So, we can also modify the LU Factorization with considerable reduction of flops. It follows a non-orthogonal algorithm using Gaussian transformation (with partial pivoting to improve numerical stability) :

Algorithm Modified LU Factorization

m=deg{a(x)}+1
n=deg{b(x)}+1
Construct matrix P
Comment : S is of order m+n
Comment : p is the number of non zero elements which are under the diagonal and which we will zero
p=0
for k=1:deg{b(x)}-1
    p=p+1
    Find r : $|S_{r,k}| = max_{k+1 \leq i \leq k+p}\{|S_{i,k}|\}$
    Interchange rows k and r
    $m_{ik} = s_{ik}/s_{kk}$, i=k+1:k+p
    $s_{ij} = s_{ij} - m_{ik}s_{kj}$, i=k+1:k+p, j=k+1:$P_k$
    Set $s_{i,j} = 0$ if $|s_{i,j}| \leq$ accuracy , i=k:m+n , j=k:m+n

Repeat the previous procedure for k=deg{b(x)} : deg{a(x)} and for constant elements p=deg{b(x)} (under the diagonal which we must zero).

Repeat the previous procedure for k=deg{a(x)}+1:deg{a(x)}+deg{b(x)}-1 and for decreasing per one in every step elements p, starting from p=deg{b(x)} (the elements under the diagonal which we must zero).

Complexity
The complexity of the previous algorithm is $O\ (\frac{5}{6}n^3)$ flops which is much fewer than the $O\ (\frac{8}{3}n^3)$ flops [2] of the general case for a $2n \times 2n$ matrix (we have taken the worse case : m=n=max$\{m, n\}$.

Error Analysis
As we have mentioned we could have catastrophical results during the numerical operations in a floating point arithmetic. Is the previous algorithm stable? The following theorem refers to the stability of this method.

Theorem 2. *Let S(a,b) be a given matrix of order k. If we perform Gaussian elimination with partial pivoting using floating point arithmetic with unit round off u and* $S^{'}(a,b)$ *is the resultant upper triangular matrix, the following relation holds :*

$$L \cdot S^{'}(a,b)=S(a,b)+E,\ \|E\|_\infty \leq (n^2+2n)\cdot \rho u \cdot \|S(a,b)\|_\infty + (\tfrac{n^2}{2}+\tfrac{n}{2})\varepsilon_G$$

where L a lower triangular matrix with units on the diagonal, $\rho = \frac{max_{i,j,k}|s_{ij}^{(k)}|}{max_{i,j}|s_{ij}|}$ *is the growth factor of the Gaussian elimination and* ε_G *is the accuracy of gaussian elimination (elements with absolute value less than* ε_G *will be set equal to zero during the gaussian elimination).* □

So, L·$S^{'}$=S+E, where $E = E^{(1)} + E^{(2)} + \ldots E^{(2n-1)}$ is the sum of the errors at every stage and it is bounded from the previous quantity.

We can use B-scaling at every stage after the first step of the LU factorization in order to keep the absolute values of the elements less or equal to 1, to avoid numerical errors. So the growth factor ρ will be less or equal to 1 and the final error will be :

$$\|\mathrm{E}\|_\infty \leq (n^2+2n)\mathrm{u}\|\mathrm{S}\|_\infty + (\tfrac{n^2}{2}+\tfrac{n}{2})\varepsilon_G.$$

Remark1
Adjusting the error analysis of Gauss to the specific form of matrix S(a,b) we attain as error bound n^2 instead of $4n^2$ that appear if we directly applied the formulas of the general error analysis to an 2n×2n matrix. □

Remark2
Using the above error analysis it can be proved that instead of computing the GCD of a(x) and b(x) we compute the GCD of $a(x)^{'}$ and $b(x)^{'}$, where

$$a(x)^{'} = x^n + a_{n-1}^{'}x^{n-1} + a_{n-2}^{'}x^{n-2} + \ldots + a_3^{'}x^3 + a_2^{'}x^2 + a_1^{'}x + a_0^{'}$$

$$b(x)^{'} = b_n^{'}x^n + b_{n-1}^{'}x^{n-1} + b_{n-2}^{'}x^{n-2} + \ldots + b_3^{'}x^3 + b_2^{'}x^2 + b_1^{'}x + b_0^{'}$$

where $a_i^{'} = a_i + \varepsilon_i$, with $|\varepsilon_i| \leq$(2n-2)gu+(n-1)ε_G , $b_i^{'} = b_i + \varepsilon_i^{'}$, with $|\varepsilon_i^{'}| \leq$2ngu +nε_G. □

3 The Resultant QR-Method

We can accommodate the QR factorization to the previous special form of S^*, reducing thereby the number of the needed flops sufficiently. Every row has some nonzero elements after the diagonal. After these elements there are only zeros. Therefore we can update only the nonzero elements, which are less or equal to m+1 or n+1 in every row. The number of the elements which we must zero at every stage is the same as in the LU-case. After these observations the complexity of the triangulation of S is decreasing a lot. It follows the algorithm of the modified QR factorization:

Algorithm Modified QR Factorization

m=deg{a(x)}+1 , n=deg{b(x)}+1
Construct matrix P
Comment : S is of order m+n and q is the number of non zero elements which are under the diagonal and which we will zero
q=0
for k=1:deg{b(x)}-1
q=q+1
$u_{k:m+n} = zeros_{1,m+n-k+1}$
[u,t]=house1($S_{k:k+q-1,k}$)
w=$u_{k:m+n}$
s_{kk}=t
$r = \sum_{i=1}^{q} w_i^2$
$b = \frac{2}{r}$
sum=$\sum_{i=k}^{k+q-1} u_i \cdot s_{i,j}$, j=k+1:k+P_k
r=b*sum
$s_{i,j} = s_{i,j} - r \cdot u_i$, i=k:k+q-1, j=k+1:k+P_k
Set $s_{i,j} = 0$ if $|s_{i,j}| \leq$ accuracy , i=k:m+n , j=k:m+n

Repeat the previous procedure for k=deg{b(x)}:n-deg{v(x)} and for constant elements q=deg{b(x)}+1 (under the diagonal which we must zero).

Repeat the previous procedure for k=n-deg{b(x)}+1:n-1 and for decreasing per 1 elements in every step, starting from q=deg{b(x)}+1 (the elements under the diagonal which we must zero).

where house1 [2] zeros the entries of the vector x after the first element :

m=max$|x_i|$, i=1,. . .,n
$x_i \equiv u_i = \frac{x_i}{m}$, i=1,. . .,n
t=sign$(u_1)\sqrt{u_1^2 + u_2^2 + \ldots + u_n^2}$
$x_1 \equiv u_1 = u_1 + t$
t=-m·t

Complexity
The previous algorithm consists of three QR-parts. The only difference of each one is the number of elements that is needed to zero. In first part, the number of non-zero elements under the diagonal at step 1 is equal to 1 and it increases until it catches the deg{b(x)}-1. In second part, the number of non-zero elements under the diagonal is constant and equal to deg{b(x)}-1 for n-2deg{b(x)}+1 steps and in third part, the number of non-zero elements under the diagonal is equal to deg{b(x)}-1 and decreases until the triangulation has been completed.

Finally : flops(part1)+flops(part2)+flops(part3)=$O\left(\frac{5}{3}n^3\right)$ flops, which is less enough than the $O(\frac{16}{3}(n)^3)$ flops that requires the classical QR factorization

of an $2n \times 2n$ (we have taken the worse case : m=n=max$\{m, n\}$). The house1 algorithm requires only $O(4(n+1))$ flops.

4 Numerical Results

Numerical results, which were arised applying the classical LU and QR factorization and the modified versions of them to polynomials, are given next. In the tables we compare the four methods. Let ε_{ac} be the accuracy of the Gaussian or QR factorization. This means that values less than ε_{ac} are supposed to be zero.

Example 1

$a(x) = x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 13132x^2 + 13068x - 5040$
$b(x) = x^4 - 28x^3 + 269x^2 - 962x + 720$
$GCD = x - 1$

Method	Relative Error	ε_{ac}	flops
LU	0	10^{-12}	1321
modified LU	0	10^{-12}	561
QR	0	10^{-10}	2298
modified QR	0	10^{-10}	1218

Example 2 [5]

$a(x) = x^{16} - 30x^{15} + 435x^{14} - 4060x^{13} + 27337x^{12} - 140790x^{11} + 573105x^{10} - 1877980x^9 + 4997798x^8 - 10819380x^7 + 18959460x^6 - 26570960x^5 + 29153864x^4 - 24178800x^3 + 14280000x^2 - 5360000x + 960000$
$b(x) = x^{14} - 140x^{12} + 7462x^{10} - 191620x^8 + 2475473x^6 - 15291640x^4 + 38402064x^2 - 25401600$
$GCD = x^4 - 10x^3 + 35x^2 - 50x + 24$

Method	Relative Error	ε_{ac}	flops
LU	$7.6194 \cdot 10^{-9}$	10^{-1}	26341
modified LU	$7.6194 \cdot 10^{-9}$	10^{-1}	6797
QR	$8.0689 \cdot 10^{-7}$	10^{-3}	39278
modified QR	$8.0689 \cdot 10^{-7}$	10^{-3}	17235

Example 3

$a(x) = x^{16} - 13.6x^{15} + 85x^{14} - 323.68x^{13} + 839.402x^{12} - 1569.52x^{11} + 2185.03x^{10} - 2305.72x^9 + 1859.53x^8 - 1146.9x^7 + 537.452x^6 - 188.616x^5 + 48.366x^4 - 8.70777x^3 + 1.02992x^2 - 0.0707343x + 0.00209228$
$b(x) = x - 0.1$
$GCD = x - 0.1$

Method	Relative Error	ε_{ac}	flops
LU	0	10^{-4}	4801
modified LU	0	10^{-4}	496
QR	0	10^{-4}	7724
modified QR	0	10^{-4}	1568

Example 4

$a(x) = x^{16} - 13.6x^{15} + 85x^{14} - 323.68x^{13} + 839.402x^{12} - 1569.52x^{11} + 2185.03x^{10} - 2305.72x^9 + 1859.53x^8 - 1146.9x^7 + 537.452x^6 - 188.616x^5 + 48.366x^4 - 8.70777x^3 + 1.02992x^2 - 0.0707343x + 0.00209228$
$b(x) = x^4 - 2.6x^3 + 1.31x^2 - 0.226x + 0.012$
$GCD = x^3 - 0.6x^2 + 0.11x + 0.006$

Method	Relative Error	ε_{ac}	flops
LU	$2.5663 \cdot 10^{-6}$	10^{-6}	7772
modified LU	$2.5663 \cdot 10^{-6}$	10^{-6}	2096
QR	$2.8106 \cdot 10^{-6}$	10^{-6}	12237
modified QR	$2.9524 \cdot 10^{-6}$	10^{-6}	4078

References

1. S. Barnet, Greatest Common Divisor from Generalized Sylvester Resultant Matrices, *Linear and Multilinear Algebra*, 8 (1980), 271-279.
2. B.N. Datta, Numerical Linear Algebra and Applications, Second Edition, Brooks/Ccole Publishing Company, United States of America, 1995.
3. N. Karcanias, M. Mitrouli and S. Fatouros, A resultant based computation of the GCD of two polynomials, *Proc. of 11th IEEE Mediteranean Conf. on Control and Automation, Rodos Palace Hotel, MED'03, June 18-20, Rhodes, Greece.*
4. M. Mitrouli, N. Karcanias and C. Koukouvinos, Further numerical aspects of ERES algorithm for the computation of the GCD of polynomials and comparison with other existing methodologies, *Utilitas Mathematica*, 50 (1996), 65-84.
5. I.S. Pace and S. Barnet, Comparison of algorithms for calculation of GCD of polynomials, *International Jornal of System Science*, 4 (1973), 211-226.