HEUGCD: How Elementary Upperbounds Generate Cheaper Data

James Davenport
Julian Padget

School of Mathematics
University of Bath
Bath, Avon BA2 7AY
England

1 – Introduction

The work presented in this paper is a direct consequence of the ideas set forth by Char *et al.* [1984b] describing a new technique for computing the greatest common divisor of polynomials. The essence of the algorithm is to project each of the polynomials taken from, say, $Z[x]$ to a corresponding element of Z, compute the *numeric* GCD and thence reconstruct the polynomial GCD by mapping a radix-N representation to a "radix-x" representation. We refer to the polynomial computed this way as the *Z-adic GCD* of the original inputs. This process is not guaranteed to succeed because the Z-adic GCD may be too small, in a sense which will be described later, to permit the *correct* reconstruction, although the theory ensures that the reconstruction will never be smaller than the true GCD. A solution to this is to pick some larger number, related to the original GCD (or, since *ex hypothesi* we do not know this, the original polynomials), and reconstruct at that point. This uncertainty as to the correct reconstruction point gives rise to the term *heuristic* GCD, although taking N (unrealistically) large renders the process deterministic. By appealing to some fundamental inequalities concerning polynomials we are able to improve radically the efficiency of the algorithm and estimate a bound at which the Z-adic computation must produce the correct answer. The sections following are devoted to a discussion of the inequalities used and their application to this particular problem, the extension of this information to include trailing coefficients and the development of our implementation of the algorithm and finally some measurements using the same suite of test problems as used in [Char *et al.* 1984b].

2 – Fundamental Inequalities

We *project* a polynomial, $p(x) \in Z[x]$, into Z by evaluating it at a given point and then we perform integer arithmetic on this representation to obtain a numeric GCD, then reconstruct $g(x)$ such that $g(x) \mid p(x)$ and $g(x) \mid q(x)$. Reconstruction is simply the inverse of the evaluation of a polynomial at a point and can be implemented as a routine which is akin to the inverse of Horner evaluation. Clearly, reconstruction is an ambiguous process - for example do we reconstruct from a value of 0 at $x=N$ the polynomial 0 or the polynomial $x-N$? In order to resolve this ambiguity, we will

normally reconstruct from a value of $x=N$ a polynomial whose coefficients lie in $(-N/2, N/2]$, and refer to this as the *minimal reconstruction* of the value provided.

First we need some notation. Let $p(x) = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_0$ be a polynomial in x. For any n let

$$\|p\|_n = \left[\sum_{i=0}^{d} |a_i|^n \right]^{1/n} .$$

In particular, we can write $\|p\|_\infty = \mathrm{Max}(a_0, \ldots, a_d)$.

We need to ensure that $g(x)$ is the *greatest* common divisor of $p(x)$ and $q(x)$ and not just some divisor. We state Cauchy's inequality [Cauchy, 1829; Mignotte, 1982]:

Proposition 1.

Let $p(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_0, \quad a_d \neq 0, \quad d \geq 1, \qquad (1)$

be a polynomial with complex coefficients. Then any root z of p satisfies

$$|z| < 1 + \frac{\mathrm{Max}\{|a_{d-1}|, \ldots, |a_0|\}}{|a_d|} \leq 1 + \frac{\|p\|_\infty}{|a_d|}$$

Clearly, if the chosen evaluation point approximates a root of either of the polynomials then the related factor will 'disappear' from the Z-adic representation and the reconstructed polynomial will be smaller than one would expect. Since we are seeking to reconstruct a GCD, the factor that disappears must be a factor of both polynomials, i.e. of their GCD. Using Cauchy's inequality we can set a lower bound on the evaluation point which thus ensures we cannot underestimate the polynomial GCD.

From the above we only obtain a lower bound. The reconstruction technique is only capable of producing the correct answer if the evaluation point is greater than $2\|g\|_\infty$, where g is the GCD of p and q. That is, we need a bound on the coefficients of the GCD. This can be found with the help of a generalised Landau-Mignotte inequality [see Landau, 1905; Mignotte 1974], which we now proceed to prove. First we need

Proposition 2. [Landau, 1905; Mignotte, 1982]

Let p be given by (1), where the coefficients can be any complex numbers. Let $z_1, \ldots z_d$ be the roots of p.

$$\text{Let } \quad M(p) = |a_d| \prod_{j=1}^{d} \mathrm{Max}\{1, |z_j|\}, \text{ then } M(p) \leq \|p\|_2 .$$

Theorem 1.

Now let $q(x) = b_e x^e + b_{e-1}x^{e-1} + \cdots + b_0$. $b_e \neq 0$ be a divisor of p above. Then

$$|b_i| \leq C_i^e \left| \frac{b_e}{a_d} \right| \|p\|_2$$

Proof. Since every root of q is a root of p, we see that $M(q)/|b_e| \leq M(p)/|a_d|$, and by Proposition 2 this is less than $\|p\|_2/|a_d|$. But $|b_i/b_e|$ is the sum of ± products of i distinct roots of q, and each product is bounded by $M(q)/|b_e|$. Hence

$$b_i \leq C_i^e M(q) \leq C_i^e \left| \frac{b_e}{a_d} \right| M(p) \leq C_i^e \left| \frac{b_e}{a_d} \right| \|p\|_2$$

3 – Valid Evaluation Points

Theorem 2.

If $N \in Z$ such that

$$|N| \geq 2 + \text{Min}\left(\frac{\|p\|_\infty}{|a_d|}, \frac{\|q\|_\infty}{|b_e|} \right)$$

and $g(x)$ is <u>any</u> reconstruction of $\gcd(p(N), q(N))$ such that $g(x) | p(x)$ and $g(x) | q(x)$ then $g = \gcd(p, q)$.

Proof. Write $G = \gcd(p, q)$, so that we have to show that $g = G$.

If $g \neq G$ then $G = gh$, and so $G(N) = g(N)h(N)$. But, since $G(x) | p(x)$, we know that $G(N) | p(N)$, and similarly that $G(N) | q(N)$, so that $G(N) | \gcd(p(N), q(N)) = g(N)$.

Hence $h(N) = \pm 1$. Now let the ρ_i be the roots of h, so that $h(N) = \Pi(N - \rho_i)$ where ρ_i are the roots of h. Then there is some ρ_i such that $|N - \rho_i| \leq 1$. But ρ_i is a root of $p(x)$ and $q(x)$, and hence is bounded by Proposition 1 (as applied to both p and q).

The important difference between this theorem regarding the choice of evaluation point and the one stated by Char *et al.* [1984b] is the division by the leading coefficient. Two significant advantages arise from this, seemingly simple, change:

(i) the size of the numbers involved is reduced thus reducing the computation time

(ii) there is an important ramification for the problem of computing the GCD of a list of polynomials because we can now use the same evaluation point for *all* the polynomials and reconstruct at the end. The scheme for the choice of

evaluation point given in [Char *et al.* 1984b] requires the application of the method only to pairs of polynomials, thus needing intermediate reconstructions, because the evaluation point is a function of the *heights* of $p(x)$ and $q(x)$ whereas the new method is either a function of the *roots* of $p(x)$ and $q(x)$ or a function of the heights of the *factors* of $p(x)$ and $q(x)$.

4 - Let's try 1/x

The technique of dividing by leading coefficients suggests the idea of including division by the *trailing* coefficient in the choice of evaluation point, that is:

$$|N| \geq 2 + 2\mathrm{Min}\left(\frac{\|p\|_\infty}{|a_0|}, \frac{\|p\|_\infty}{|a_d|}, \frac{\|q\|_\infty}{|b_0|}, \frac{\|q\|_\infty}{|b_e|} \right). \qquad (2)$$

This is indeed possible, although we shall need some notation to demonstrate how. Let $Z[x]^*$ be $Z[x]-xZ[x]$, i.e. the polynomials not divisible by x. Given

$$p = \sum_{i=0}^{d} a_i x^i, \text{ define } p^\sim \text{ as } \sum_{i=0}^{d} a_{d-i} x^i,$$

the *reverse* of p. On $Z[x]^*$, $^\sim$ is an isomorphism, and is indeed an involution, since $p^{\sim\sim}=p$. This is certainly not true on the whole of $Z[x]$, since $1^\sim = x^\sim$.

Furthermore, $^\sim$ preserves multiplication, since $(pq)^\sim = p^\sim q^\sim$, and hence it also preserves division and the computation of greatest common divisors. Furthermore, it certainly preserves $\|\|_\infty$, since that is a property of the coefficients alone, and interchanges leading and trailing coefficients. This indeed proves that we can choose an N as bounded by (2), provided that we evaluate $\gcd(p,q)$ or $\gcd(p^\sim,q^\sim)$ depending on whether the minimum was obtained by means of a leading coefficient or a trailing coefficient. Since we can easily remove any factors of x from p and q before computing the gcd of $p^\sim$ and $q^\sim$, the restriction to $Z[x]^*$ causes no problems.

In fact, of course, we do not actually construct the reverses of the polynomials. We can evaluate the reverse of a polynomial at N, i.e. evaluate the polynomial at $1/N$ and clear the denominator directly, by a process little more complicated than ordinary Horner evaluation, and the same goes for reconstruction.

5 - Systematic Divisors

It is often the case that *no* reconstruction of the integer gcd can give us the true gcd. Take, for example, the coprime polynomials $(x-1)(x-2)$ and $(x+1)(x+2)$. Their gcd is always 1. For any integer N, though, $(N-1)(N-2)$ and $(N+1)(N+2)$ are always both even, and so their gcd is even. And an even number can never reconstruct to yield 1. We call such an integer that is bound to divide the integer gcd (in excess of

that which the polynomial gcd gives) a *systematic divisor* of the polynomials. How do we deal with systematic divisors?

We adopt a similar approach to Char *et al.* [1984b]. Let us first insist that our input polynomials are primitive. Then their gcd has to be primitive. Hence, if the reconstruction of $M=\gcd(p(N),q(N))$ is not primitive, we divide by its content first, thus obtaining the *primitive reconstruction* of M. The effect of this is to insert an extra factor of two in our bounds, as can be seen by comparing Theorem 2 with the following result.

Theorem 3.

If $N \in Z$ such that

$$|N| \geqslant 2 + 2\mathrm{Min}\left(\frac{\|p\|_\infty}{|a_d|}, \frac{\|q\|_\infty}{|b_e|} \right)$$

and $g(x)$ is the primitive reconstruction of $\gcd(p(N),q(N))$ such that $g(x) \mid p(x)$ and $g(x) \mid q(x)$, then $g = \gcd(p,q)$.

Proof. Write $G = \gcd(p,q)$, so that we have to show that $g = G$.

If $g \neq G$ then $G = gh$, and so $G(N) = g(N)h(N)$. But, since $G(x) \mid p(x)$, we know that $G(N) \mid p(N)$, and similarly that $G(N) \mid q(N)$ so that $G(N) \mid \gcd(p(N),q(N)) = cg(N)$, where c is the content that was removed in computing the primitive reconstruction, and is at most $N/2$ since every coefficent of the reconstruction is at most $N/2$.

Hence $h(N) = \pm c$. Now let the ρ_i be the roots of h, so that $h(N) = \Pi(N - \rho_i)$ where ρ_i are the roots of h. Then there is some ρ_i such that $|N - \rho_i| \leqslant c \leqslant N/2$, that is $|\rho_i| \geqslant N/2$. But ρ_i is a root of $p(x)$ and $q(x)$, and hence is bounded by Proposition 1 (as applied to both p and q).

There is a related question: that of the congruence class of N. Consider, for example, $p=x^{\alpha}+x^{\beta}+x^{\gamma}+M$ and $q=x^{\alpha'}+x^{\beta'}+M'$, where M and M' are both divisible by many powers of 2. If we evaluate x at an odd value, we will conclude that p and q have no even factor, and probably that they are relatively prime. However, if we evaluate x at an even factor, we will necessarily deduce that p and q have a large common factor. We could invent cases where even numbers were good and odd numbers bad, and do the same mod 3 etc. The point is that any particular congruence class may well be *unfavourable*. We might call this phenomenon that of *semi-systematic factors*. It might seem that systematic factors and semi-systematic factors could prevent us from ever reconstructing the correct gcd. Char *et al.* [1984b] present arguments to show that this occurs with low probability, decreasing as N increases, but R. Dvornicich has pointed out the following, stronger, result.

Theorem 4. Given two polynomials $p(x)$ and $q(x)$ with gcd $g(x)$, there is a number M such that $|\gcd(p(N),q(N))/g(N)| \leq M$ for all integers N.

Proof. Let $p' = p/g$, $q' = q/g$, so that p' and q' are coprime polynomials. Then $p(N) = g(N)p'(N)$, and similarly for q, so that the value on the left is $|\gcd(p'(N),q'(N))|$. Let M be the resultant $\mathrm{Res}_x(p',q')$. Then there are polynomials $a(x)$ and $b(x)$ in $Z[x]$ such that $a(x)p'(x)+b(x)q'(x) = M$, from the definition of the resultant. Then $a(N)p'(N)+b(N)q'(N) = M$, so that M has to be divisible by $\gcd(p'(N),q'(N))$.

Corollary. If we choose $N > 2M + 2\,\|\gcd(p(x),q(x))\|_\infty$, which can be achieved by choosing $N > 2M + 2\|p\|_2$, then we know that any systematic or semi- systematic divisor will show up as a content, and thus be removed by the process of taking the primitive reconstruction.

Of course, we do not advocate taking this bound - we merely wish to point out that, if the evaluation point is increased indefinitely, we are *guaranteed* to get the correct answer. Hence the phrase *heuristic* in the title refers to the performance of the algorithm, rather than to its (total) correctness.

6 - Where to Evaluate

Theorem 3, (or its analogue for reverses), guarantees that we will never produce the wrong answer (in the sense of saying that g is the gcd when it is not) from the heuristic process. But, if we evaluate at an $N < \|\gcd(p,q)\|_\infty/2$, we will not reconstruct the correct polynomial, but at best one that differs from it by a multiple of N in some coefficient, with a corresponding correction elsewhere.

Essentially this leads to four strategies:

a) the miserly, which evaluates at the smallest possible numbers: ± Cauchy's bound (i.e. Theorem 3), ± (Cauchy's bound +1) ... ;

b) the extravagant, which evaluates at numbers guaranteed to be large enough (assuming that there is no spurious common divisor of the two integers): ± the Landau-Mignotte bound (i.e. Theorem 1), ± (Landau-Mignotte bound +1) ... ;

c) the semi-extravagant, which evaluates at numbers expected to be large enough (assuming that there is no spurious common divisor of the two integers): ± the chosen bound, ± (the chosen bound +1) ... ; for example Char *et al.*

d) the compromise, which attempts to maximise the advantages of method (a) and another method, by starting with small numbers and working up to larger ones.

Method (a) is clearly very expensive if the height of the GCD is much greater than Cauchy's bound, while method (b) is clearly expensive if the height of the GCD is

much less than the Landau-Mignotte bound (one particular case of this is when the GCD is trivial, which is well known to be the worst case for PRS-based code, and the best for modular methods [Davenport, 1981, pp. 136-7]). Method (c) is likely to be cheaper than method (b), but depends on the choice of bound.

We currently adopt method (d), evaluating at Cauchy's bound, then (if that fails) following strategy (c), choosing as our bound a modification of Theorem 1, viz.

Hope 1'.
Let $q = b_e x^e + b_{e-1}x^{e-1} + \cdots + b_0$, $b_e \neq 0$ be a divisor of $p = a_d x^d + a_{d-1}x^{d-1} + \ldots + a_0$. Then quite often

$$|b_t| \leq \left| \frac{b_e}{a_d} \right| \|p\|_\infty,$$

In fact our strategy is slightly more complicated. Let C be Cauchy's bound, and H be the bound given by Hope 1'. Then we evaluate at C, $[H]$, $[He]$, $[He^2]$, $[He^3]$, ..., where the notation $[X]$ means "the number (integral part of) X or $X+1$", chosen to alternate parity with the previous one.

This strategy is not completely different from that adopted by [Char *et al*, 1984b], though their bound is greater (and they do not have the initial Cauchy step), and their multiplying factor is $(3+\sqrt{5})/2$, rather than e.

This question of multiplying factors deserves further attention. The arguments given in Char *et al* [1984b] are *prima facie* convincing, but in fact their choice of multiplying factor has an interesting drawback. The solutions of $a_{n+2}=3a_{n-1}-a_n$ are of the form $a_n = \lambda(3+\sqrt{5})/2 + \mu(3-\sqrt{5})/2$. Since $(3-\sqrt{5})/2<1$, this means that the solution to this equation are, in general, close to multiples of powers of $(3+\sqrt{5})/2$. But, if two consecutive a_i are even, all successive a_i are even, and this seems a bad idea in view of the remarks on semi-systematic factors made earlier.

Since this phenomenon is related to the simplicity of the *minimal equation for* $(3+\sqrt{5})/2$, we decided to choose a transcendental for our multiplying factor, and settled on e since it was close to the factor of Char *et al*, [1984b], and was transcendental with a well-known irregular continued fraction. We were then shocked to discover that the number 20 gave rise to a sequence of eight even numbers, and then added code to ensure that, thereafter, our numbers alternate between odd and even. Clearly, similar problems could arise modulo other primes, but we have not seen such difficulties.

7 - Cost of trial division

It is impossible to separate the cost of an algorithm for polynomials from the polynomial representation being used. Our code was implemented in REDUCE [Hearn,

1982], and used its representation of polynomials, which is essentially "Recursive, Sparse, Variables included" in the terminology of Stoutemyer [1984]. While it is not totally clear, it would appear that MAPLE (in which the work of Char *et al* [1984b] is implemented) uses "Recursive Sparse N-ary Cambridge Prefix" (see Char *et al.*, [1983]). At this stage we are unable to say what effect this might have on the performance of the implementations. REDUCE uses "naive" $O(n^2)$ algorithms for its polynomial arithmetic, and the underlying LISP system [Fitch & Norman, 1977; Padget & Davenport, 1985] uses similarly naive algorithms for its integer arithmetic. We suspect that MAPLE is similar.

The most worrying performance issue in HEUGCD is that of failed reconstructions. If we choose a number N such that the reconstruction of the primitive part of $\gcd(p(n),q(N))$ does not divide both p and q, this experiment can be very expensive: much more so than a successful reconstruction and test division. To appreciate this, consider the cost of seeing if $(x-2)$ divides $(x^{1000}-1)$, which involves constructing numbers as big as 2^{1000}. This means that false reconstructions at large numbers are to avoided where possible, and that trial division can become a major bottleneck for this algorithm.

For this reason, we have investigated various alternative techniques to the obvious trial division algorithm, while we were initially very encouraged by experiments on our own examples, the test data of Char *et al.* [1984b] do not give cause for much joy in this respect, and a wider variety of problems need to be investigated.

8 - Measurements

The major contribution of the improved algorithm is the reduction in size of the intermediate Z-adic representation, in other words the numbers are smaller and so the computation is frequently faster. In order to compare the improved method with the original scheme both versions have been implemented in REDUCE and the same set of test problems, for which timings were presented in [Char *et al.* 1984b], have been run. As remarked earlier, the new algorithm for determining the evaluation point is now a function of the roots in the first instance and directly related to the heights of any divisors in the second instance. This means it is not necessary to reconstruct for each pair of polynomials, rather the complete GCD can be computed in one go. For this reason we have tried to test the new scheme on the GCD of a *list* of polynomials because this should be advantageous. We concur with the findings reported in Char *et al.* [1984b] regarding the effectiveness of the PRS algorithm in REDUCE [Hearn 79] for these problems (note however that trivial GCD's are a known bad case for PRS and that such problems comprise half the test cases). For completeness we include some timings for the distributed REDUCE Hensel-based algorithm which is included in the REDUCE factorisation package [Moore & Norman, 1981].

The main results are given in the columns headed 'HENSEL', 'NEW', 'WAT' and 'PRS'. The problems are those univariate cases taken from [Char *et al.* 1984a]. The notation *m*a means the first polynomial from the m^{th} test case, and *m*b means the second. The first two tables show times for taking the GCD of various pairs of polynomials. The third table gives times for computing the GCD of lists of polynomials. Odd numbered tables were computed on the *Darkstar* 68000-based system (8 MHz) running Cambridge LISP at the University of Bath, whilst the second table was computed on a VAX 11/750 running PSL 3.2 at the Rand Corporation (where we could afford to run the PRS method).

Table 1. **68000 & Cambridge LISP**

Problem	HENSEL	NEW	WAT
1a,1b	20820	840	2660
2a,2b	14700	7000	5520
3a,3b	4920	620	9420
4a,4b	28080	11880	16300

Table 2. **VAX 11/750 & PSL**

Problem	HENSEL	NEW	WAT	PRS
1a,1b	56287	2227	6222	10862966
2a,2b	23664	8755	11118	2436984
3a,3b	8228	1530	23817	8532113
4a,4b	85204	26095	34340	1481363

Table 3. **68000 & Cambridge LISP (lists)**

Problem	HENSEL	NEW	WAT
$1a^2$,1a*1b,$1b^2$	116940	4400	28720
$2a^2$,2a*2b,$2b^2$	216880	81640	79240
$3a^2$,3a*3b,$3b^2$	187560	4400	143520
$4a^2$,4a*4b,$4b^2$	266020	124000	169340

The new version of the algorithm has the clearest advantage in the problems which have trivial GCDs (the first and third results in the above tables), which are also the examples on which Hensel performs best. This is because the first evaluation is taken at the bound given by Cauchy's inequality, which is a small number relative to the bound from Landau's inequality (or the heuristic bound) on the heights of the factors. Even though it is for a small number this initial evaluation is not necessarily computationally cheap and enforces an overhead in those cases where the answer is non-trivial; this partly explains why the new scheme is slower (on the 68000, but not on the VAX) in both versions of problem #2. (Note that the difference is almost the

same in each case, the discrepancy arising from the evaluation of three rather than two polynomials at the low evaluation point). In problem #4, however, this cost is inconsequential in comparison to the gain from using the heuristic Landau bound which renders a much lower evaluation point than the approach taken in Char *et al.* [1984b].

We note that polynomials of the form p^2, pq, and q^2 form the worst case for Hensel, since the cofactors always have factors in common with the GCDs. This explains the very poor figures of the Hensel-like algorithm in the third table.

9 - Conclusion

There is clearly more to the business of computing polynomial GCDs via integer GCDs than meets the eye. We do not pretend to have solved all the problems, but we now have a method that is uniformly cheaper on the VAX, and is never significantly more expensive on the 68000, than that of [Char *et al.*, 1984b], and which is cheaper than Hensel in the cases when Hensel is cheaper than the original method, i.e. relatively prime or nearly relatively prime polynomials. In particular, for problems like $1a^2, 1a*1b, 1b^2$, which are, for differing reasons, very bad cases for PRS, Hensel and the original heuristic method, we have fairly impressive running times.

We hope that other authors will take these questions further, as we intend doing ourselves.

10 - Acknowledgements

Many of the ideas presented above were conceived away from the department. We wish to acknowledge the following for providing stimulating surroundings and pleasant refreshment: Norwood bar, Bath Senior Common Room bar, The Rummer, The Eagle, The Old Spring and La Caverna. We are also grateful to M.J.T. Guy for explaining how stupid we are, and to R. Dvornicich for permission to include his result as Theorem 4.

11 - References

[Cauchy, 1829]

Cauchy, M.A., *Exercices de Mathématiques, Quatrième Année*, Les Oeuvres Complètes d'Augustin Cauchy, IIe Série, Tome IX, (Bibliothèque Nationale, Paris), pp. 122-123.

[Char *et al.*, 1983]

Char, B.W., Geddes, K.O., Gentleman, W.M., and Gonnet, G.H., *The Design of MAPLE: a Compact, Portable and Powerful Computer Algebra System*, Proc. EUROCAL 83 (Springer Lecture Notes in Computer Science 162), pp. 101-115.

[Char *et al.*, 1984a]

Char, B. W., Fee, G. J., Geddes, K. O., Gonnet, G. H., Monagan, M. B., and Watt, S. W., *On the Design and Performance of the MAPLE System.* University of Waterloo, Research Report CS-84-13, June 1984

[Char *et al.*, 1984b]

Char, B. W., Geddes, K. O. & Gonnet, G. H., *GCDHEU: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computation.* Proc. EUROSAM 84 (Springer Lecture Notes in Computer Science 174) pp. 285-296.

[Davenport, 1981]

Davenport, J. H., *On the Integration of Algebraic Functions.* Springer Lecture Notes in Computer Science 102, 1981.

[Fitch & Norman, 1977]

Fitch, J. P. & Norman, A. C., *Implementing LISP in a High-Level Language.* Software - Practice & Expreience 7(1977) pp. 713-725.

[Hearn, 79]

Hearn, A. C., *Non-Modular Computation of Polynomial Gcd Using Trial Division.* Proc EUROSAM 79 (Springer Lecture Notes in Computer Science 72), pp. 227-239.

[Hearn, 82]

Hearn, A. C., *REDUCE - A Case Study in Algebra System Development.* Proc EUROCAM 82 (Springer Lecture Notes in Computer Science 144), pp. 263-272.

[Landau, 1905]

Landau, E., *Sur quelques théorèmes de M. Petrovic rélatifs aux zeros des fonctions analytiques.* Bull. Math. Soc. France 33(1905) pp. 251-261.

[Mignotte, 1974]

Mignotte, M., *An Inequality About Factors of Polynomials.* Math. Comp. 28(1974) pp. 1153-1157.

[Mignotte, 1982]

Mignotte, M., *Some Useful Bounds.* Symbolic & Algebraic Computation (Computing Supplementum 4), Springer-Verlag 1982, pp. 259-263.

[Moore & Norman, 1981]

Moore, P. M. A. & Norman, A. C., *Implementing a Polynomial Factorization and GCD Package.* Proc. SYMSAC 81, ACM, New York, 1981, pp. 109-116.

[Padget & Davenport, 1985]

Padget, J. A. & Davenport, J. H., *Number Bases in an Algebra System.* To appear in SIGSAM Bulletin.

[Stoutemyer, 1984]

Stoutemyer, D. R., *Which Polynomial Representation is Best: Surprises Abound.* Proc. 1984 MACSYMA Users' Conference, General Electric, Schenectady, 1984, pp. 221-243.