International Journal of Pure and Applied Mathematics
Volume 118 No. 3 2018, 713-721
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: 10.12732/ijpam.v118i3.18

ijpam.eu

A NOTE ON EUCLIDEAN AND EXTENDED EUCLIDEAN ALGORITHMS FOR GREATEST COMMON DIVISOR FOR POLYNOMIALS

Anton Iliev[1 §], Nikolay Kyurkchiev[2]
[1,2]Faculty of Mathematics and Informatics
University of Plovdiv Paisii Hilendarski
24, Tzar Asen Str., 4000 Plovdiv, BULGARIA

Abstract: In this note we gave new interpretations of Euclid idea for Greatest Common Divisor for Polynomials (GCDP) and Extended Euclidean Algorithm for Greatest Common Divisor for Polynomials (EEAGCDP). The reason of this interest is wide usage of these algorithms [50], [34]. In our implementation we reduce the number of iterations and now they are 50% of wide spread implementation of Euclidean GCDP and EEAGCDP. In every serious book of algorithms the Euclidean algorithms are part of basic examples [1]-[29], [31]-[50]. Visual C# 2017 programming environment is used.

AMS Subject Classification: 11A05, 68W01
Key Words: greatest common divisor, extended Euclidean greatest common divisor for polynomials, Euclidean algorithm for polynomials, Knuth's algorithm, reduced number of iterations

1. Introduction

Our work is next part of research in [27]-[30].
Euclidean algorithm for polynomials is well known (see [15], [37]):

Received: 2017-12-11
Revised: 2018-04-10
Published: April 15, 2018

§Correspondence author

Algorithm 1.
INPUT: two polynomials $a(x)$ and $b(x)$.
OUTPUT: the greatest common divisor of $a(x)$ and $b(x)$.
1. While $b(x)$!= 0 do the following:
1.1 Set $r(x)\leftarrow a(x)$ mod $b(x)$, $a(x)\leftarrow b(x)$, and $b(x)\leftarrow r(x)$.
2. Return($a(x)$).

Extended Euclidean algorithms for polynomials ([15], [37]) is:
Algorithm 2.
INPUT: two polynomials $a(x)$ and $b(x)$.
OUTPUT: $d(x) = \gcd(a(x), b(x))$ and polynomials $s(x)$, $t(x)$ which satisfy $s(x)a(x) + t(x)b(x) = d(x)$.
1. Set $s2(x)\leftarrow$ 1, $s1$(x)$\leftarrow$ 0, $t2$(x)$\leftarrow$ 0, and $t1$(x)$\leftarrow$ 1.
2. While $b(x)$!= 0 do the following:
2.1 $q(x)\leftarrow a(x)$ div $b(x)$, and $r(x)\leftarrow a(x)$ - $b(x)q(x)$.
2.2 $s(x)\leftarrow s2(x)$ - $q(x)s1(x)$, and $t(x)\leftarrow t2(x)$ - $q(x)t1(x)$.
2.3 $a(x)\leftarrow b(x)$, and $b(x)\leftarrow r(x)$.
2.4 $s2(x)\leftarrow s1(x)$, $s1(x)\leftarrow s(x)$, $t2(x)\leftarrow t1(x)$, and $t1(x)\leftarrow t(x)$.
3. Set $d(x)\leftarrow a(x)$, $s(x)\leftarrow s2(x)$, and $t(x)\leftarrow t2(x)$.
4. Return($d(x)$,$s(x)$,$t(x)$).

2. Main Results

Now we set the task to optimize Euclidean GCDP algorithm and EEAGCDP. For testing we will use the following computer: processor - Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz, 2592 Mhz, 4 Core(s), 8 Logical Processor(s), RAM 16 GB, Microsoft Windows 10 Enterprise x64.

with the following programming environment (see Fig. 1.).

We suggest the following algorithms.
Algorithm 3.
INPUT: two polynomials $a(x)$ and $b(x)$.
OUTPUT: the greatest common divisor of $a(x)$ and $b(x)$.
1a. If degree of $a(x)$ is greater than degree of $b(x)$. While (true) do the following:
1a.1 set $r(x)\leftarrow a(x)$ mod $b(x)$.
1a.2 If $r(x) = 0$ set $\gcd(x) = b(x)$, and break.

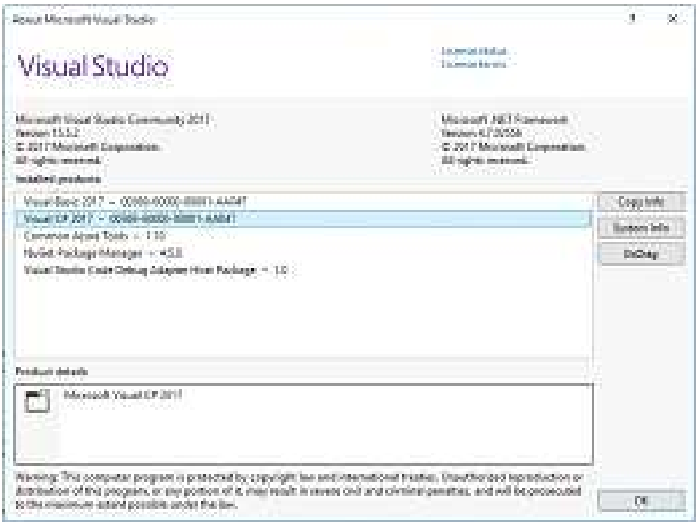

Figure 1: Visual C# 2017

1a.3 set $r1(x) \leftarrow b(x)$ mod $r(x)$.
1a.4 If $r1(x) = 0$ set gcd$(x) = r(x)$, and break.
1a.5 set $a(x) \leftarrow r(x)$, and $b(x) \leftarrow r1(x)$.
1b. [else] If degree of $b(x)$ is greater than or equal to the degree of $a(x)$. While (true) do the following:
1b.1 set $r(x) \leftarrow b(x)$ mod $a(x)$.
1b.2 If $r(x) = 0$ set gcd$(x) = a(x)$, and break.
1b.3 set $r1(x) \leftarrow a(x)$ mod $r(x)$.
1b.4 If $r1(x) = 0$ set gcd$(x) = r(x)$, and break.
1b.5 set $b(x) \leftarrow r(x)$, and $a(x) \leftarrow r1(x)$.
2. [Make monic] Set c != 0 as the leading coefficient of gcd(x);
$d(x) = c$^{-1}gcd(x);
Return$(d(x))$.

Algorithm 4.
INPUT: two polynomials $a(x)$ and $b(x)$.
OUTPUT: $d(x) = \gcd(a(x), b(x))$ and polynomials $s(x)$, $t(x)$ which satisfy $s(x)a(x) + t(x)b(x) = d(x)$.
1. Set $ao(x) = a(x)$, and $bo(x) = b(x)$.

2a. If degree of $a(x)$ is greater than degree of $b(x)$. Set $s2(x) \leftarrow 1$, and $s1$(x)$\leftarrow$ 0. While (true) do the following:
2a.1 $q(x) \leftarrow a(x)$ div $b(x)$, and $r(x) \leftarrow a(x)$ - $b(x)q(x)$.
2a.2 $s(x) \leftarrow s2(x)$ - $q(x)s1(x)$, $s2(x) \leftarrow s1(x)$, and $s1(x) \leftarrow s(x)$.
2a.3 If $r(x) = 0$ then set $d(x) \leftarrow b(x)$, $s(x) \leftarrow s2(x)$, $t(x) \leftarrow (b(x) - s(x)ao(x))bo^{-1}(x)$, and break.
2a.4 $q(x) \leftarrow b(x)$ div $r(x)$, and $r1(x) \leftarrow b(x)$ - $r(x)q(x)$.
2a.5 $s(x) \leftarrow s2(x)$ - $q(x)s1(x)$, $s2(x) \leftarrow s1(x)$, and $s1(x) \leftarrow s(x)$.
2a.6 If $r1(x) = 0$ then set $d(x) \leftarrow a(x)$, $s(x) \leftarrow s2(x)$, $t(x) \leftarrow (a(x) - s(x)ao(x))bo^{-1}(x)$, and break.
2a.7 $a(x) \leftarrow r(x)$, and $b(x) \leftarrow r1(x)$.
2b. [else] If degree of $b(x)$ is greater than or equal to the degree of $a(x)$. Set $s2(x) \leftarrow 0$, and $s1$(x)$\leftarrow$ 1. While (true) do the following:
2b.1 $q(x) \leftarrow b(x)$ div $a(x)$, and $r(x) \leftarrow b(x)$ - $a(x)q(x)$.
2b.2 $s(x) \leftarrow s2(x)$ - $q(x)s1(x)$, $s2(x) \leftarrow s1(x)$, and $s1(x) \leftarrow s(x)$.
2b.3 If $b(x) = 0$ then set $d(x) \leftarrow a(x)$, $s(x) \leftarrow s2(x)$, $t(x) \leftarrow (a(x) - s(x)ao(x))bo^{-1}(x)$, and break.
2b.4 $q(x) \leftarrow a(x)$ div $r(x)$, and $r1(x) \leftarrow a(x)$ - $r(x)q(x)$.
2b.5 $s(x) \leftarrow s2(x)$ - $q(x)s1(x)$, $s2(x) \leftarrow s1(x)$, and $s1(x) \leftarrow s(x)$.
2b.6 If $a(x) = 0$ then set $d(x) \leftarrow b(x)$, $s(x) \leftarrow s2(x)$, $t(x) \leftarrow (b(x)$ - $s(x)ao(x))bo^{-1}(x)$, and break.
2b.7 $b(x) \leftarrow r(x)$, and $a(x) \leftarrow r1(x)$.
3. [Make monic] Set c !=0 as the leading coefficient of $d(x)$.
$(d(x), s(x), t(x)) = (c^{-1}d(x), c^{-1}s(x), c^{-1}t(x))$.
Return($d(x)$, $s(x)$, $t(x)$).

The asymptotic of number of divisions of Knuth's revision of Euclid's GCD is known [34], [40] using CAS Mathematica here we will seek approximation of the data where first coordinate of every point is N and second coordinate is average CPU time in seconds. We will use the example given in [15]: $a(x) =$ $7x$^11 + x^9 + $7x$^2 + 1, $b(x) =$ -$7x$^7 - x^5 + $7x$^2 + 1. The gcd$(x) =$ $d(x)$ is x^2+ 1/7. We will solve this example up to 100 000 000 times using classical algorithm 1 and new algorithm 3. We calculate the CPU time taken by algorithms 1 and 3. Data1 are data taken from Euclidean algorithm [15], [37] and data2 are data which we received from new algorithm 3. The reader can be convinced of the benefits of the new method (see Fig. 2).
data1:={{1000000,0.944},{2000000,1.527},{3000000,2.281},
{4000000,3.015},{5000000,3.761},{6000000,4.546},
{7000000,5.301},{8000000,6.063},{9000000,6.806},

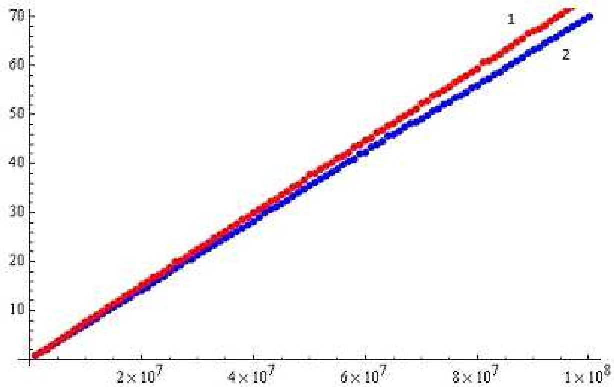

Figure 2: Euclid algorithm (red line - 1) and Iliev-Kyurkchiev algorithm (blue line - 2)

{10000000,7.667},{11000000,8.327},{12000000,9.136},
{13000000,9.797},{14000000,10.582},{15000000,11.342},
{16000000,12.095},{17000000,12.93},{18000000,13.646},
{19000000,14.367},{20000000,15.259},{21000000,15.888},
{22000000,16.817},{23000000,17.236},{24000000,17.887},
{25000000,18.99},{26000000,19.968},{27000000,20.293},
{28000000,21.037},{29000000,21.789},{30000000,22.447},
{31000000,23.134},{32000000,23.932},{33000000,24.794},
{34000000,25.537},{35000000,26.257},{36000000,27.033},
{37000000,27.748},{38000000,28.58},{39000000,29.194},
{40000000,29.992},{41000000,30.724},{42000000,31.427},
{43000000,32.216},{44000000,32.887},{45000000,33.718},
{46000000,34.32},{47000000,35.197},{48000000,35.819},
{49000000,36.709},{50000000,37.694},{51000000,38.123},
{52000000,38.867},{53000000,39.627},{54000000,40.253},
{55000000,41.116},{56000000,41.624},{57000000,42.351},
{58000000,43.436},{59000000,43.887},{60000000,44.758},
{61000000,45.376},{62000000,46.518},{63000000,46.949},
{64000000,47.748},{65000000,48.338},{66000000,49.061},
{67000000,49.738},{68000000,50.609},{69000000,51.277},
{70000000,52.313},{71000000,52.729},{72000000,53.912},
{73000000,54.431},{74000000,55.041},{75000000,55.782},
{76000000,56.585},{77000000,57.268},{78000000,58.002},

{79000000,58.843},{80000000,59.517},{81000000,60.784},
{82000000,60.995},{83000000,61.737},{84000000,62.459},
{85000000,63.203},{86000000,63.999},{87000000,64.789},
{88000000,65.456},{89000000,66.671},{90000000,67.156},
{91000000,67.702},{92000000,68.405},{93000000,69.187},
{94000000,69.904},{95000000,70.647},{96000000,71.471},
{97000000,72.129},{98000000,72.93},{99000000,73.723},
{100000000,74.297}};

data2:={{1000000,0.897},{2000000,1.49},{3000000,2.128},
{4000000,2.894},{5000000,3.576},{6000000,4.256},
{7000000,4.981},{8000000,5.76},{9000000,6.355},
{10000000,7.092},{11000000,7.813},{12000000,8.505},
{13000000,9.232},{14000000,9.925},{15000000,10.652},
{16000000,11.463},{17000000,12.069},{18000000,12.766},
{19000000,13.662},{20000000,14.193},{21000000,14.882},
{22000000,15.625},{23000000,16.368},{24000000,16.994},
{25000000,17.883},{26000000,18.669},{27000000,19.378},
{28000000,20.212},{29000000,20.569},{30000000,21.316},
{31000000,21.98},{32000000,22.838},{33000000,23.516},
{34000000,24.124},{35000000,24.814},{36000000,25.482},
{37000000,26.264},{38000000,26.877},{39000000,27.568},
{40000000,28.293},{41000000,29.018},{42000000,29.931},
{43000000,30.806},{44000000,31.171},{45000000,31.944},
{46000000,32.622},{47000000,33.354},{48000000,34.031},
{49000000,34.805},{50000000,35.409},{51000000,36.163},
{52000000,36.91},{53000000,37.574},{54000000,38.207},
{55000000,38.966},{56000000,39.764},{57000000,40.421},
{58000000,41.021},{59000000,42.038},{60000000,42.406},
{61000000,43.397},{62000000,43.87},{63000000,44.609},
{64000000,45.846},{65000000,46.01},{66000000,46.639},
{67000000,47.489},{68000000,48.171},{69000000,48.529},
{70000000,49.15},{71000000,49.81},{72000000,50.766},
{73000000,51.203},{74000000,52.102},{75000000,52.628},
{76000000,53.546},{77000000,53.964},{78000000,54.675},
{79000000,55.551},{80000000,56.085},{81000000,56.886},
{82000000,57.418},{83000000,58.273},{84000000,58.794},
{85000000,59.71},{86000000,60.223},{87000000,61.003},
{88000000,61.615},{89000000,62.571},{90000000,63.361},

{91000000,63.801},{92000000,64.632},{93000000,65.309},
{94000000,65.803},{95000000,66.673},{96000000,67.276},
{97000000,68.174},{98000000,68.708},{99000000,69.536},
{100000000,70.216}};

We can conclude that Algorithms 3 and 4 are faster than the Algorithms 1 and 2 respectively because we reduce some computational operations.

Acknowledgments

This work has been supported by the project FP17-FMI008 of Department for Scientific Research, Paisii Hilendarski University of Plovdiv.

References

[1] A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, 1st ed., Addison-Wesley, Boston (1974).

[2] A. Aho, J. Ullman, J. Hopcroft, *Data Structures and Algorithms*, 1st ed., Addison-Wesley, Boston (1987).

[3] A. Akritas, A new method for computing polynomial greatest common divisors and polynomial remainder sequences, *Numerische Mathematik*, 52 (1988), 119-127.

[4] A. Akritas, G. Malaschonok, P. Vigklas, On the Remainders Obtained in Finding the Greatest Common Divisor of Two Polynomials, *Serdica Journal of Computing*, 9 (2015), 123-138.

[5] M. Alsuwaiyel, *Algorithms: Design Techniques and Analysis*, Lecture Notes Series on Computing, revised ed., World Scientific Publishing Company, Hackensack (2016).

[6] L. Ammeraal, *Algorithms and Data Structures in C++*, John Wiley & Sons Inc., New York (1996).

[7] T. M. Apostol, *Introduction to Analytic Number Theory*, Springer-Verlag, New York (1976).

[8] S. Baase, A. Gelder, *Computer Algorithms, Introduction to Design and Analysis*, 3rd ed., Addison-Wesley, Boston (2000).

[9] G. Brassard, P. Bratley, *Fundamentals of Algorithmics*, international ed., Pearson, (2015).

[10] D. Bressoud, *Factorization and primality testing*, Springer Verlag, New York (1989).

[11] F. Chang, Factoring a Polynomial with Multiple-Roots, *World Academy of Science, Engineering and Technology*, 47 (2008), 492-495.

[12] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer, New York (1996).

[13] Th. Cormen, *Algorithms Unlocked*, MIT Press, Cambridge (2013).

[14] Th. Cormen, Ch. Leiserson, R. Rivest, Cl. Stein, *Introduction to Algorithms*, 3rd ed., The MIT Press, Cambridge (2009).

[15] R. Crandall, C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer-Verlag, New York (2005).

[16] J. D. Dixon, The number of steps in the Euclidean algorithm, *J. Number Theory*, 2 (1970), 414-422.

[17] A. Drozdek, *Data Structures and Algorithms in C++*, 4th ed., Cengage Learning, Boston (2013).

[18] J. Erickson, *Algorithms*, University of Illinois Press (2009).

[19] J. Gareth, J. Jones, *Elementary Number Theory*, Springer-Verlag, New York (1998).

[20] K. Garov, A. Rahnev, *Textbook-notes on programming in BASIC for facultative training in mathematics for 9.-10. grade of ESPU*, Sofia (1986). (in Bulgarian)

[21] S. Goldman, K. Goldman, *A Practical Guide to Data Structures and Algorithms Using JAVA*, Chapman & Hall/CRC, Taylor & Francis Group, New York (2008).

[22] A. Golev, *Textbook on algorithms and programs in C#*, University Press "Paisii Hilendarski", Plovdiv (2012).

[23] M. Goodrich, R. Tamassia, D. Mount, *Data Structures and Algorithms in C++*, 2nd ed., John Wiley & Sons Inc., New York (2011).

[24] R. Graham, D. Knuth, O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, 2nd ed., Addison-Wesley, Boston (1994).

[25] D. H. Greene, D. E. Knuth, *Mathematics for the Analysis of Algorithms*, 2nd ed., Birkhauser, Boston (1982).

[26] H. A. Heilbronn, On the average length of a class of finite continued fractions. In: *Number Theory and Analysis* (Turan, P., ed.), 87-96, Plenum Press, New York (1969).

[27] A. Iliev, N. Kyurkchiev, A Note on Knuth's implementation of Euclid's Greatest Common Divisor Algorithm, *International Journal of Pure and Applied Mathematics*, 117 (2017), 603-608.

[28] A. Iliev, N. Kyurkchiev, A. Golev, A Note on Knuth's implementation of Extended Euclidean Greatest Common Divisor Algorithm, *International Journal of Pure and Applied Mathematics*, 118 (2018), 31-37.

[29] A. Iliev, N. Kyurkchiev, A. Rahnev, A Note on Adaptation of the Knuth's Extended Euclidean Algorithm for Computing Multiplicative Inverse, *International Journal of Pure and Applied Mathematics*, 118 (2018), 281-290.

[30] A. Iliev, N. Valchanov, T. Terzieva, Generalization and Optimization of Some Algorithms, *Collection of scientific works of National Conference "Education in Information Society"*, Plovdiv, ADIS, May 12-13, (2009), 52-58 (in Bulgarian), http://sci-gems.math.bas.bg/jspui/handle/10525/1356

[31] E. Kaltofen, H. Rolletschek, Computing greatest common divisors and factorizations in quadratic number fields, *Math. Comp.*, 53 (1990), 697-720.

[32] J. Kleinberg, E. Tardos, *Algorithm Design*, Addison-Wesley, Boston (2006).

[33] D. E. Knuth, Evaluation of Porters constant, *Comp. Maths. Appls.*, 2 (1976), 137-139.

[34] D. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Boston (1998).

[35] Hr. Krushkov, *Programming in C#*, Koala press, Plovdiv (2017). (in Bulgarian)

[36] A. Levitin, *Introduction to the Design and Analysis of Algorithms*, 3rd ed., Pearson, Boston (2011).

[37] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, 5th ed., CRC Press LLC, New York (2001).

[38] P. Nakov, P. Dobrikov, *Programming =++Algorithms*, 5th ed., Sofia (2015). (in Bulgarian)

[39] G. H. Norton, A shift-remainder GCD algorithm. In: *Applied Algebra. Algebraic Algorithms and Error Correcting Codes* (Huguet, L., Poli, A., eds.), Springer LNCS, 356 (1989), 350-356.

[40] G. H. Norton, On the Asymptotic Analysis of the Euclidean Algorithm, *J. Symbolic Computation*, 10 (1990), 53-58.

[41] J. W. Porter, On a theorem of Heilbronn, *Mathematika*, 22 (1975), 20-28.

[42] A. Rahnev, K. Garov, O. Gavrailov, *Textbook for extracurricular work using BASIC*, MNP Press, Sofia (1985). (in Bulgarian)

[43] A. Rahnev, K. Garov, O. Gavrailov, *BASIC in examples and tasks*, Government Press "Narodna prosveta", Sofia (1990). (in Bulgarian)

[44] H. Rolletschek, On the number of divisions of the Euclidean algorithm applied to Gaussian integers, *J. Symbolic Computation*, 2 (1986), 261-291.

[45] H. Rolletschek, Shortest division chains in imaginary quadratic number fields. In: *Symbolic and Algebraic Computation* (Gianni, P., ed.). Springer LNCS 358 (1990), 231-243.

[46] D. Schmidt, *Euclid's GCD Algorithm* (2014).

[47] R. Sedgewick, K. Wayne, *Algorithms*, 4th ed., Addison-Wesley, Boston (2011).

[48] S. Skiena, *The Algorithm Design Manual*, 2nd ed., Springer, New York (2008).

[49] A. Stepanov, *Notes on Programming* (2007).

[50] E. Weisstein, *CRC Concise Encyclopedia of Mathematics*, Chapman & Hall/CRC, New York (2003).