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 widespread 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]):
Algorithm 1.
INPUT: two polynomials $a(x)$ and $b(x)$.
OUTPUT: the greatest common divisor of $a(x)$ and $b(x)$.
1. While $b(x) \neq 0$ 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 $s_2(x) \leftarrow 1$, $s_1(x) \leftarrow 0$, $t_2(x) \leftarrow 0$, and $t_1(x) \leftarrow 1$.
2. While $b(x) \neq 0$ 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 $s(x) \leftarrow s_2(x) - q(x)s_1(x)$, and $t(x) \leftarrow t_2(x) - q(x)t_1(x)$.
   2.3 $a(x) \leftarrow b(x)$, and $b(x) \leftarrow r(x)$.
   2.4 $s_2(x) \leftarrow s_1(x)$, $s_1(x) \leftarrow s(x)$, $t_2(x) \leftarrow t_1(x)$, and $t_1(x) \leftarrow t(x)$.
3. Set $d(x) \leftarrow a(x)$, $s(x) \leftarrow s_2(x)$, and $t(x) \leftarrow t_2(x)$.
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.
1a.3 set \( r_1(x) \leftarrow b(x) \mod r(x) \).
1a.4 If \( r_1(x) = 0 \) set \( \gcd(x) = r(x) \), and break.
1a.5 set \( a(x) \leftarrow r(x) \), and \( b(x) \leftarrow r_1(x) \).
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 \( r_1(x) \leftarrow a(x) \mod r(x) \).
1b.4 If \( r_1(x) = 0 \) set \( \gcd(x) = r(x) \), and break.
1b.5 set \( b(x) \leftarrow r(x) \), and \( a(x) \leftarrow r_1(x) \).
2. [Make monic] Set \( c \neq 0 \) 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 \( a_0(x) = a(x) \), and \( b_0(x) = b(x) \).
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 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 \( \text{gcd}(x) = d(x) \) 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)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000000</td>
<td>7.667</td>
<td></td>
</tr>
<tr>
<td>1300000</td>
<td>9.797</td>
<td></td>
</tr>
<tr>
<td>1600000</td>
<td>12.095</td>
<td></td>
</tr>
<tr>
<td>1900000</td>
<td>14.367</td>
<td></td>
</tr>
<tr>
<td>2200000</td>
<td>16.817</td>
<td></td>
</tr>
<tr>
<td>2500000</td>
<td>18.99</td>
<td>19.968</td>
</tr>
<tr>
<td>2800000</td>
<td>21.037</td>
<td>21.789</td>
</tr>
<tr>
<td>3100000</td>
<td>23.134</td>
<td>23.932</td>
</tr>
<tr>
<td>3400000</td>
<td>25.537</td>
<td>26.257</td>
</tr>
<tr>
<td>3700000</td>
<td>27.748</td>
<td>28.58</td>
</tr>
<tr>
<td>4000000</td>
<td>29.992</td>
<td>30.724</td>
</tr>
<tr>
<td>4300000</td>
<td>32.216</td>
<td>32.887</td>
</tr>
<tr>
<td>4600000</td>
<td>34.32</td>
<td>35.197</td>
</tr>
<tr>
<td>4900000</td>
<td>36.709</td>
<td>37.694</td>
</tr>
<tr>
<td>5200000</td>
<td>38.867</td>
<td>39.627</td>
</tr>
<tr>
<td>5500000</td>
<td>41.116</td>
<td>41.624</td>
</tr>
<tr>
<td>5800000</td>
<td>43.436</td>
<td>43.887</td>
</tr>
<tr>
<td>6100000</td>
<td>45.376</td>
<td>46.518</td>
</tr>
<tr>
<td>6400000</td>
<td>47.748</td>
<td>48.338</td>
</tr>
<tr>
<td>6700000</td>
<td>49.738</td>
<td>50.609</td>
</tr>
<tr>
<td>7000000</td>
<td>52.313</td>
<td>52.729</td>
</tr>
<tr>
<td>7300000</td>
<td>54.431</td>
<td>55.041</td>
</tr>
<tr>
<td>7600000</td>
<td>56.585</td>
<td>57.268</td>
</tr>
</tbody>
</table>
data2:=

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

\[
\begin{align*}
&\{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\}, \\
&\{52000000,36.961\}, \{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\},
\end{align*}
\]
\{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


